Capítulo 1

Preliminares

La explosión de datos digitales ha generado una cantidad masiva de información disponible en línea, lo que ha impulsado el desarrollo de Sistemas de Recomendación (SR) personalizados. Estos sistemas utilizan técnicas de aprendizaje automático para sugerir productos, servicios o contenidos relevantes a los usuarios en función de sus preferencias y comportamientos previos. Los SR son ampliamente utilizados en aplicaciones en línea, como tiendas, plataformas de video en línea, redes sociales y más.

Los métodos existentes para los SR se pueden clasificar en tres grandes categorías: métodos basados en contenido, métodos de Filtrado Colaborativo (FC) y métodos híbridos. Los métodos basados en contenido hacen uso de los perfiles de los usuarios o las descripciones de los artículos, recomendando artículos similares a aquellos que al usuario le han gustado en el pasado (Lang, 1995; Li, Cheung, y She, 2016; Pazzani y Billsus, 1997). En contraste, los métodos basados en FC recomiendan a un usuario artículos que han sido apreciados por otros usuarios con preferencias similares. Billsus, Pazzani, y cols. (1998) y Mnih y Salakhutdinov (2007) son ejemplos de enfoques de FC basados en el historial del usuario, como calificaciones previas, sin emplear información adicional sobre los artículos.

Una de las limitaciones del FC es que el rendimiento de la recomendación disminuye significativamente cuando las calificaciones son escasas. Además, el FC no puede recomendar nuevos elementos que no hayan recibido ninguna calificación, lo que se conoce como el problema de arranque en frío (Mnih y Salakhutdinov, 2007). Por último, los métodos híbridos combinan técnicas basadas en contenido y FC, buscando aprovechar lo mejor de ambos enfoques (C. Wang y Blei, 2011). En esta tesis, utilizamos un enfoque híbrido que explota información sobre usuarios, ítems y la relación entre ellos.

Por otro lado, en el ámbito del aprendizaje automático, los modelos generativos han emergido como una herramienta fundamental para modelar la distribución de los datos observados. Estos modelos buscan describir cómo se generan los datos a partir de variables latentes no observadas, proporcionando así un marco probabilístico robusto para el análisis y síntesis de datos (Kingma y Welling, 2014). En este contexto, los Autoencoders Variacionales (VAEs) se presentan como una extensión probabilística de los Autoencoders (AE) tradicionales, utilizando un enfoque bayesiano para modelar la incertidumbre en los datos mediante inferencia variacional.

El modelado probabilístico de un VAE se basa en la relación entre las variables observadas x y variables latentes z, expresada mediante la distribución conjunta:

$$p_\theta(x, z) = p_\theta(x \mid z)p(z), \tag{1.1}$$

donde $p(z)$ es la priori sobre las variables latentes y define el proceso de generación de datos. Sin embargo, dado que las variables latentes no son observables, la inferencia requiere calcular la distribución posterior:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)p(z)}{p_\theta(x)}, \tag{1.2}$$

pero el cálculo de la marginal resulta intratable:

$$p_\theta(x) = \int p_\theta(x \mid z)p(z)dz. \tag{1.3}$$

Para abordar esta intractabilidad, los VAEs emplean inferencia variacional aproximando $p(z|x)$ mediante una distribución paramétrica, minimizando la divergencia de Kullback-Leibler (KL):

$$D_{KL}(q_\phi(z \mid x) \parallel p_\theta(z \mid x)). \tag{1.4}$$

Para estimar los parámetros, en lugar de optimizar la log-verosimilitud, se usa una cota inferior, conocida como *Evidence Lower Bound* (ELBO, por sus siglas en inglés):

$$\log p_\theta(x) \geq \mathbb{E}q_\phi(z \mid x)[\log p_\theta(x \mid z)] - D_{KL}(q_\phi(z \mid x) \parallel p(z \mid x)). \tag{1.5}$$

El primer término de ELBO es la log-verosimilitud esperada, mientras que el segundo término regulariza la aproximación a la distribución latente prior. Esto permite que la optimización de ELBO se interprete como una función de costo con un término de regularización (Liang, Krishnan, Hoffman, y Jebara, 2018). Se dará más detalles en el capítulo 2.

En el contexto de recomendaciones, con cada usuario u se asocia un vector latente z_u, que captura sus preferencias y permite predecir la probabilidad de interacción con los distintos ítems. En nuestro problema de recomendación, la observación x_u se modela como una muestra de una distribución multinomial con probabilidad $\pi(z_u)$, y para la estimación de los parámetros, en lugar de usar directamente la log-verosimilitud de los datos, la cual es intratable debido a la integral sobre las variables latentes, trabajamos con la ecuación (1.5) del ELBO. Esta elección se fundamenta en el principio bayesiano de incorporar incertidumbre sobre las variables latentes, y permite estimar los parámetros del modelo generativo θ y de la red de inferencia ϕ mediante gradiente estocástico, sin necesidad de evaluar directamente la verosimilitud marginal (Yang, 2007).

Para optimizar ELBO, es necesario calcular gradientes a través de la distribución latente z, lo que se logra con lo que se conoce como el truco de reparametrización, explicada más adelante en la sección 2.4. En este enfoque, en lugar de muestrear directamente de $q_\phi(z \mid x)$, se reexpresa z en términos de una variable auxiliar $\epsilon \sim \mathcal{N}(0, I)$:

$$z = \mu_\phi(x) + \sigma_\phi(x) \odot \epsilon, \tag{1.6}$$

donde $\mu_\phi(x)$ y $\sigma_\phi(x)$ representan la media y desviación estándar, obtenidas a partir de los datos de entrada, y $\odot$ denota el producto elemento a elemento (Liang y cols., 2018).

En esta tesis, para generar recomendaciones proponemos una variante de modelos locales basada en Autoencoders Variacionales Multinomiales (MultVAE), inspirada en los Local Collaborative Autoencoders (LOCA) propuestos en Choi, Jeong, Lee, y Lee (2021). Este enfoque busca capturar tanto patrones locales como globales en los datos, logrando personalizar las recomendaciones.

1.1. Objetivo

En este trabajo de tesis se explora cómo aprovechar información adicional tanto sobre los usuarios como sobre el contenido de las películas en el contexto de modelos locales con Autoencoders Variacionales Multinomiales (Mult-VAE) y que usarán inferencia variacional para la estimación de parámetros (Liang y cols., 2018). Estos modelos locales logran capturar patrones latentes no lineales de interacción usuario-ítem dentro de subcomunidades específicas, maximizando la cobertura mediante una combinación de enfoques locales y una mayor diversidad de modelos para captar mejor las preferencias del usuario.

Para lograr este objetivo, se propone un Sistema de Recomendación (SR) de películas que predice la probabilidad de interacción de un usuario con respecto a cada

ítem. Esta probabilidad se obtendrá mediante una variante de métodos locales que hacen uso de autoencoders variacionales multinomiales para capturar patrones latentes en los datos. Al investigar el potencial de esta técnica para modelar la incertidumbre en los datos, se espera contribuir al avance de la investigación en SR y mejorar la experiencia del usuario en aplicaciones en línea.

1.2. Punto de partida del problema

Los datos utilizados para entrenar los modelos de SR pueden ser de diferentes tipos. Los tipos más comunes incluyen matrices de calificaciones explícitas, matrices de interacción usuario-ítem, características de usuarios o ítems, y datos de retroalimentación implícita, como el tiempo de permanencia en una página web, la frecuencia de visitas o los patrones de navegación. En nuestro caso, partimos de una matriz binaria de interacciones entre usuarios e ítems, $\mathbb{X} \in \{0,1\}^{U \times I}$, donde cada entrada x_{ui} representa la interacción entre el usuario u y el ítem i, con $u \in \{1,\ldots,U\}$ e $i \in \{1,\ldots,I\}$. Las entradas x_{ui} toman el valor de 1 si hubo interacción entre el usuario u y el ítem i, y 0 si no hubo interacción. El vector $x_u = [x_{u1}, x_{u2}, \ldots, x_{uI}]^\top$ contiene el historial de interacciones del usuario u con todos los ítems.

Dada esta matriz de interacción $\mathbb{X}$, el objetivo es generar una lista clasificada con el top-N ítems que tienen mayor probabilidad de ser de interés para el usuario u. Para lograrlo, se entrena un modelo de recomendación $M(\mathbb{X};\theta) : \{0,1\}^{U \times I} \to \mathbb{R}^{U \times I}$, parametrizado por θ, que infiere una interacción $\hat{x}_{ui}$ entre el usuario u y el ítem i. El modelo produce probabilidades de interacción, y se seleccionan las N mayores para conformar el conjunto de ítems recomendados. Para el caso de un solo usuario, usaremos la notación $M(x_u;\theta) : \{0,1\}^I \to \mathbb{R}^I$, donde x_u es una fila de la matriz de interacciones $\mathbb{X}$.

Aunque el problema de recomendación es comúnmente clasificado como un problema de big data, en realidad la cantidad de interacciones observadas es relativamente

pequeña. Esto se debe a que la mayoría de los usuarios interactúan únicamente con una fracción reducida de los ítems disponibles, lo que da como resultado una matriz de interacciones $\mathbb{X}$ que es sparse y binaria. Esta escasez de interacciones plantea desafíos en la capacidad del modelo para hacer predicciones precisas.

Además de las interacciones, también contamos con información adicional descriptiva sobre los usuarios y los ítems, la cual nos permite inferir las preferencias individuales de los usuarios. Nuestro enfoque se centrará en aprovechar tanto las interacciones usuario-ítem como las características adicionales de los usuarios y los ítems para mejorar las predicciones, particularmente en el contexto de la recomendación del top-N ítems.

Un desafío central en los SR es la representación eficiente de usuarios e ítems en un espacio de menor dimensión. Dado que las interacciones suelen ser dispersas, es deseable capturar las relaciones subyacentes de manera más compacta. Los modelos de factores latentes, que proyectan tanto usuarios como ítems en espacios de baja dimensión, han surgido como una solución efectiva. Estos modelos asumen que las preferencias de un usuario pueden ser representadas como una combinación ponderada de factores latentes no observados.

En la siguiente sección, se detallan diferentes métodos que se emplean comúnmente en los SR, todos se basan en métodos de reducción de dimensión para extraer factores latentes como: la factorización de matrices, factorización probabilística y análisis factorial.

1.3. Métodos populares de reducción de dimensión para SR

Los SR trabajan con conjuntos de datos complejos y voluminosos, que a menudo contienen una gran cantidad de características o atributos relacionados con los elementos a recomendar y los usuarios que los consumen.

Es aquí donde los métodos de reducción de dimensión entran en juego como una herramienta esencial. Estos métodos se utilizan para resumir los datos, manteniendo la esencia de los datos originales, pero reduciendo la complejidad al eliminar características redundantes o ruidosas mediante la obtención de factores latentes.

1.3.1. Métodos de Factorización Matricial

Los modelos de factores latentes basados en Métodos de Factorización Matricial (MFM) caracterizan ítems y usuarios a través de factores de menor dimensión. Estos métodos ofrecen precisión predictiva y escalabilidad, adaptándose a diversas situaciones. La fortaleza de los MFM radica en su capacidad para integrar información adicional, permitiendo inferir preferencias de usuarios a partir de comentarios implícitos, que reflejan su comportamiento, como historial de compras o navegación.

En los MFM se mapean tanto a usuarios como a ítems a un espacio común de factores latentes de dimensionalidad D, de modo que las interacciones entre usuario e ítems se modelan como productos internos en ese espacio. En consecuencia, cada ítem i se asocia con un vector $q_i \in \mathbb{R}^{D\times I}$, y cada usuario u se asocia con un vector $p_u \in \mathbb{R}^{D\times U}$ (Rennie y Srebro, 2005). El producto punto resultante, $p_u^T q_i$, sirve de medida de similitud y captura la interacción entre el usuario u y el ítem i, es decir, el interés general del usuario en las características del ítem. Este producto punto refleja

la interacción del usuario u para el ítem i, denotado por $x_{ui} \in X^{U \times I}$ donde X es una matriz sparse de ratings, lo que lleva a la estimación

$$\hat{x}_{ui} = p_u^T q_i.$$

Para aprender los vectores de factores p_u y q_i, el sistema minimiza el error cuadrático regularizado en el conjunto de calificaciones conocidas

$$\min_{p^*, q^*} \sum_{(u,i) \in \mathrm{K}} \left(x_{ui} - p_u^T q_i\right)^2 + \lambda \left(\|q_i\|^2 + \|p_u\|^2\right) \tag{1.7}$$

Aquí, K es el conjunto de pares (u, i) para los cuales x_{ui} es conocido (el conjunto de entrenamiento).

El sistema aprende el modelo ajustándose a las interacciones previamente observadas. Sin embargo, el objetivo es generalizar esas calificaciones anteriores de manera que prediga las interacciones futuras y desconocidas. Por lo tanto, el sistema debe evitar el sobreajuste de los datos observados al regularizar los parámetros aprendidos, cuyas magnitudes son penalizadas. La constante λ es un término de regularización y suele determinarse mediante validación cruzada (Koren, Bell, y Volinsky, 2009b).

Una estrategia común para minimizar la Ecuación (1.7) es emplear el algoritmo de Descenso de Gradiente Estocástico (SGD, por sus siglas en inglés). Este método itera sobre las observaciones del conjunto de entrenamiento y, para cada par usuario-ítem (u, i), estima el valor x_{ui} y calcula el error de predicción correspondiente como:

$$e_{ui} \stackrel{\text{def}}{=} x_{ui} - p_u^T q_i$$

Posteriormente, los parámetros se actualizan en la dirección opuesta al gradiente

con una magnitud proporcional a γ, obteniendo:

$$q_i \leftarrow q_i + \gamma \cdot (e_{ui} \cdot p_u - \lambda \cdot q_i)$$
$$p_u \leftarrow p_u + \gamma \cdot (e_{ui} \cdot q_i - \lambda \cdot p_u)$$

Esto permite una optimización con un tiempo de ejecución relativamente rápido.

1.3.2. Métodos Probabilísticos de Factorización Matricial

Los métodos de Factorización Matricial Probabilística (FMP) modelan la matriz de interacciones entre usuarios e ítems como el producto de dos matrices de menor rango, una para los usuarios y otra para los ítems. Los modelos probabilísticos escalan de manera lineal con el número de observaciones y ofrecen un buen desempeño en conjuntos de datos altamente dispersos. Además, permiten estimar la probabilidad de que un usuario prefiera un ítem en particular, utilizando principios de inferencia bayesiana para la estimación de parámetros, lo que facilita una mejor captura de la incertidumbre en los datos.

Mnih y Salakhutdinov (2007) proponen un modelo FMP partiendo de la siguiente definición de la distribución condicional sobre las interacciones observadas

$$P\left(\mathbb{X} \mid p_u, q_i, \sigma^2\right) = \prod_{u=1}^{U} \prod_{i=1}^{I} \left[\mathcal{N}\left(\mathbb{X}_{ui} \mid p_u^T q_i, \sigma^2\right)\right]^{I_{ui}},$$

donde $\mathcal{N}(\mathbb{X}|\mu, \sigma^2)$ es la función de densidad de probabilidad (pdf) de una distribución Gaussiana con media μ y varianza σ^2, e I_{ui} es la función indicadora de interacción entre el usuario y el ítem, done es igual a 1 si el usuario u interactuó con el ítem i e igual a 0 en otro caso (Dueck, Frey, Dueck, y Frey, 2004; Tipping y Bishop, 1999). Además, se establecen las a priori de los vectores de usuarios y de características de

ítems con Gaussianas esféricas de media cero como

$$P\left(p_u \mid \sigma_p^2\right) = \prod_{u=1}^{U} \mathcal{N}\left(p_u \mid 0, \sigma_p^2 \mathbf{I}\right),$$
$$P\left(q_i \mid \sigma_q^2\right) = \prod_{i=1}^{I} \mathcal{N}\left(q_i \mid 0, \sigma_q^2 \mathbf{I}\right),$$

donde $\mathbf{I}$ representa a una matriz identidad. Luego, la distribución log posterior sobre el usuario y las características de los ítems está dado por

$$\begin{aligned}\ln P\left(p_u, q_i \mid \mathbb{X}, \sigma^2, \sigma_q^2, \sigma_p^2\right) = &-\frac{1}{2\sigma^2}\sum_{u=1}^{U}\sum_{i=1}^{I} I_{ui}\left(\mathbb{X}_{ui} - p_u^T q_i\right)^2 - \frac{1}{2\sigma_p^2}\sum_{u=1}^{U} p_u^T p_u \\ &- \frac{1}{2\sigma_q^2}\sum_{i=1}^{I} q_i^T q_i \\ &- \frac{1}{2}\left(\left(\sum_{u=1}^{U}\sum_{i=1}^{I} I_{ui}\right)\ln\sigma^2 + U\ln\sigma_p^2 + I\ln\sigma_q^2\right) + C,\end{aligned}$$

donde C es una constante que no depende de los parámetros. La función objetivo del modelo es la suma de los errores cuadráticos con términos cuadráticos de regularización:

$$e = \frac{1}{2}\sum_{u=1}^{U}\sum_{i=1}^{I} I_{ui}\left(\mathbb{X}_{ui} - p_u^T q_i\right)^2 + \frac{\lambda_p}{2}\sum_{u=1}^{U}\|p_u\|_{Fro}^2 + \frac{\lambda_q}{2}\sum_{i=1}^{I}\|q_i\|_{Fro}^2,$$

donde $\lambda_p = \sigma^2/\sigma_p^2, \lambda_q = \sigma^2/\sigma_q^2$, y $\|\cdot\|_{Fro}^2$ denota a la norma de Frobenius. La eficiencia en el entrenamiento de modelos FMP proviene de encontrar solo estimaciones puntuales de los parámetros del modelo e hiperparámetros, en lugar de inferir la distribución posterior completa sobre ellos (Rendle, 2010). Este enfoque ha demostrado ser efectivo en sistemas de recomendación, donde la factorización de matrices se emplea para modelar las interacciones usuario-ítem y mejorar la precisión de las recomendaciones personalizadas Koren, Bell, y Volinsky (2009a).

1.3.3. Recomendaciones basadas en Autoencoders

Los Autoencoders (AE) son redes neuronales que aprenden representaciones comprimidas de datos no etiquetados a través de un proceso de codificación y decodificación, como se ilustra en la Figura 1.1.

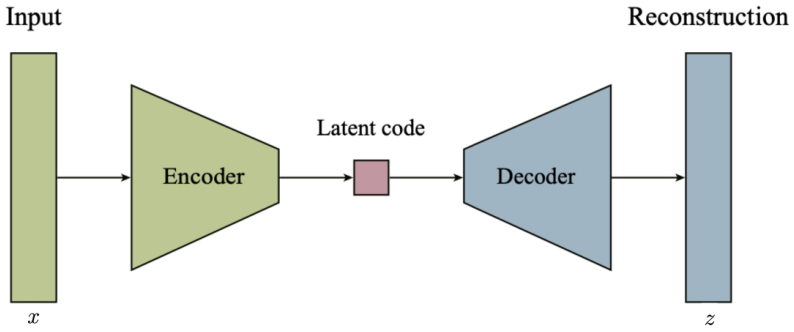


Figura 1.1: Estructura de un Autoencoder, compuesta por dos componentes: el Encoder y el Decoder. La entrada es procesada por el Encoder para producir un código latente, que el Decoder utiliza posteriormente para reconstruir la salida.

En un AE tradicional, la red se entrena para reconstruir la entrada a partir de su representación latente, que es un punto único en el espacio latente. El objetivo es minimizar el error de reconstrucción entre la entrada original y la salida reconstruida. Matemáticamente, este entrenamiento se realiza minimizando el error cuadrático medio:

$$\mathcal{L}(x, z) = \frac{1}{N} \sum_{i=1}^{N} (x_i - z_i)^2 + \Omega(\theta) \tag{1.8}$$

donde x es la entrada, z es la salida reconstruida, y $\Omega(\theta)$ es una función de regularización que penaliza la complejidad del modelo.

Los AE son particularmente útiles en sistemas de recomendación para modelar interacciones complejas entre usuarios e ítems. La representación latente captura características clave, como similitudes y relaciones, facilitando recomendaciones precisas y contextualizadas.

En esta tesis nos enfocamos en una extensión de esta familia de modelos: los Autoencoders Variacionales (VAE). En la literatura, se suele introducir VAE a partir

de un AE. Existen otros enfoques como el de Welling (2020), el cual los introduce desde una perspectiva bayesiana la cual se explicará más adelante en el Capítulo 2.

A diferencia de los AE tradicionales, que utilizan un mapeo determinista, los VAE introducen un componente estocástico en la codificación. Esto permite una exploración más amplia del espacio latente y la generación de nuevas instancias que conservan las características estadísticas de los datos originales (Kingma y Welling, 2014).

El entrenamiento de un VAE también involucra tanto un encoder como un decoder. Sin embargo, se añade una capa de complejidad ya que el Encoder no codifica una entrada en un único punto, sino como una distribución sobre el espacio latente. Este enfoque se ilustra en la Figura 1.2.

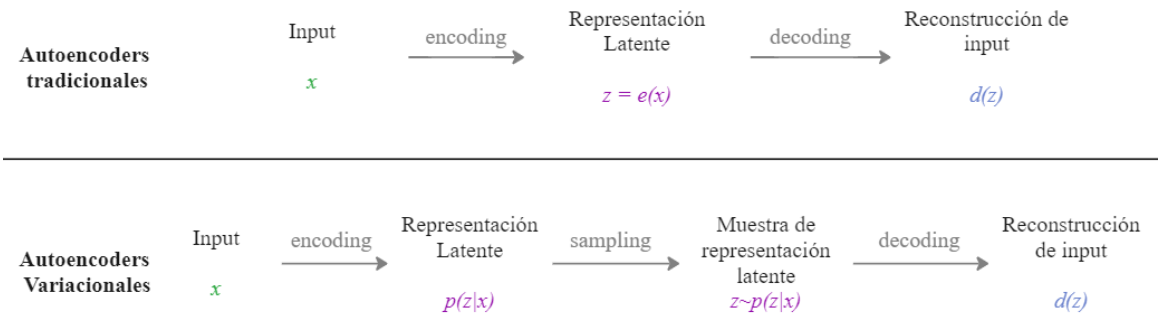


Figura 1.2: Comparación entre un Autoencoder tradicional y un Autoencoder Variacional, destacando la codificación estocástica del segundo.

Capítulo 2

Construcción de Autoencoders Variacionales Multinomiales

Como se mencionó en el capítulo 1, nuestro objetivo es explorar cómo aprovechar información adicional tanto sobre los usuarios como sobre el contenido de las películas en el contexto de modelos locales con Autoencoders Variacionales Multinomiales (Mult-VAE). Para esto, en este capítulo se presentan los fundamentos y componentes necesarios para construir un Mult-VAE aplicado a sistemas de recomendación. Se explicarán la motivación de la elección del modelo VAE, sus fundamenos y componentes principales para su construcción, así como el uso del ELBO como función objetivo, y el truco de reparametrización para permitir su entrenamiento eficiente.

2.1. Fundamentos del Autoencoder Variacional (VAE)

En aprendizaje automático, una distinción importante es aquella entre el modelado generativo y el modelado discriminativo. Los modelos discriminativos se enfocan en aprender una relación directa entre los datos de entrada x y sus etiquetas y, es decir, modelan directamente la probabilidad condicional $p(y \mid x)$. En este enfoque

nos preguntamos: dadas las características de entrada, ¿qué clase o etiqueta les corresponde? Esto es útil para tareas como la clasificación. En cambio, los modelos generativos buscan entender cómo se generan los datos. En este caso, la pregunta es: dada una clase y, ¿qué características son esperables en los datos?. Para ello, modelan la distribución conjunta $p(x, z)$, donde z representa variables latentes que capturan factores ocultos que influyen en la generación de los datos observados. A partir de esta formulación, también es posible derivar distribuciones marginales como $p(x)$, o condicionales como $p(y \mid x)$ si las etiquetas son parte del modelo. Gracias a esto, los modelos generativos no solo permiten hacer predicciones, sino también simular datos realistas y adaptarse mejor a nuevas situaciones no vistas durante el entrenamiento.

Este tipo de modelado resulta especialmente útil cuando se cuenta con pocos datos etiquetados o se desea aprender patrones generales sin supervisión directa. Precisamente, los *VAEs* permiten reconocer la estructura latente en los datos mediante la estimación de $p(z \mid x)$, es decir, capturan la incertidumbre en los datos a través de una variable latente z. Además, permiten generar nuevas muestras al modelar explícitamente la distribución $p(x \mid z)$, lo cual los hace útiles tanto para tareas discriminativas como de generación (Welling, 2020).

Para entrenarlos, en lugar de evaluar una distribución posterior $p(z \mid x)$ individualmente para cada dato, se entrena una red neuronal, llamada *encoder*, que aprende a aproximar todas estas distribuciones de manera eficiente. Esta aproximación se realiza mediante inferencia variacional. Así, el problema de inferencia se convierte en uno de optimización: en lugar de calcular la distribución posterior exacta, la cual suele ser intratable, se busca una aproximación dentro de una familia de distribuciones parametrizadas, maximizando una cota inferior del log-verosímil conocida como *Evidence Lower Bound* (ELBO) (Bishop y Bishop, 2024), la cual será explicada más adelante en este capítulo.

Dentro de la arquitectura del VAE, el complemento de la red neuronal del encoder es el *decoder*, también llamado red generativa, Mientras que el encoder aprende a

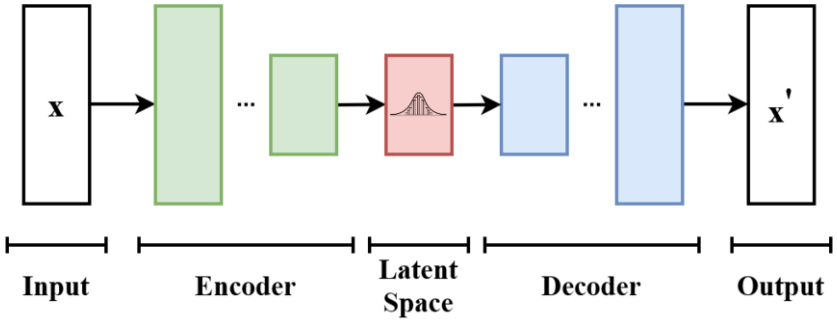


Figura 2.1: Esquema básico de un Autoencoder Variacional. Ilustración tomada de https://en.wikipedia.org/wiki/Variational_autoencoder

aproximar la distribución posterior $q(z \mid x)$, el decoder se encarga de modelar la distribución $p(x \mid z)$, es decir, cómo se generan los datos a partir de las variables latentes (Bishop y Bishop, 2024). Ambos componentes se implementan como redes neuronales con parámetros independientes, pero se entrenan de forma conjunta para maximizar el ELBO, la Figura 2.1 ilustra esta estructura básica.

A diferencia de los autoencoders clásicos, donde el objetivo es reconstruir una entrada de forma determinista, en el VAE se define explícitamente una distribución probabilística sobre el espacio latente y se genera una distribución sobre los datos en el espacio original.

2.1.1. Ejemplo Ilustrativo: Generación de Imágenes con VAE

Como ejemplo ilustrativo, generamos imágenes de dígitos escritos a mano del conjunto de datos *MNIST* (2023) mediante un VAE. Hacemos esto alimentando nuestra red neuronal con un conjunto de datos de estas imágenes para que aprenda sus características. En el modelo, el decoder toma como entrada una imagen de un dígito numérico del conjunto de datos, el cual cuenta con una alta dimensionalidad y la comprime de manera que solo contiene información útil. Luego, el decoder intenta reconstruir la entrada original a partir de esta representación comprimida, tratando

de hacerla lo más similar posible a la entrada original, ver Figura 2.2.

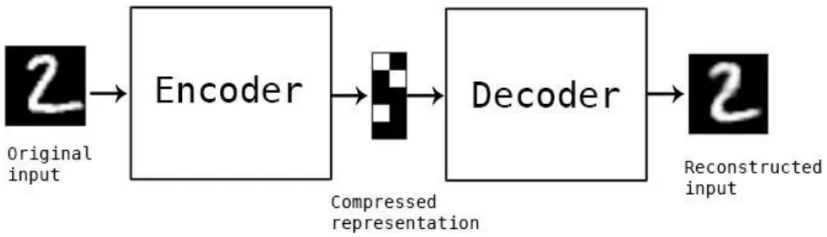


Figura 2.2: Autoencoder para MNIST. Imagen recuperada de *Auto-Encoder: What Is It? And What Is It Used For? (Part 1)* (2023)

Las imágenes que VAE intenta reconstruir y las imágenes reconstruidas por VAE se pueden ver en la Figura 2.3. Con 100 épocas de entrenamiento se obtuvo reconstrucciones de las imágenes bastante buenas.

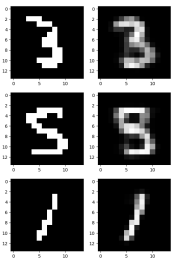


Figura 2.3: Imágenes que VAE intenta reconstruir a la izquierda y las imágenes reconstruidas por VAE a la derecha.

También ejemplificamos la generación de nuevas imágenes con VAE. En la Figura 2.4 se muestra el resultado de generar imágenes muestreando aleatoriamente desde el espacio latente y decodificando el vector muestreado.

En las siguientes secciones se discuten fundamentos que ayudan a construir un VAE: modelos probabilísticos e inferencia variacional, estimación de la función objetivo *ELBO*, su gradiente, su optimización basado en gradiente estocástico y finalmente, en la Sección (2.5) introducimos el modelo Autoencoder Variacional Multinomial (Mult-VAE) en SR.

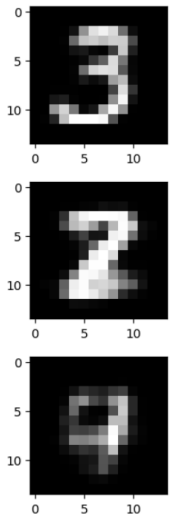


Figura 2.4: Imágenes generadas con VAE a partir de una muestra aleatoria. El modelo cuenta con capas lineales o totalmente conectados con un total de 3 capas en el encoder, y un decoder con 2 capas. Entrenamiento de 100 épocas.

2.2. Modelos Probabilísticos e Inferencia Variacional

En el campo del aprendizaje automático, a menudo estamos interesados en aprender modelos probabilísticos de diversos fenómenos a partir de datos. Kingma, Welling, y cols. (2019) describen a los modelos probabilísticos como descripciones matemáticas de tales fenómenos y en cierto sentido, las formas más completas de modelos probabilísticos especifican todas las correlaciones y dependencias de orden superior entre las variables en el modelo, en forma de una distribución de probabilidad conjunta sobre esas variables.

Sea x el vector que representa el conjunto de todas las variables observadas cuya distribución conjunta nos gustaría modelar. Suponemos que la variable observada x es una muestra aleatoria de un proceso subyacente desconocido, cuya verdadera distribución de probabilidad $p^*(x)$ es desconocida. Intentamos aproximar este proceso subyacente con un modelo elegido $p_\theta(x)$, con parámetros θ (Kingma y cols., 2019). Esto es:

$$x \sim p_\theta(x)$$

En aprendizaje automático se busca un valor de los parámetros θ tal que la función de distribución de probabilidad dada por el modelo, $p_\theta(x)$ se aproxime a la verdadera distribución de los datos, denotada por $p^*(x)$, tal que para cada x observada:

$$p_\theta(x) \approx p^*(x)$$

Para poder trabajar con modelos más complejos introducimos a la variable latente z, la cual es una representación de baja dimensión que captura las características latentes importantes de los datos de entrada x. Esta variable latente se asocia con una distribución de probabilidad, un ejemplo clásico es una mezcla de gaussianas (extension de una distribución normal), donde z marca de cual gausiana de la mezcla proviene x.

Sea $p_\theta(x)$ la verosimilitud marginal entre los datos x y la variable latente z

$$p(x) = \int p_\theta(x \mid z)\, p(z)\, dz. \tag{2.1}$$

Esta probabilidad marginal no cuenta con una solución analítica o un estimador eficiente: la integral que define $p(x)$ es, en general, intratable para modelos generativos complejos, y evaluarla requiere métodos de aproximación como Monte Carlo, los cuales son costosos o ineficientes en contextos de entrenamiento con grandes cantidades de datos. La intratabilidad de $p_\theta(x)$ está relacionada con la intratabilidad de la distribución posterior $p_\theta(z \mid x)$.

La inferencia variacional ayuda con la dificultad de la intratabilidad, ya que aproxima a la posterior intratable $p(z|x)$ mediante una distribución variacional más simple $q(z)$. En (Liang y cols., 2018) consideran a $q(z)$ como una Gaussiana con matriz de

covarianza diagonal definida de la siguiente manera:

$$q(z) = \mathcal{N}(\mu, \text{diag}(\sigma^2)).$$

En Kingma y Welling (2014) se propone reemplazar los parámetros individuales (μ, σ^2) para cada punto de datos por una función g_ϕ, parametrizada por una red neuronal, que depende de la observación x:

$$g_\phi(x) \equiv [\mu_\phi(x), \sigma_\phi(x)] \in \mathbb{R}^{2K},$$

donde $\mu_\phi(x)$ y $\sigma_\phi(x)$ son vectores de dimensión K, y determinan los parámetros de una distribución Gaussiana multivariada diagonal:

$$q_\phi(z \mid x) = \mathcal{N}\left(\mu_\phi(x), \text{diag}(\sigma_\phi^2(x))\right). \tag{2.2}$$

Esto significa que, dadas las observaciones x, el encoder produce como salida los parámetros de la distribución variacional $q_\phi(z \mid x)$, que es optimizada para aproximar la posterior intratable $p(z \mid x)$ (Rezende, Mohamed, y Wierstra, 2014b). Al combinar esta distribución con el modelo generativo $p_\theta(x \mid z)$, se construye una arquitectura neuronal que se asemeja a un autoencoder probabilístico, base fundamental del VAE.

2.2.1. Divergencia de Kullback-Leibler

La divergencia de Kullback-Leibler (KL) es una medida asimétrica que cuantifica la diferencia entre dos distribuciones de probabilidad $p_1(x)$ y $p_2(x)$. Se define como:

$$\mathrm{KL}(p_1 \,\|\, p_2) = \mathbb{E}_{x \sim p_1(x)} \left[\log \left(\frac{p_1(x)}{p_2(x)} \right) \right], \tag{2.3}$$

donde x es una variable aleatoria que sigue la distribución p_1. Una propiedad fundamental de esta divergencia es que siempre es no negativa:

$$\mathrm{KL}(p_1 \,\|\, p_2) \geq 0, \tag{2.4}$$

y es igual a cero si, y solo si, $p_1(x) = p_2(x)$ para todo x (Popkes, 2019).

En el contexto de los VAE, el objetivo es estimar la distribución posterior sobre las variables latentes z dada una observación x. Por la regla de Bayes, esta distribución se define como:

$$p_\theta(z \mid x) = \frac{p_\theta(x \mid z)\, p(z)}{p(x)}, \tag{2.5}$$

donde $p_\theta(x \mid z)$ es el modelo generativo (decoder), $p(z)$ es la distribución previa sobre los latentes (usualmente una Gaussiana estándar), y $p(x)$ es la marginal de los datos, la cual sabemos que, en general, es intratable para modelos generativos complejos. Por ello, en lugar de calcular directamente la distribución posterior $p_\theta(z \mid x)$, se recurre a una aproximación mediante una distribución variacional $q_\phi(z \mid x)$, parametrizada por el encoder.

El objetivo de la inferencia variacional es entonces encontrar los parámetros de q_ϕ que permitan aproximar lo mejor posible a la posterior verdadera. Esta aproximación se formaliza minimizando la divergencia de Kullback-Leibler entre ambas distribuciones:

$$\mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)). \tag{2.6}$$

Así, la divergencia KL se convierte en un criterio central para guiar la optimización del modelo. Esta idea conduce a la formulación del *Evidence Lower Bound* (ELBO), que será desarrollada en la siguiente sección.

2.3. Estimación usando ELBO

Para estimar la verosimilitud marginal de los datos $p_\theta(x)$, podríamos seguir el camino tradicional que ofrece la regla de Bayes. En este marco, la distribución marginal se expresa como:

$$p_\theta(x) = \frac{p_\theta(x \mid z)\, p(z)}{p_\theta(z \mid x)}. \tag{2.7}$$

Sin embargo, como se discutió en la sección anterior, calcular directamente la distribución posterior $p_\theta(z \mid x)$ resulta intratable para modelos generativos complejos, ya que requiere evaluar la integral que define $p(x)$ en el denominador. Esto limita el uso directo de esta expresión para propósitos de entrenamiento.

Una alternativa a este problema es la propuesta por Kingma y cols. (2019), quienes sugieren reemplazar el cálculo de $p_\theta(x)$ por una cota inferior optimizable, conocida como *Evidence Lower Bound* (ELBO). Esta cota permite aproximar la log-verosimilitud marginal de los datos mediante inferencia variacional, al sustituir la posterior exacta $p_\theta(z \mid x)$, que es intratable, por una distribución aproximada $q_\phi(z \mid x)$, parametrizada por el encoder del modelo. La diferencia entre ambas distribuciones se cuantifica mediante la medida de KL:

$$\mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)), \tag{2.8}$$

la cual se minimiza para que $q_\phi(z \mid x)$ se aproxime lo más posible a la posterior

verdadera.

Para la derivación del ELBO detallaremos dos caminos o alternativas. La primer derivación alternativa del ELBO parte directamente de la expresión de la divergencia KL entre $q_\phi(z \mid x)$ y $p_\theta(z \mid x)$, para cualquier elección de q_θ:

$$\begin{aligned}\mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)) &= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z \mid x) - \log p_\theta(z \mid x)\right].\end{aligned}$$

Sustituyendo la expresión de Bayes (2.7) para $p_\theta(z \mid x)$, se obtiene:

$$\begin{aligned}\mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)) &= \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z \mid x) - \log\left(\frac{p_\theta(x \mid z)\, p(z)}{p(x)}\right)\right] \\ &= \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z \mid x) - \log p_\theta(x \mid z) - \log p(z) + \log p_\theta(x)\right] \\ &= \log p_\theta(x) - \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log q_\phi(z \mid x) - \log p(z)\right] \\ &= \log p_\theta(x) - \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] + \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z))\end{aligned}$$

Reordenando los términos, obtenemos la siguiente expresión para la log-verosimilitud de los datos $\log p_\theta(x)$:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z)) + \mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)).$$

Finalmente, al reorganizar los términos y utilizar que la divergencia KL es no negativa, se obtiene la expresión final del ELBO $\mathcal{L}$ como cota inferior de $\log p_\theta(x)$:

$$\mathcal{L}_{\theta,\phi}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x \mid z)\right] - \mathrm{KL}(q_\phi(z \mid x) \,\|\, p(z)). \tag{2.9}$$

Otra formulación de esta aproximación surge al reescribir la log-verosimilitud a partir de una distribución variacional arbitraria $q_\phi(z \mid x)$. A partir de esta elección, podemos escribir la log-verosimilitud marginal como una esperanza bajo $q_\phi(z \mid x)$, ya que el valor es constante con respecto a la variable de integración:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x) \right]. \tag{2.10}$$

Este paso es simplemente una reescritura que aprovecha que $\log p_\theta(x)$ no depende de z, por lo tanto su valor esperado bajo cualquier distribución válida sigue siendo el mismo.

A continuación, se utiliza la relación entre la distribución conjunta y la condicional, reescribiendo $\log p_\theta(x)$ como:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{p_\theta(z \mid x)} \right) \right]. \tag{2.11}$$

En el siguiente paso, se introduce y multiplica por la unidad en forma de $\frac{q_\phi(z|x)}{q_\phi(z|x)}$. Esto permite factorizar la expresión en dos partes:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \cdot \frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right) \right]. \tag{2.12}$$

Aplicando propiedades del logaritmo, la expresión se descompone como la suma de dos términos:

$$\log p_\theta(x) = \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right) \right] + \mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{q_\phi(z \mid x)}{p_\theta(z \mid x)} \right) \right]. \tag{2.13}$$

El primer término de la derecha,

$$\mathbb{E}_{q_\phi(z|x)} \left[\log \left(\frac{p_\theta(x, z)}{q_\phi(z \mid x)} \right) \right],$$

es lo que se conoce como ELBO, y constituye el término que se optimiza durante el entrenamiento del VAE. Este término puede interpretarse como una forma de balancear dos objetivos: la reconstrucción de los datos a partir del espacio latente (a través de $p_\theta(x \mid z)$) y el ajuste de la distribución latente $q_\phi(z \mid x)$ a una distribución previa $p(z)$.

El segundo término,

$$\mathbb{E}_{q_\phi(z|x)}\left[\log\left(\frac{q_\phi(z \mid x)}{p_\theta(z \mid x)}\right)\right],$$

es la divergencia KL entre la distribución variacional q_ϕ y la posterior verdadera p_θ. Este término siempre es no negativo, y se anula si y solo si $q_\phi(z \mid x) = p_\theta(z \mid x)$ para todo x. Es decir, cuando la aproximación es exacta.

La ecuación (2.13) establece que la log-verosimilitud de los datos puede descomponerse como la suma del ELBO y esta divergencia KL:

$$\log p_\theta(x) = \mathcal{L}_{\theta,\phi}(x) + \mathrm{KL}(q_\phi(z \mid x) \,\|\, p_\theta(z \mid x)). \tag{2.14}$$

Dado que el segundo término es no negativo, se sigue directamente que

$$\mathcal{L}_{\theta,\phi}(x) \leq \log p_\theta(x).$$

Es decir, el ELBO actúa como una cota inferior a la log-verosimilitud marginal (Bishop y Bishop, 2024). Cuanto más cercana sea la distribución variacional $q_\phi(z \mid x)$ a la posterior verdadera $p_\theta(z \mid x)$, más ajustada será la cota. En el límite en que ambas distribuciones coinciden, la divergencia KL es cero y la ELBO se vuelve exactamente igual a la log-verosimilitud.

Esta descomposición no solo justifica el uso del ELBO como función objetivo optimizable, sino que también permite interpretar la divergencia KL como la brecha entre la cota y el valor real que se busca estimar. Esta interpretación motiva el uso del *truco de reparametrización* para su optimización, la cual veremos en las siguientes secciones.

2.3.1. Optimización del ELBO basado Gradiente Estocástico

Una ventaja clave del ELBO es que, a diferencia de la verosimilitud marginal, puede optimizarse de manera directa respecto a los parámetros del modelo generativo θ y del modelo de inferencia ϕ utilizando técnicas basadas en descenso de gradiente. En particular, cuando se dispone de un conjunto de datos $\mathcal{D} = \{x^{(i)}\}_{i=1}^{N}$ con muestras independientes e idénticamente distribuidas (i.i.d.), la función objetivo a optimizar corresponde a la suma de los ELBOs individuales:

$$\mathcal{L}_{\theta,\phi}(\mathcal{D}) = \sum_{x \in \mathcal{D}} \mathcal{L}_{\theta,\phi}(x). \tag{2.15}$$

En la práctica, esta suma se aproxima mediante muestras por lotes (minibatches) y se actualiza usando gradiente estocástico (SGD) (Bottou, Curtis, y Nocedal, 2018). No obstante, calcular el gradiente exacto de $\mathcal{L}_{\theta,\phi}(x)$ respecto a θ y ϕ resulta difícil en general, debido a que la esperanza involucrada en el ELBO se toma con respecto a la distribución $q_\phi(z \mid x)$, la cual depende de los mismos parámetros que se desean optimizar.

Welling (2020) menciona que para los parámetros del modelo generativo θ, dado que el modelo $p_\theta(x, z)$ es una función explícita y diferenciable respecto a θ, se puede intercambiar el operador de gradiente con la esperanza:

$$\begin{aligned}
\nabla_\theta \mathcal{L}_{\theta,\phi}(x) &= \nabla_\theta \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x, z) - \log q_\phi(z \mid x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\nabla_\theta \log p_\theta(x, z)\right] \\
&\approx \nabla_\theta \log p_\theta(x, z),
\end{aligned}$$

donde la última línea corresponde a un estimador de Monte Carlo con una sola muestra de $z \sim q_\phi(z \mid x)$. El término $\log q_\phi(z \mid x)$ no depende de θ, por lo que su gradiente se anula.

En contraste, el cálculo del gradiente respecto a los parámetros ϕ del modelo de inferencia es más delicado. Debido a que ϕ aparece tanto en el integrando como en la medida de integración $q_\phi(z \mid x)$, no es posible intercambiar directamente el operador de gradiente con la esperanza. En general, se cumple que:

$$\nabla_\phi \mathcal{L}_{\theta,\phi}(x) = \nabla_\phi \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) - \log q_\phi(z \mid x)\right] \tag{2.16}$$

$$\neq \mathbb{E}_{q_\phi(z|x)} \left[\nabla_\phi \left(\log p_\theta(x,z) - \log q_\phi(z \mid x)\right)\right], \tag{2.17}$$

como se discute en (Mnih y Gregor, 2014). Este problema impide aplicar directamente el gradiente estocástico tradicional. No obstante, en el caso de variables latentes continuas, este obstáculo puede resolverse mediante el *truco de reparametrización*, mencionado anteriormente y detallado a continuación.

2.4. Truco de Reparametrización

El *truco de reparametrización* es una estrategia que permite optimizar el ELBO en modelos con variables latentes continuas, cuando la distribución variacional $q_\phi(z \mid x)$ depende de los parámetros que deseamos ajustar. Como se explica en Bishop y Bishop (2024), este truco transforma una esperanza tomada con respecto a una distribución dependiente de parámetros en una esperanza sobre una distribución fija, trasladando la dependencia paramétrica a una transformación diferenciable. De este modo, se vuelve posible aplicar métodos basados en retropropagación de gradientes de manera eficiente.

El problema central aparece al querer derivar el ELBO respecto a los parámetros del encoder ϕ, ya que la distribución $q_\phi(z \mid x)$ está definida sobre el mismo espacio sobre el que se realiza la integración. Para resolver esta dificultad, en lugar de calcular

el gradiente de una esperanza sobre una distribución dependiente de ϕ, se expresa la variable aleatoria como una transformación de una variable auxiliar ϵ, que sigue una distribución conocida y fija.

Así, el cálculo del gradiente se transforma en un problema de cambiar la densidad de una variable aleatoria a través de una función determinista (Kingma y Welling, 2014). Esta idea se apoya en el principio del cambio de variables en probabilidad, permitiendo considerar los datos como el resultado de aplicar una función sobre una variable latente con distribución conocida.

Para ilustrar esta idea, consideremos el caso más simple: una variable aleatoria

$$x \sim \mathcal{N}(\mu, \sigma^2).$$

En lugar de trabajar directamente con esta distribución al momento de muestrear, podemos expresar x como una transformación de una variable auxiliar

$$\epsilon \sim \mathcal{N}(0, 1),$$

mediante la siguiente relación:

$$x = \mu + \sigma\epsilon. \tag{2.18}$$

Aquí, ϵ contiene toda la aleatoriedad, mientras que μ y σ están explícitamente presentes en la transformación. Si quisiéramos optimizar alguna función objetivo que depende de muestras de x, esta formulación nos permitiría calcular derivadas respecto a μ o σ, ya que la operación de muestreo ha sido sustituida por una operación determinista diferenciable.

Esta idea se extiende naturalmente al caso multivariado, tal como se utiliza en los VAEs.

2.4.1. Aplicación a VAEs

En los VAEs, un caso común es asumir que la distribución variacional $q_\phi(z \mid x)$ es Gaussiana multivariada con covarianza diagonal, cuya media y desviación estándar son producidas por el encoder:

$$q_\phi(z \mid x) = \mathcal{N}(z \mid \mu_\phi(x), \mathrm{diag}(\sigma^2_\phi(x))). \tag{2.19}$$

En este caso, el muestreo de z se reparametriza de manera análoga al ejemplo unidimensional, como:

$$\begin{aligned} z =& \mu_\phi(x) + \sigma_\phi(x) \cdot \epsilon, \\ \epsilon \sim& \mathcal{N}(0, I). \end{aligned}$$

Aquí, $\mu_\phi(x)$ y $\sigma_\phi(x)$ son salidas de redes neuronales que dependen de los datos x, mientras que ϵ sigue una distribución fija e independiente de ϕ. La función $g(\epsilon, \phi, x)$ definida por la ecuación anterior es diferenciable respecto a ϕ, por lo que puede utilizarse dentro de un procedimiento de optimización por retropropagación.

Como señalan Bishop y Bishop (2024) y Mnih y Gregor (2014), esta reparametrización permite convertir el problema de derivar una esperanza sobre una distribución dependiente de parámetros en un problema equivalente, pero donde los parámetros aparecen de forma explícita en el integrando. Esta transformación hace que los gradientes puedan estimarse de manera eficiente mediante técnicas estándar de aprendizaje profundo.

La Figura 2.5 ilustra este procedimiento. En el panel izquierdo se observa que, en el caso original, no es posible retropropagar a través de la muestra z, debido a su dependencia estocástica de ϕ. En contraste, el panel derecho muestra cómo, al expresar z como una función determinista de ϵ, se habilita la derivación completa mediante el uso del truco de reparametrización.

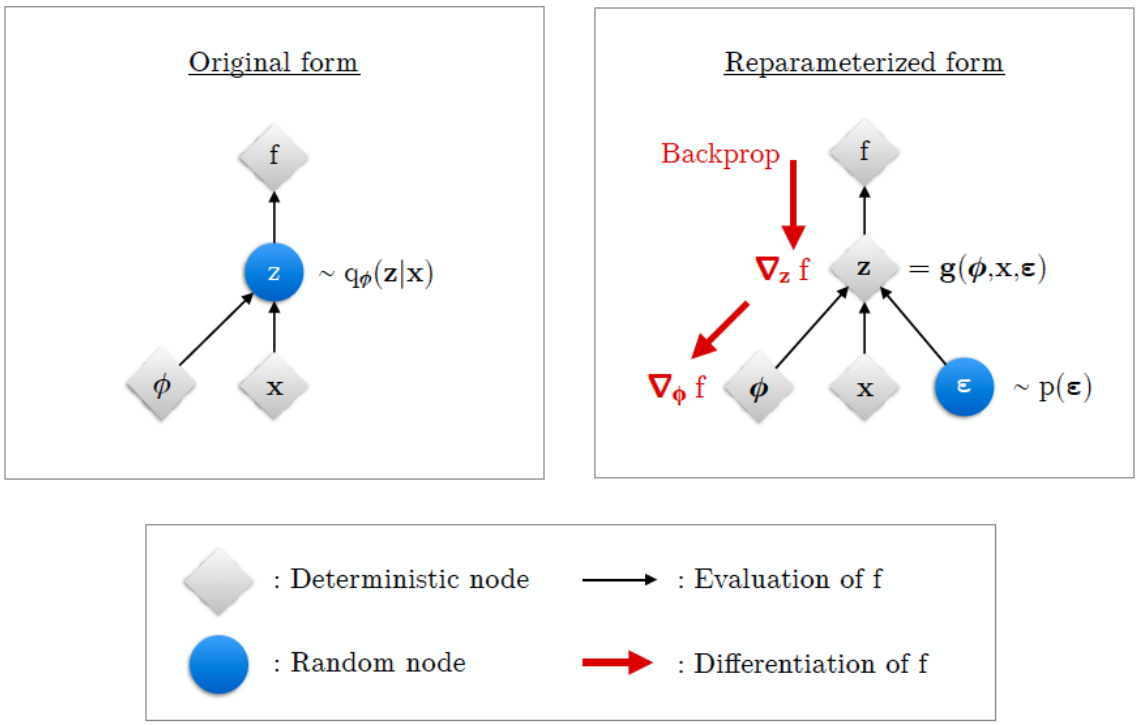


Figura 2.5: Ilustración del truco de reparametrización. La variable latente $z \sim q_\phi(z \mid x)$ se representa como una transformación determinista de una variable auxiliar $\epsilon \sim \mathcal{N}(0, I)$, permitiendo retropropagar los gradientes respecto a ϕ. Figura adaptada de Kingma y Welling (2014).

2.4.2. Gradiente del ELBO

Gracias al truco de reparametrización, descrito en la sección anterior, es posible derivar eficientemente el ELBO respecto a los parámetros del modelo. Al reescribir la variable latente $z \sim q_\phi(z \mid x)$ como una transformación determinista de una variable auxiliar $\epsilon \sim p(\epsilon)$, podemos intercambiar los operadores de gradiente y esperanza. Esto permite obtener un estimador de Monte Carlo del ELBO diferenciable con respecto a los parámetros θ y ϕ (Kingma y Welling, 2014; Rezende, Mohamed, y Wierstra, 2014a).

A partir de la ecuación (2.13), el ELBO puede reescribirse como una esperanza sobre $p(\epsilon)$:

$$\begin{aligned}\mathcal{L}_{\theta,\phi}(x) &= \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z \mid x)\right] \\ &= \mathbb{E}_{p(\epsilon)}\left[\log p_\theta(x,z) - \log q_\phi(z \mid x)\right],\end{aligned}$$

donde $z = g(\epsilon, \phi, x)$. Esta formulación permite estimar el ELBO usando una única muestra de $\epsilon \sim \mathcal{N}(0, I)$, generando una estimación de Monte Carlo del tipo:

$$\begin{aligned}\epsilon &\sim p(\epsilon), \\ z &= g(\epsilon, \phi, x), \\ \tilde{\mathcal{L}}_{\theta,\phi}(x) &= \log p_\theta(x,z) - \log q_\phi(z \mid x).\end{aligned}$$

Esta expresión puede implementarse en librerías como PyTorch o TensorFlow utilizando grafos computacionales diferenciables, lo cual permite aplicar retropropagación y técnicas estándar de descenso de gradiente estocástico (Kingma y Welling, 2014). El estimador $\nabla\widetilde{\mathcal{L}}(\theta, \phi; x)$ se calcula de forma eficiente y se utiliza para actualizar los parámetros del modelo por medio de SGD (Bottou y cols., 2018).

Este enfoque se resume en el Algoritmo 1, el cual describe el procedimiento de optimización estocástica del ELBO utilizando el truco de reparametrización. El algoritmo fue originalmente propuesto por Kingma y Welling (2014) bajo el nombre de *Auto-Encoding Variational Bayes* (AEVB). El estimador reparametrizado del ELBO también es conocido como el *Stochastic Gradient Variational Bayes* (SGVB) estimator.

Algorithm 1 Optimización estocástica del ELBO. Algoritmo extraído y adaptado de Kingma y Welling (2014).

Entrada: conjunto de datos D; modelo de inferencia $q(z|x)$; modelo generativo $p(x, z)$.
Inicialización: parámetros θ, ϕ.
Mientras no haya convergencia:

- Muestra un minibatch $M \subset D$
- Para cada $x \in M$, muestrea $\epsilon \sim \mathcal{N}(0, I)$
- Calcula $z = g(\epsilon, \phi, x)$
- Evalúa $\widetilde{\mathcal{L}}(\theta, \phi; M, \epsilon)$ y sus gradientes
- Actualiza θ y ϕ con SGD

2.5. Autoencoders Variacionales Multinomiales (Mult-VAE) para Sistemas de Recomendación

Liang y cols. (2018) proponen el modelo *Mult-VAE*, un modelo probabilístico con variables latentes diseñado específicamente para sistemas de recomendación basados en retroalimentación implícita. Su enfoque se basa en una arquitectura VAE donde las observaciones corresponden a historiales binarizados de interacción usuario-ítem, y se asume una verosimilitud multinomial para capturar la naturaleza discreta y de conteo de estos datos.

En Mult-VAE, cada historial de interacciones del usuario $x_u \in \mathbb{N}^I$ se modela como una muestra de una distribución multinomial parametrizada por una distribución latente sobre ítems:

$$x_u \sim \text{Multinomial}(N_u, \pi(z_u)), \tag{2.20}$$

donde $N_u = \sum_i x_{ui}$ es el número total de interacciones del usuario u, y $\pi(z_u) \in$

$\mathbb{R}^I$ representa una distribución de probabilidad sobre los I ítems, condicionada a la variable latente z_u. Esta variable latente $z_u \in \mathbb{R}^K$ se extrae de una distribución a priori Gaussiana estándar:

$$z_u \sim \mathcal{N}(0, I),$$
$$\pi(z_u) \propto \exp\{f_\theta(z_u)\},$$

donde f_θ es una red neuronal multicapa (decoder) parametrizada por θ que transforma la representación latente en una distribución sobre ítems mediante softmax. El modelo genera así una distribución sobre los ítems que busca asignar alta probabilidad a los ítems con los que el usuario ha interactuado.

La log-verosimilitud de los datos x_u, condicionada a la variable latente z_u, está dada por:

$$\log p_\theta(x_u \mid z_u) \propto \sum_i x_{ui} \log \pi_i(z_u). \tag{2.21}$$

Esto implica que el modelo solo se enfoca en maximizar la probabilidad de los ítems observados (interacciones positivas), lo cual es apropiado para entornos de recomendación donde no se observa retroalimentación negativa explícita.

Para estimar los parámetros θ, el modelo emplea inferencia variacional mediante un encoder que predice $\mu_\phi(x_u)$ y $\sigma_\phi(x_u)$, los cuales determinan la distribución aproximada $q_\phi(z_u \mid x_u)$. Dado que la posterior verdadera $p(z_u \mid x_u)$ es intratable, se recurre a inferencia variacional, evitando así métodos de muestreo más costosos como MCMC.

Este enfoque permite capturar representaciones compactas y personalizadas para cada usuario, aprendidas a partir de datos implícitos binarizados, y ha demostrado ser competitivo frente a modelos basados en factorización matricial, especialmente en escenarios con datos escasos o con ruido (Liang y cols., 2018).

Capítulo 3

Local Collaborative Autoencoders (LOCA)

En la sección 2.5 se introdujo el uso de Autoencoders Variacionales Multinomiales (Mult-VAE) como modelo generativo para sistemas de recomendación. Sin embargo, un único modelo global puede no ser suficiente para capturar la heterogeneidad de preferencias en conjuntos de datos amplios. Este capítulo presenta el enfoque de modelos locales propuesto por Choi y cols. (2021), el cual consiste en combinar múltiples Mult-VAE especializados en distintas regiones del espacio latente para mejorar la personalización de las recomendaciones .

3.1. Motivación

Los modelos globales son enfoques que buscan capturar patrones generales en un conjunto de datos mediante un único modelo que abarca todo el espacio de los datos. Sin embargo, esta aproximación puede pasar por alto variaciones específicas y relaciones locales dentro de los datos. En contraste, los modelos locales se centran en identificar y modelar patrones y variaciones a nivel de subcomunidades.

En el contexto de SR, la diversidad de usuarios e ítems hace relevante capturar estas relaciones locales, ya que reflejan interacciones más personalizadas. Como resultado, se pueden generar recomendaciones más precisas y adaptadas a los intereses individuales de los usuarios (López, 2013). Una estrategia común para implementar modelos locales es a través de técnicas de vecindario, como el filtrado colaborativo basado en vecindarios, el cual identifica usuarios o ítems similares y explora las subcomunidades de interés.

Un enfoque tradicional para mejorar la precisión de un modelo de SR es incrementar su complejidad, como ocurre con las redes neuronales profundas añadiendo más capas. Sin embargo, esta estrategia requiere grandes volúmenes de datos de entrenamiento, lo que no siempre es posible. Para abordar esta limitación, Choi y cols. (2021) proponen una alternativa: en lugar de aumentar la complejidad de un solo modelo global, se emplean múltiples modelos más simples que, al combinarse, capturan mejor las características locales.

Esta propuesta revisa la suposición de que la matriz de interacciones $\mathbb{X}$ es de bajo rango de forma global. Trabajos previos como Lee, Kim, Lebanon, y Singer (2013) y Lee, Kim, Lebanon, Singer, y Bengio (2016) sugirieron en cambio una suposición de bajo rango local. La idea de modelos locales bajo la suposición de bajo rango local se basa en dividir la matriz de interacciones en submatrices más pequeñas, donde cada submatriz representa una subcomunidad de usuarios e ítems con relaciones específicas y coherentes. En lugar de asumir un bajo rango global, se supone que cada una de estas submatrices tiene un bajo rango local, reflejando patrones de comportamiento e intereses más específicos.

En la Sección 1.3.1, se discutieron métodos de factorización en los que la matriz de interacciones usuario-ítem es aproximada a través de la descomposición en factores latentes. Estos son métodos globales que permiten reducir la complejidad de la matriz original a un espacio latente de menor dimensión, logrando así matrices de menor rango.

En contraste con esta visión global, el enfoque de LLORMA (Local Low-Rank Matrix Approximation) de Lee y cols. (2013, 2016) introduce la suposición de bajo rango local. LLORMA utiliza múltiples modelos locales que se solapan y representan diferentes aspectos locales de las preferencias de los usuarios. Cada submatriz se factoriza de forma independiente y luego se combinan las predicciones de todos los modelos locales para aproximar la matriz global. La combinación de estos modelos locales permite capturar patrones de subcomunidades que los modelos globales no pueden captar, generando así recomendaciones más precisas para grupos de usuarios con intereses particulares.

Otros enfoques son los relacionados con GLSLIM (*Global and Local Sparse Linear Method*) y sGLSVD (*staged Global and Local Singular Value Decomposition*), los cuales proponen mecanismos específicos para integrar tanto factores globales como locales. GLSLIM (Christakopoulou y Karypis, 2016) combina un modelo global con varios modelos locales construidos a partir de la similitud ítem-ítem. En este caso, las submatrices utilizadas por los modelos locales son disjuntas, es decir, no se solapan, lo que permite capturar relaciones específicas entre ítems que podrían pasar desapercibidas en un modelo únicamente global.

De manera similar, sGLSVD sigue el mismo enfoque, utilizando SVD como su modelo base (Christakopoulou y Karypis, 2018). La idea es mantener una estructura que pueda aprender tanto factores globales que afectan a todos los usuarios e ítems como factores locales que capturan la estructura más específica de subcomunidades. A diferencia de los modelos locales previos, sGLSVD y GLSLIM buscan mantener modelos locales más independientes, lo que facilita la identificación de patrones específicos de cada subcomunidad sin depender de una superposición de modelos.

En los trabajos de Christakopoulou y Karypis (2016, 2018); Lee, Bengio, Kim, Lebanon, y Singer (2014); Lee y cols. (2013); K. Wang, Peng, Jin, Sha, y Wang (2016) se señala que, los modelos locales existentes no exploran completamente el potencial de la suposición de bajo rango local. Por ejemplo, en algunos estudios, el

tamaño de los modelos locales es cercano al del modelo global, dejando sin descubrir la diversidad de patrones locales. Por otro lado, Lee y cols. (2014) y Lee y cols. (2013) trabajan con subcomunidades pequeñas, pero no logran superar el desempeño de un modelo global debido a la falta de datos de entrenamiento.

Dados estos enfoques, se destaca la importancia de equilibrar la capacidad de capturar tanto factores globales como locales, una idea que Choi y cols. (2021) toman como base para su modelo LOCA (Local Collaborative Autoencoders). LOCA utiliza la aproximación de LLORMA para generar modelos locales que se solapan, y al igual que en GLSLIM y sGLSVD, estiman un modelo global junto con múltiples modelos locales basados en ítems. Estos enfoques buscan explotar tanto factores globales como factores locales para una mejor cobertura de las preferencias de todos los usuarios.

La arquitectura de LOCA utiliza modelos basados en VAEs como modelo base, ya que las capas de activación no lineales de los VAEs permiten representar patrones no lineales significativos en el modelo local. Esto lo hace especialmente efectivo para capturar variaciones específicas en las subcomunidades y mejorar la precisión de las recomendaciones. Además, LOCA se enfoca en usuarios, aunque la misma lógica puede aplicarse a ítems, lo que lo convierte en un modelo versátil y adaptable a diferentes contextos de sistemas de recomendación (Choi y cols., 2021).

El modelo LOCA se basa en el algoritmo *divide y vencerás* compuesto por los siguientes tres pasos, ver Figura 3.1:

I. Descubrir conjuntos de subcomunidades locales (divide).

II. Entrenar un modelo local por cada subcomunidad (vencerás).

III. Realizar inferencia de las preferencias de cada usuario al combinar el modelo global con los múltiples modelos locales (agregación).

A continuación, se describen las estrategias para identificar comunidades locales de usuarios con preferencias similares mediante la selección de representantes, denomi-

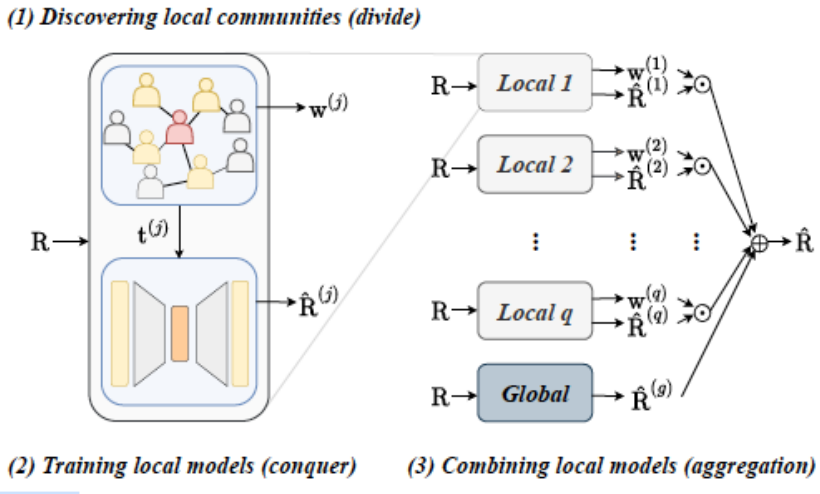


Figura 3.1: Arquitectura del modelo LOCA. $\mathcal{T}$ y $\mathcal{W}$ se utilizan para entrenar e inferir modelos locales, ambos pesos se calculan en base a la similitud de cada usuario con un usuario ancla.

nados *usuarios ancla*, y cómo definir sus usuarios vecinos, quienes serán considerados para el entrenamiento de cada modelo local. Además, se detallan los criterios para seleccionar estos anclas y el proceso de entrenamiento de los modelos locales para cada subcomunidad, así como su combinación final.

3.2. Descubriendo subcomunidades locales

El primer paso consiste en identificar comunidades locales de usuarios con intereses similares. Dado un conjunto de usuarios $\mathcal{U}$, se seleccionan q usuarios ancla que actúan como representantes de cada modelo local. Este conjunto de anclas se denota como $\mathcal{A} = a^{(1)}, a^{(2)}, ..., a^{(q)}$. Los usuarios que pertenecen a cada modelo local son aquellos más similares (cercanos en términos de una distancia elegida) a su respectivo usuario ancla, conformando así subcomunidades con características similares.

3.2.1. Anclas y Distancias

Para cada ancla, se busca un conjunto de usuarios vecinos estimando la distancia entre el ancla y el resto de los usuarios, seleccionando a los más cercanos. La distancia considerada es la arc coseno, escalada en el intervalo $[0, 1]$:

$$s = \text{dist}\left(a^{(j)}, u\right) = \arccos\left(\frac{a^{(j)} \cdot u}{\|a^{(j)}\| \cdot \|u\|}\right). \tag{3.1}$$

Los vectores $a^{(j)}$ y u utilizados en la ecuación (3.1) representan embeddings latentes, es decir, representaciones vectoriales compactas de los usuarios en un espacio de menor dimensión. Estos embeddings capturan patrones latentes de comportamiento o preferencia basados en los datos de interacción, y su calidad afecta directamente la capacidad del modelo para identificar vecindades significativas. En Choi y cols. (2021), los autores emplean embeddings generados por un modelo de recomendación previamente entrenado, utilizando estas representaciones como base para calcular las similitudes entre usuarios.

En este trabajo, consideramos diferentes estrategias para obtener dichos embeddings. En una de las estrategias se utiliza las representaciones latentes aprendidas por un modelo *Mult-VAE* preentrenado, como el propuesto por Liang y cols. (2018), donde cada usuario u es representado por el vector de medias $\mu_\phi(x_u)$ producido por el encoder del VAE.

Además, se exploran técnicas clásicas de reducción de dimensionalidad para generar embeddings a partir de los datos de interacción usuario-ítem, así como de información adicional como los atributos demográficos de los usuarios o las categorías de los ítems. Entre los métodos considerados se incluyen *ISOMAP*, *t-SNE* y *UMAP*, los cuales transforman las filas de la matriz binarizada de interacción (posiblemente enriquecida con metadatos) en puntos de un espacio latente de menor dimensión.

Para definir la similitud entre usuarios, se utiliza una función kernel, $K_h(s)$, para-

metrizada por un parámetro de ancho de banda $h > 0$, el cual actua como parámetro de escala. En Choi y cols. (2021), se utiliza la función kernel de Epanechnikov, definida como:

$$K_h(s) \propto \left(1 - s^2\right) \mathbb{1}\left[s < h\right], \tag{3.2}$$

donde $\mathbb{1}$ es la función indicadora que toma el valor 1 si $s < h$, y 0 en caso contrario. Esta función kernel mide la similitud entre usuarios, seleccionando aquellos cuyas distancias (medidas por la función de arc coseno) se encuentren dentro del ancho de banda h. De esta manera, el parámetro h actúa como un filtro que controla el grado de vecindad, agrupando usuarios con características latentes similares y facilitando la construcción de subcomunidades coherentes para los modelos locales.

Para considerar un rango amplio para el entrenamiento de una submatriz y para respetar la intuición de los modelos locales, se adoptan diferentes conjuntos de vectores para entrenamiento y prueba. Dado un conjunto de usuarios ancla $\mathcal{A} = \{a^{(1)}, a^{(2)}, \ldots, a^{(q)}\}$, se construyen dos conjuntos de vectores de pesos: $\mathcal{T} = \{t^{(1)}, t^{(2)}, \ldots, t^{(q)}\}$ y $\mathcal{W} = \{w^{(1)}, w^{(2)}, \ldots, w^{(q)}\}$. Para cada usuario ancla $a^{(j)}$, los vectores $t^{(j)} = \{t_1^{(j)}, t_2^{(j)}, \ldots, t_U^{(j)}\}$ y $w^{(j)} = \{w_1^{(j)}, w_2^{(j)}, \ldots, w_U^{(j)}\}$ representan los pesos asignados a los U usuarios del conjunto total, utilizados para entrenamiento y prueba, respectivamente.

A partir de las funciones kernel $K_{h_\mathcal{T}}(s)$ y $K_{h_\mathcal{W}}(s)$, se calculan los componentes de estos vectores de pesos, evaluando cada uno en la distancia $s = \text{dist}(a^{(j)}, u)$ entre el usuario ancla y un usuario u. En particular, cada entrada del vector $t^{(j)}$ se define como:

$$t_u^{(j)} = K_{h_\mathcal{T}}\left(\text{dist}(a^{(j)}, u)\right), \qquad \text{para todo } u \in \mathcal{U}. \tag{3.3}$$

De forma análoga, los pesos de inferencia $w_u^{(j)}$ se obtienen con:

$$w_u^{(j)} = K_{h_\mathcal{W}}\left(\text{dist}(a^{(j)}, u)\right). \tag{3.4}$$

De esta manera, los vectores $t^{(j)}$ y $w^{(j)}$ representan la similitud entre el usuario ancla y el resto de los usuarios, bajo distintas escalas de vecindad determinadas por los anchos de banda $h_{\mathcal{T}}$ y $h_{\mathcal{W}}$. En particular, la forma funcional del kernel de Epanechnikov garantiza que solo los usuarios cercanos al ancla reciben un peso distinto de cero, actuando como un filtro que limita la vecindad efectiva a una región local controlada. Esta construcción permite definir vecindades distintas para entrenamiento e inferencia, favoreciendo la estabilidad del modelo y la generalización (Choi y cols., 2021).

3.2.2. Selección de anclas

La selección de las anclas es importante para identificar a los representantes de cada subcomunidad local. Dado que tanto usuarios como ítems pueden servir como anclas, en este trabajo nos enfocamos en los usuarios para representar de manera efectiva las características de cada subcomunidad. La elección del método de selección de anclas impacta directamente la velocidad de predicción, la precisión de las recomendaciones y el rendimiento global de los modelos.

En esta subsección, exploramos dos estrategias de selección de anclas: la primera se lleva a cabo en el espacio original de calificaciones, $\mathbb{X}$, y la segunda en un subespacio derivado de $\mathbb{X}$, que condensa tanto información de calificaciones como datos adicionales sobre los usuarios.

Ambas estrategias se analizan considerando tres enfoques distintos de selección de anclas: (1) selección basada en cobertura sobre la representación latente z de $\mathbb{X}$, (2) selección mediante agrupamiento, tomando como anclas a los centroides de cada grupo, y (3) selección aleatoria de entradas en la matriz de datos.

Primera Estrategia: Selección de anclas en el espacio de calificaciones

En la primera estrategia, la selección de anclas se realiza directamente en el espacio original de la matriz de calificaciones, $\mathbb{X}$. Presentamos a continuación tres métodos de selección:

1. **Seleccion por Cobertura**

   El objetivo principal en la construcción de modelos locales es garantizar que cada usuario quede cubierto por al menos uno de los modelos especializados. Para ello, se busca seleccionar como anclas a aquellos usuarios que se encuentren rodeados de muchos otros, es decir, que sean altamente representativos dentro del espacio de similitud definido. Esta elección permite que las subcomunidades resultantes sean informativas.

   Con esta motivación, Lee y cols. (2013) y Choi y cols. (2021) proponen un algoritmo *greedy* que selecciona iterativamente anclas que cubren la mayor cantidad posible de usuarios aún no representados. El procedimiento se basa en la construcción de un grafo no dirigido y no ponderado $\mathcal{G} = (\mathcal{U}, E)$, donde los nodos representan usuarios y las aristas codifican relaciones de similitud entre ellos. La existencia de una arista entre dos usuarios u_i y u_j está determinada por si su similitud, medida mediante el kernel $K_{h_\mathcal{W}}$, es positiva. Formalmente, la matriz de adyacencia del grafo se define como:

$$A_{u_i u_j} = \begin{cases} 1, & \text{si } K_{h_\mathcal{W}}(\text{dist}(u_i, u_j)) > 0, \\ 0, & \text{en otro caso.} \end{cases} \tag{3.5}$$

   La idea general del algoritmo (ver Algoritmo 2) es construir un conjunto de anclas $\mathcal{A}$ que maximice la cobertura de nodos en el grafo, es decir, que permita alcanzar una porción significativa de usuarios a través de sus vecindades locales. Se inicializan dos conjuntos vacíos: el conjunto de anclas $\mathcal{A}$ y el conjunto de usuarios cubiertos C, donde este último contiene a los usuarios que ya están

dentro de la vecindad de alguna ancla seleccionada.

En cada iteración, se selecciona como nueva ancla aquel nodo $u \in \mathcal{U} \setminus \mathcal{A}$ que tenga el mayor número de conexiones hacia nodos aún no cubiertos (es decir, en $\mathcal{U} \setminus C$). Este nodo se agrega a $\mathcal{A}$ y su vecindad —definida según el soporte del kernel $K_{h_\mathcal{W}}$— se añade al conjunto de usuarios cubiertos C. El proceso se repite hasta que se han seleccionado q anclas. Si en algún momento se alcanza cobertura completa (es decir, $C = \mathcal{U}$) antes de completar las q selecciones, el conjunto C se reinicia para permitir la generación de nuevas anclas en regiones distintas del espacio latente.

Algorithm 2 Selección de Anclas por Cobertura

Inicializar conjuntos vacíos: $\mathcal{A} \leftarrow \emptyset$, $C \leftarrow \emptyset$
while $|\mathcal{A}| < q$ **do**
    if $C = \mathcal{U}$ **then**
        Reiniciar $C \leftarrow \emptyset$
    end if
    Seleccionar $u \in \mathcal{U} \setminus \mathcal{A}$ con mayor número de conexiones hacia $\mathcal{U} \setminus C$
    $\mathcal{A} \leftarrow \mathcal{A} \cup \{u\}$
    $C \leftarrow C \cup \{v \in \mathcal{U} : K_{h_\mathcal{W}}(\text{dist}(u, v)) > 0\}$
end while

Por ejemplo, consideremos una matriz de similitud binaria entre ocho usuarios, representada como:

$$
\begin{array}{c} \\ U_1 \\ U_2 \\ U_3 \\ U_4 \\ U_5 \\ U_6 \\ U_7 \\ U_8 \end{array}
\begin{array}{c}
\begin{array}{cccccccc} U_1 & U_2 & U_3 & U_4 & U_5 & U_6 & U_7 & U_8 \end{array} \\
\left(\begin{array}{cccccccc}
0 & 0 & 0 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \\
1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 \\
0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\
1 & 1 & 1 & 0 & 0 & 1 & 0 & 0 \\
1 & 0 & 0 & 1 & 0 & 0 & 0 & 0
\end{array} \right)
\end{array}
\tag{3.6}
$$

Esta matriz define un grafo no dirigido donde los nodos representan usuarios y las aristas indican relaciones de vecindad según la función kernel (ver Figura 3.2). Si decidimos usar dos usuarios ancla, estos serían u_4 y u_7, debido a que tienen el mayor número de conexiones. Este procedimiento asegura que las anclas seleccionadas sean representativas de sus respectivas subcomunidades. Esta estrategia garantiza que cada ancla cubra una parte sustancial del grafo, facilitando así la identificación de subcomunidades relevantes y mejorando la calidad de los modelos locales entrenados sobre ellas. representativas de sus respectivas subcomunidades.

2. Selección por Agrupamiento:

   En Choi y cols. (2021) se menciona que Christakopoulou y Karypis (2016, 2018); Lee y cols. (2016) emplean un enfoque basado en agrupamiento para seleccionar anclas. Este método forma grupos de usuarios en el espacio latente utilizando alguna medida de similitud o distancia, y luego elige los centroides de estos grupos como anclas representativas.

   Aunque este procedimiento puede mejorar la cobertura y coherencia de los modelos locales, su desempeño depende críticamente de decisiones de modelado

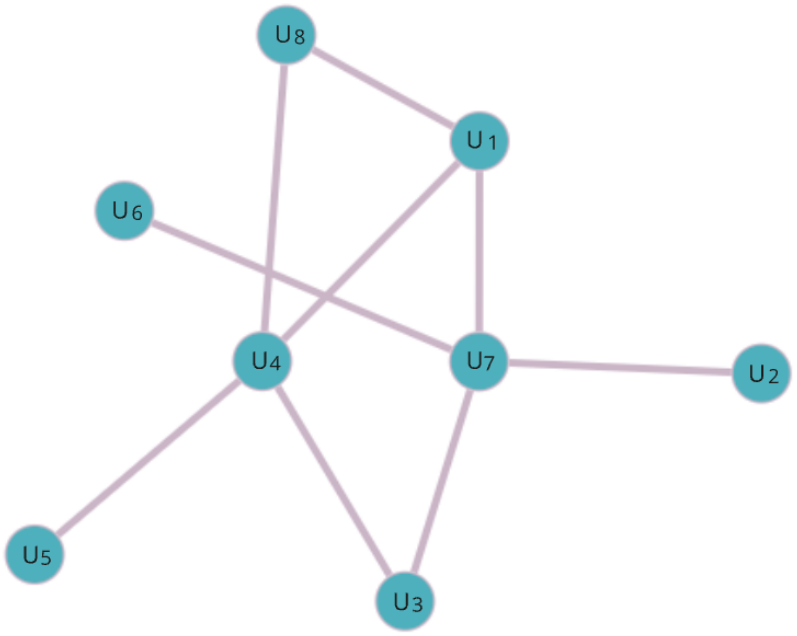


Figura 3.2: Grafo no dirigido y sin pesado correspondiente a la matriz de similitud entre 8 usuarios.

como el número de clústers a definir y la elección de la métrica de distancia. Además, el costo computacional del agrupamiento previo, especialmente en conjuntos de datos grandes, puede representar una limitación práctica en contextos donde se requiere escalabilidad.

3. Selección Aleatoria:

Este enfoque implica la selección aleatoria de q anclas a partir de las observaciones de $\mathbb{X}$. Aunque es sencillo y rápido, no aprovecha la información disponible sobre los usuarios, por lo que los representantes seleccionados podrían no reflejar adecuadamente las características de cada subcomunidad, limitando el potencial de los modelos locales para capturar patrones específicos. Este método de selección solamente es considerado como referencia para evaluar otros métodos.

Segunda Estrategia: Selección de anclas utilizando información adicional y reducción dimensional

La estrategia que proponemos en esta tesis tiene como objetivo mejorar la selección de anclas usando más información sobre los usuarios. A diferencia de los métodos que se basan únicamente en los vectores de interacción, esta estrategia busca enriquecer la representación de cada usuario incluyendo variables que capturen sus patrones de comportamiento y preferencias.

En particular, se consideran características de películas, como los géneros más populares en el conjunto de datos, junto con métricas agregadas como el número total de películas vistas por usuario y su calificación promedio por género (ver sección 5.1). Estas variables permiten construir un perfil más completo de cada usuario, que luego puede ser utilizado para identificar representantes más diversos y significativos como anclas.

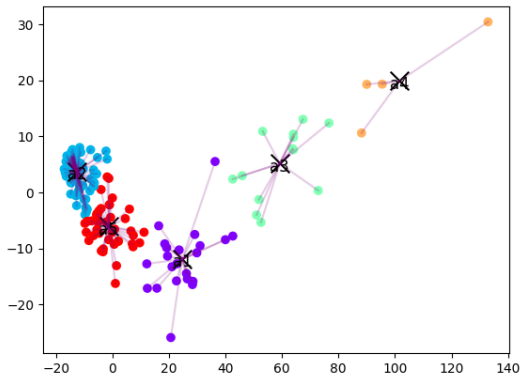
Para facilitar el análisis y la selección en este espacio enriquecido, se aplica una técnica de reducción de dimensionalidad que proyecta los perfiles de usuario a un subespacio más compacto y estructurado. Este subespacio conserva las relaciones latentes más relevantes entre los usuarios, y permite aplicar métodos de agrupamiento o selección por densidad para identificar usuarios representativos. Con ello, se espera que los modelos locales construidos sobre estas anclas capturen mejor la diversidad estructural presente en la población de usuarios.

La transformación resultante condensa la información de calificaciones e incorpora aspectos adicionales sobre las preferencias de los usuarios, facilitando su análisis. Posteriormente, sobre esta matriz de datos aumentada se aplican técnicas de reducción de dimensionalidad como ISOMAP, UMAP y t-SNE (ver A.1.2) para transformar los datos a un espacio de menor dimensión.

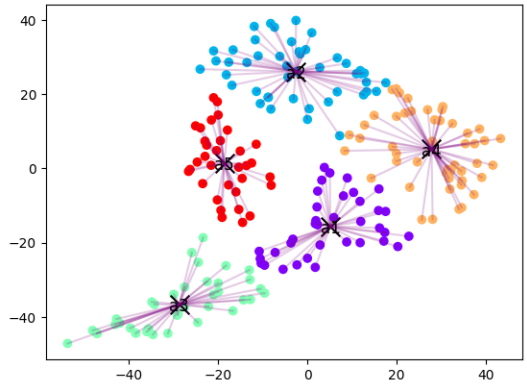
Tras reducir la dimensionalidad, se aplican métodos de agrupamiento no supervisado como K-means y agrupamiento jerárquico (ver A.1.3 y A.1.4) sobre la represen-

tación reducida. Estas técnicas permiten identificar grupos de usuarios con similitudes latentes.

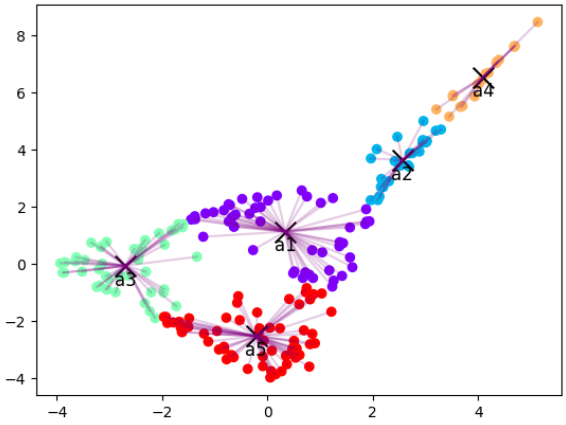
Los centroides de cada grupo se seleccionan como anclas, sirviendo como representantes de los modelos locales. En las figuras 3.3a, 3.3b y 3.3c se ilustran 5 aprupaciones de 200 usuarios de MovieLens con K-medias sobre la representación reducida resultante de: ISOMAP, t-SNE y UMAP. Los puntos x indican las anclas seleccionadas para cada grupo, y las líneas muestran la distancia de cada usuario a su respectiva ancla.



(a) Agrupamiento K-Medias sobre representación ISOMAP.



(b) Agrupamiento K-Medias sobre representación t-SNE.



(c) Agrupamiento K-Medias sobre representación UMAP.

Figura 3.3: Agrupamiento por K-Medias de 200 usuarios aleatorios del conjunto de datos descrito en la sección 5.1 utilizando diferentes técnicas de reducción de dimensionalidad (ISOMAP, t-SNE y UMAP). Los puntos representados con x' son las anclas respectivas de cada grupo, y cada recta muestra la distancia de cada usuario hacia su respectiva ancla.

Los detalles sobre cómo se construyen y entrenan los modelos locales a partir de estas anclas se presentan en el Capítulo 5.

3.3. Entrenamiento de Modelos Locales

Cada modelo local se entrena en una subcomunidad de usuarios con su peso vector de pesos $t^{(j)}$ correspondiente. La función objetivo para el aprendizaje de un modelo local se formula de la siguiente manera:

$$\underset{\theta^{(j)}}{\operatorname{argmin}} \sum_{x_u \in \mathbb{X}} t_u^{(j)} \mathcal{L}\left(x_u, M^{\text{local}}\left(x_u; \theta^{(j)}\right)\right) + \lambda \Omega\left(\theta^{(j)}\right),$$

donde $\mathcal{L}$ representa a la función de costo del problema de recomendación, x_u es el vector de interacción del usuario u con todos los I ítems en $\mathbb{X}$, $M^{\text{local}}(x_u, \theta)$ es el modelo local de recomendación parametrizado por θ con $M^{\text{local}}(x, \theta) : \{0,1\}^{U \times I} \rightarrow \mathbb{X}^{U \times I}$ y $t_u^{(j)}$ es el peso escalar para el vector de interacción del usuario u, x_u. Cuando $t_u^{(j)}$ es cero, el usuario u correspondiente no es tomado en cuenta para el entrenamiento del modelo local. Por otro lado cuando todos los componentes de $t^{(j)}$ son 1, es equivalente al entrenamiento del modelo global.

Para el entrenamiento, se elige el ancho de banda h_T para abarcar a un número suficientemente grande de usuarios. Choi y cols. (2021) mencionan que considerar más vecinos con un ancho de banda mayor que h_W resulta en la captura de dependencias entre usuarios. Cuando h_T es demasiado pequeño, la mayoría de los $t_u^{(j)}$ se vuelven cero, dando como resultado una submatriz extremadamente pequeña para entrenar. Esto hace que sea difícil capturar patrones locales ocultos de los usuarios, lo que conlleva a un rendimiento subóptimo.

3.4. Combinando Modelos Locales

El modelo LOCA combina las predicciones generadas por múltiples modelos locales mediante un esquema de agregación ponderada, donde los pesos asignados a cada modelo dependen de la similitud entre el usuario objetivo y el ancla que define dicho modelo local.

A diferencia de los modelos tradicionales, para el entrenamiento LOCA utiliza un conjunto de pesos $\mathcal{W}$ diferente a $\mathcal{T}$. En la ecuación (3.2), se utiliza un umbral de prueba $h_{\mathcal{W}}$ el cual puede ser más pequeño que el parámetro $h_{\mathcal{T}}$ utilizado en el entrenamiento. Esta diferencia permite que el modelo aprenda con una vecindad más amplia y estable durante el entrenamiento, mientras que en la prueba se concentre en un conjunto más restringido de usuarios altamente similares al objetivo. Esta estrategia, propuesta por Choi y cols. (2021), mejora la especialización de las predicciones sin sacrificar la generalización del modelo.

Dado que cada modelo local descubre subcomunidades coherentes relativamente pequeñas, su combinación puede no garantizar la cobertura de todo el conjunto de usuarios. Para abarcar este problema se entrena también un modelo global sobre el conjunto completo de usuarios, el cual aprende correlaciones globales entre todos los usuarios, y es equivalente a asignar pesos iguales a todos los usuarios. Para obtener esta cobertura, se entrena el modelo global y se lo combina con los modelos locales.

Siguiendo la propuesta de Choi y cols. (2021), la combinación de predicciones se realiza mediante un esquema de regresión no paramétrica de tipo Nadaraya-Watson (Nadaraya, 1964; Watson, 1964), el cual permite estimar la salida como un promedio ponderado de las predicciones locales, sin necesidad de especificar una forma funcional explícita. Si denotamos por $\hat{\mathbb{X}}_{\text{global}}$ la predicción del modelo global y por $\hat{\mathbb{X}}^{(j)}_{\text{local}}$ la predicción del modelo local centrado en el ancla $a^{(j)}$, la predicción final se define como:

$$\hat{\mathbb{X}} = \alpha \hat{\mathbb{X}}_{\text{global}} + (1-\alpha) \sum_{j=1}^{q} w^{(j)} \odot \hat{\mathbb{X}}^{(j)}_{\text{local}} \oslash w.$$

El término $w^{(j)}$ representa el vector de pesos de agregación asociados al modelo local j, y se calcula evaluando la función kernel $K_{h_{\mathcal{W}}}$ sobre la distancia entre el usuario objetivo u y el ancla $a^{(j)}$, es decir:

$$w_u^{(j)} = K_{h_{\mathcal{W}}}\left(\text{dist}(a^{(j)}, u)\right) \qquad \text{(ver ecuación 3.4)}.$$

La suma total de los pesos para cada usuario se define como $w = \sum_{j=1}^{q} w^{(j)}$, y se utiliza para normalizar la contribución relativa de cada modelo local. Los símbolos $\odot$ y $\oslash$ denotan el producto y la división elemento a elemento, respectivamente. Gracias a la flexibilidad del esquema de regresión no paramétrica, los pesos $w^{(j)}$ se ajustan automáticamente según la ubicación del usuario objetivo en el espacio latente, adaptando así la combinación final a las características tanto locales como globales de los datos.

El entrenamiento del modelo global, tiene como objetivo minimizar el error de reconstrucción, el cual está definido por la siguiente función de pérdida:

$$\underset{\theta^{(g)}}{\text{argmin}} \sum_{x_u \in \mathbb{X}} \mathcal{L}\left(x_u, M^{\text{global}}\left(x_u; \theta^{(g)}\right)\right) + \lambda \Omega\left(\theta^{(g)}\right),$$

donde x_u es el vector de interacción del usuario u en el conjunto $\mathbb{X}$ y $\theta^{(g)}$ son los parámetros del modelo global $M^{\text{global}}(\cdot)$.

En este capítuo revisamos la definición de los modelos locales, el mecanismo de selección de anclas, la construcción de subcomunidades y la combinación final de predicciones para abarcar el marco completo del modelo LOCA, enfoque permite integrar de manera efectiva tanto estructuras globales como patrones locales en los datos de interacción. En el siguiente capítulo, se presentan los experimentos diseñados

para evaluar el desempeño de esta propuesta en distintos conjuntos de datos.

Capítulo 4

Experimentos y Evaluación de Desempeño de LOCA

En este capítulo se exploran distintas configuraciones del modelo LOCA para encontrar su mejor desempeño. Se varían parámetros como el número de modelos locales y la dimensión de los embeddings utilizados, y se evalúan con diferentes métricas. El objetivo es comparar y analizar el rendimiento del modelo LOCA propuesto por Choi y cols. (2021) bajo estas estrategias, utilizando el conjunto de datos de MovieLens.

En la Sección 4.1, se realiza una breve descripción de los conjuntos de datos. La Sección 4.2, describe la preparación de los datos para entrenamiento y prueba, además se especifica la arquitectura considerada para los modelos. Posteriormente, en la Sección 4.3 se definen las métricas utilizadas para realizar la evaluación de los experimentos. Finalmente en 4.4, se comparan diferentes resultados al reparametrizar los modelos propuestos por el estado del arte.

Para el entrenamiento del modelo Mult-VAE, se implementó el algoritmo de Optimización Estocástica del ELBO (1), siguiendo la metodología propuesta en Kingma y Welling (2014). La implementación se realizó utilizando la librería pública PyTorch[1],

[1] https://pytorch.org

Dataset	Usuarios	Ítems	Ratings	Escala	Densidad
MovieLens 1M	6,040	3,706	1,000,209	[1-5]	4.47 %
MovieLens 10M	69,878	10,681	10,000,054	[1-5]	1.34 %

Tabla 4.1: Características de los conjuntos de datos. Fuente: Harper y Konstan (2015)
Nota: La única columna calculada, Densidad, representa el porcentaje de celdas en la matriz completa de elementos de usuario que contienen valores de calificación.

que ofrece una infraestructura eficiente y flexible para el desarrollo de algoritmos de aprendizaje profundo. Esta elección permitió aprovechar las capacidades de PyTorch para construir y entrenar el modelo Mult-VAE de manera efectiva, facilitando la optimización de los parámetros θ y ϕ mediante gradientes estocásticos.

4.1. Descripción del conjunto de datos

Los conjuntos de datos *MovieLens 1M* y *MovieLens 10M* son considerados referencias del mundo real en sistemas de recomendación para implementar modelos de manera práctica. MovieLens 1M está compuesto por $1,000,209$ calificaciones $R \in \{1, 2, 3, 4, 5\}$ de aproximadamente $3,706$ películas realizadas por $6,040$ usuarios. Por otro lado, MovieLens 10M contiene $10,000,054$ calificaciones $R \in \{1, 2, 3, 4, 5\}$ de aproximadamente $10,681$ películas realizadas por $69,878$ usuarios, considerando únicamente a aquellos usuarios que han calificado al menos 20 películas. La Tabla 4.1 proporciona detalles adicionales acerca de estos conjuntos de datos.

Con el fin de comparar y analizar los resultados experimentales de las propuestas mencionadas en el Capítulo 3, se utiliza principalmente el conjunto de datos *MovieLens 1M*; el cual cuenta con tres conjuntos de información: uno que contiene las calificaciones realizadas por los usuarios hacia los ítems, otro que describe los usuarios, y otro que detalla los ítems.

Descripción de información sobre calificaciones

Este conjunto de datos cuenta con las variables: UserID, MovieID, Rating y Timestamp. Cada variable tiene las siguientes características:

- **UserIDs**: rango de 1 a 6040 ($u \in \{1, 2, ..., 6040\}$).
- **MovieIDs**: rango de 1 a 3952 ($i \in \{1, 2, ..., 3952\}$).
- **Ratings**: en una escala del 1 al 5. (solo se consideran enteros).

Descripción de información sobre usuarios

MovieLens 1M presenta información geográfica y demográfica acerca de los usuario. En particular, se cuenta con las variables: UserID, Género, Edad, Ocupación y Código Postal. Estas variables tienen las siguientes características:

- **Género:** se denota con una "M" para hombre y una "F" para mujer.
- **Edad:** discretizada en los rangos:
  - 1: "Menor a 18"
  - 18: "18-24"
  - 25: "25-34"
  - 35: "35-44"
  - 45: "45-49"
  - 50: "50-55"
  - 56: "56+"

- **Ocupación:** con categorías:

$$\{0, 1, ..., 19\} := \{\text{“Otro” o no especificada, “Académico/a / Educador/a”, “Artista”, “Oficinista / Administrador/a”, “Estudiante de posgrado”, “Servicio al cliente”, “Doctor/a / Personal médico”, “Ejecutivo/a”, “Granjero/a”, “Amo/a de casa”, “Estudiante de edu. básica”, “Abogado/a”, “Programador/a”, “Retirado/a”, “Marketing”, “Científico/a”, “Trabajador/a autónomo/a”, “Técnico/a / Ingeniero/a”, “Desempleado/a”, “Escritor/a”}.}$$

Harper y Konstan (2015) mencionan que toda la información demográfica es proporcionada voluntariamente por los usuarios y no se verifica su precisión. Además, solo los usuarios que han proporcionado alguna información demográfica se incluyen en el conjunto de datos.

Descripción de información sobre películas

Las variables que se presentan para describir las películas son los Títulos y los Géneros. Los Títulos son idénticos a los títulos proporcionados por IMDB (incluyendo año de lanzamiento) y entre los Géneros considerados están: Acción, Aventura, Animación, Infantil, Comedia, Crimen, Documental, Drama, Fantasía, Cine Negro, Terror, Musical, Misterio, Romance, Ciencia Ficción, Suspenso, Guerra y Western.

4.2. Configuración Experimental

Existen varios enfoques para evaluar un modelo de SR, no hay una estrategia estándar de referencia. En particular, una estrategia de evaluación es la propuesta

por Liang y cols. (2018), donde parte del historial de interacciones de los usuarios retenidos (para validación y prueba) se utiliza para aprender las representaciones necesarias a nivel de usuario para el modelo. Posteriormente, se calculan métricas de evaluación observando qué tan bien clasifica el modelo el resto del historial de interacciones no vistas de los usuarios retenidos.

Para aplicar este enfoque, se evalúa el desempeño de los modelos teniendo en cuenta la escasez de datos (*sparsity*). Denotamos por $\text{Idx}_u = \text{Idx}_1, \text{Idx}_2, ..., \text{Idx}_n$ al conjunto de índices de los ítems con los que el usuario u ha interactuado. La evaluación del modelo se realiza dividiendo aleatoriamente la matriz de interacción $\mathbb{X}$ en los conjuntos $\mathbb{X}_{\text{train}}$ (entrenamiento) y $\mathbb{X}_{\text{test}}$ (evaluación de usuarios retenidos). La Figura 4.1 ilustra esta división, mostrando los datos de entrenamiento en rosa y los datos reservados para prueba en violeta.

Es importante asegurar que esta división sea representativa y equilibrada. En este contexto, como es habitual, el conjunto de datos $\mathbb{X} \in \mathbb{N}^{U \times I}$ se divide inicialmente en dos subconjuntos: un conjunto de entrenamiento $\mathbb{X}_{\text{train}}$, conformado por la mayoría de los usuarios, y un conjunto de evaluación $\mathbb{X}_{\text{test}}$, que contiene un subconjunto de usuarios reservados para validación y prueba (ver Figura 4.1). Para cada usuario

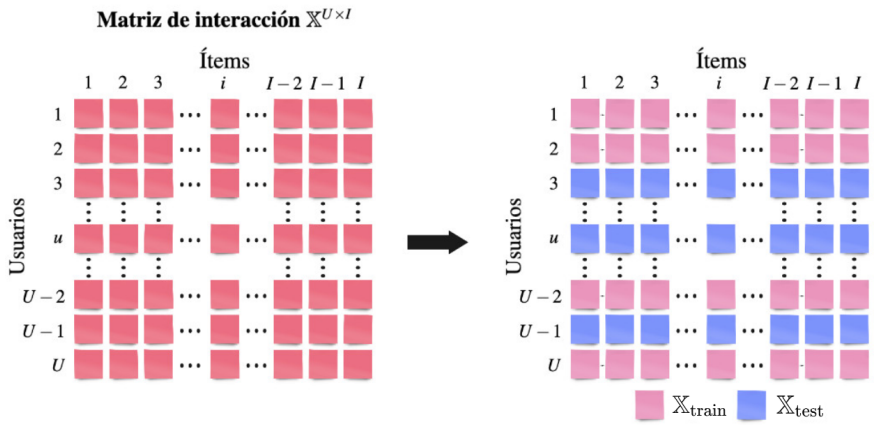


Figura 4.1: División del conjunto de datos en los conjuntos de entrenamiento y prueba.

$u \in \mathbb{X}_{\text{test}}$, se seleccionan aleatoriamente (sin reemplazo) algunos índices del conjunto de ítems Idx_u. Los elementos correspondientes a estos índices se ocultan y forman el

conjunto de validación $\mathbb{X}_{\text{eval}}$, que representa los ítems "no vistos" por el modelo para ese usuario. Estos se utilizan para evaluar la capacidad del modelo de recomendar ítems relevantes que el usuario aún no ha visto.

El resto de las interacciones visibles del usuario en $\mathbb{X}_{\text{test}}$ conforman el conjunto de evaluación $\mathbb{X}_{\text{eval}}$, el cual se utiliza como base para calcular las métricas de desempeño del modelo. Esta segunda partición del conjunto $\mathbb{X}_{\text{test}}$ se ilustra en la Figura 4.2.

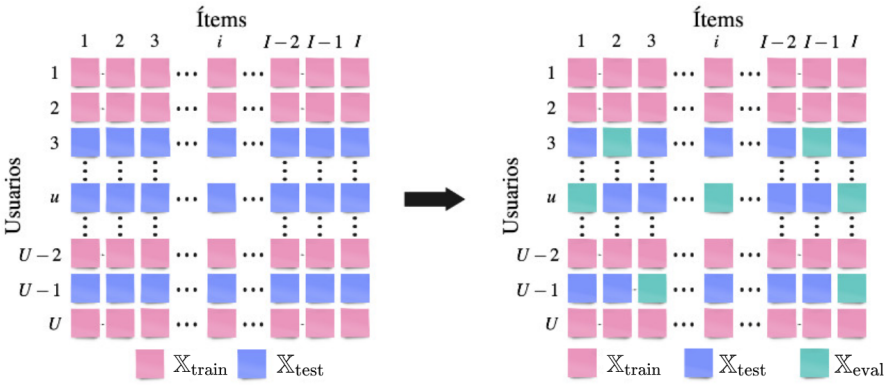


Figura 4.2: División del conjunto de prueba en los conjuntos de evaluación y validación.

Arquitectura del modelo Mult-VAE

La arquitectura del modelo Mult-VAE conserva la estructura tanto del modelo generativo $f_\theta(\cdot)$ como del modelo de inferencia $g_\phi(\cdot)$. Para los experimentos, se utilizó un perceptrón multicapa (MLP) con 0, 1 y 2 capas ocultas, ya que se observó que incrementar el número de capas no mejoraba el rendimiento del modelo. La dimensión de cada capa se puede ajustar; por ejemplo, una posible configuración para un modelo generativo MLP con una capa oculta en el Mult-VAE es la siguiente:

$$[I \to 600 \to \dim(z) \to 600 \to I],$$

donde 600 es la dimensión de las capas ocultas y $\dim(z)$ es la dimensión de la capa latente, la cual se variará en los experimentos descritos en la Sección 4.4.1.

4.3. Métricas de Evaluación

A la hora de evaluar la efectividad y el rendimiento de los SR de películas de MovieLens, es fundamental utilizar métricas de evaluación apropiadas (Gunawardana y Shani, 2009). Estas métricas nos permiten medir cuán precisas y relevantes son las recomendaciones generadas por el sistema, y nos ayudan a comparar y mejorar diferentes enfoques y algoritmos de recomendación (Herlocker, Konstan, Terveen, y Riedl, 2004).

En el ámbito de las recomendaciones, nuestro objetivo principal es generar un conjunto de R recomendaciones para un usuario. En nuestro caso, seleccionamos R ítems con la mayor probabilidad de ser de interés para cada usuario, ya que nuestros modelos Mult-VAE asignan una probabilidad de que un usuario interactuará con cierto ítem (Liang y cols., 2018).

Un SR usualmente devuelve una lista ordenada con ítems relevantes para cada usuario. Esta relevancia, en nuestro caso, es la interacción entre usuario e ítems. Las métricas de evaluación empleadas para evaluar nuestros SR son la Precisión@R, el Recall@R, y el NDCG@R, detalladas a continuación.

4.3.1. Precisión y Recall

Precision y recall son dos métricas fundamentales para evaluar la calidad de las recomendaciones generadas por un SR. Ambas métricas se basan en la relevancia de los ítems recomendados, pero difieren en su enfoque.

Precisión ($Prec@R$) mide la fracción de ítems relevantes en una lista de recomendaciones entre los primeros R ítems recomendados. Esto indica cuántos de los

ítems recomendados son realmente relevantes para el usuario:

$$Prec@R = \sum_{r=1}^{R} \frac{\text{rel}_r}{R},$$

donde rel_r es un valor binario que toma 1 si el ítem recomendado en la posición r es relevante y 0 en caso contrario.

Recall (*Recall@R*), por otro lado, evalúa la proporción de ítems relevantes correctamente identificados en las primeras R recomendaciones con respecto al número total N de ítems relevantes para el usuario. Esto proporciona una medida de cuántos de los ítems relevantes totales fueron efectivamente recomendados:

$$Recall@R = \sum_{r=1}^{R} \frac{\text{rel}_r}{N},$$

donde rel_r tiene la misma definición que para la precisión (Manning, Raghavan, y Schütze, 2008).

La diferencia principal entre ambas métricas reside en el denominador: la precisión considera la longitud de la lista de recomendaciones R, mientras que el recall se normaliza con respecto al total de ítems relevantes N en el conjunto de datos para cada usuario.

4.3.2. NDCG (Normalized Discounted Cumulative Gain)

Normalized Discounted Cumulative Gain (NDCG) es una métrica que evalúa la calidad de una lista de recomendaciones considerando no solo si los ítems relevantes están presentes, sino también en qué posición aparecen.

A diferencia de métricas como *Recall@R*, donde cada ítem relevante tiene el mismo peso sin importar su lugar en la lista, NDCG introduce una ponderación que otorga mayor valor a los ítems relevantes que aparecen en las primeras posiciones. Esta

ponderación refleja el supuesto de que los usuarios prestan más atención a los primeros resultados de una lista recomendada.

La fórmula que introduce esta ponderación es el *Discounted Cumulative Gain* (DCG), definido como:

$$DCG@R = \sum_{r=1}^{R} \frac{\text{rel}_r}{\log(r+1)},$$

donde rel_r representa la relevancia del ítem en la posición r. El denominador atenúa el aporte de los ítems conforme se alejan del inicio de la lista.

Posteriormente, NDCG aplica una normalización para facilitar la comparación entre usuarios o modelos con diferentes cantidades de ítems relevantes. Esta normalización se realiza dividiendo el $DCG@R$ obtenido por el $IDCG@R$, que representa el valor máximo posible de $DCG@R$ bajo un ordenamiento perfecto de los ítems relevantes:

$$NDCG@R = \frac{DCG@R}{IDCG@R},$$

donde $IDCG@R$ se calcula ordenando los ítems relevantes en el mejor orden posible, es decir, colocando los más relevantes en las primeras posiciones. El resultado es una métrica acotada entre 0 y 1, donde valores cercanos a 1 indican que la lista recomendada está muy cerca del orden ideal (Evidently AI team, 2024).

Tomemos un ejemplo de cálculo de DCG. Supongamos que tenemos una lista ordenada con puntuaciones binarias $[1, 0, 1, 0, 1]$, y queremos calcular $DCG@3$.

Figura 4.3: $DCG@3$ en lista de recomendaciones con puntuaciones binarias $[1, 0, 1, 0, 1]$

Podemos sumar las puntuaciones de relevancia de todos los ítems hasta la posición

$R = 3$, ponderadas por el descuento logarítmico correspondiente, obteniendo:

$$DCG@3 = \frac{1}{\log(1+1)} + \frac{0}{\log(1+2)} + \frac{1}{\log(1+3)} = 1.5$$

El ítem en la posición 1 es relevante y contribuye con $1/1 = 1$. No hay descuento en la posición 1. El ítem en la posición 2 es irrelevante, por lo que no contribuye con nada (0) y el ítem en tercera posición es relevante. La suma resultante es $1 + 0 + 0.5 = 1.5$. Este es el valor de DCG de la lista en $R = 3$.

En el caso de puntuaciones binarias, en IDCG el orden ideal es una lista con los ítems relevantes al inicio de la lista. Usando IDCG en el ejemplo de las puntuaciones binarias $[1, 0, 1, 0, 1]$, el orden ideal sería:

$$[1, 1, 1, 0, 0],$$

considerando que el cálculo de IDCG está dado por

$$IDCG@R = \sum_{r=1}^{R} \frac{\text{rel}_r}{\log(r+1)},$$

con $R = 3$ obtendríamos

$$IDCG@3 = \frac{1}{\log(1+1)} + \frac{1}{\log(1+2)} + \frac{1}{\log(1+3)} \approx 3.07$$

Entonces, el $NDCG@3$ sería:

$$NDCG@3 = \frac{DCG@3}{IDCG@3} \approx \frac{1.5}{3.07} \approx 0.488$$

4.4. Desempeño de LOCA con Mult-VAE

En esta sección se busca encontrar la representación ideal del embedding a partir de la exploración del comportamiento de LOCA al considerar diferentes configuraciones de sus parámetros. Uno de los principales parámetros considerados en este análisis es la dimensión de la representación latente z ($\dim(z)$), obtenida mediante el entrenamiento del modelo Mult-VAE.

A continuación presentamos distintos experimentos para entender el efecto sobre el desempeño de LOCA de:

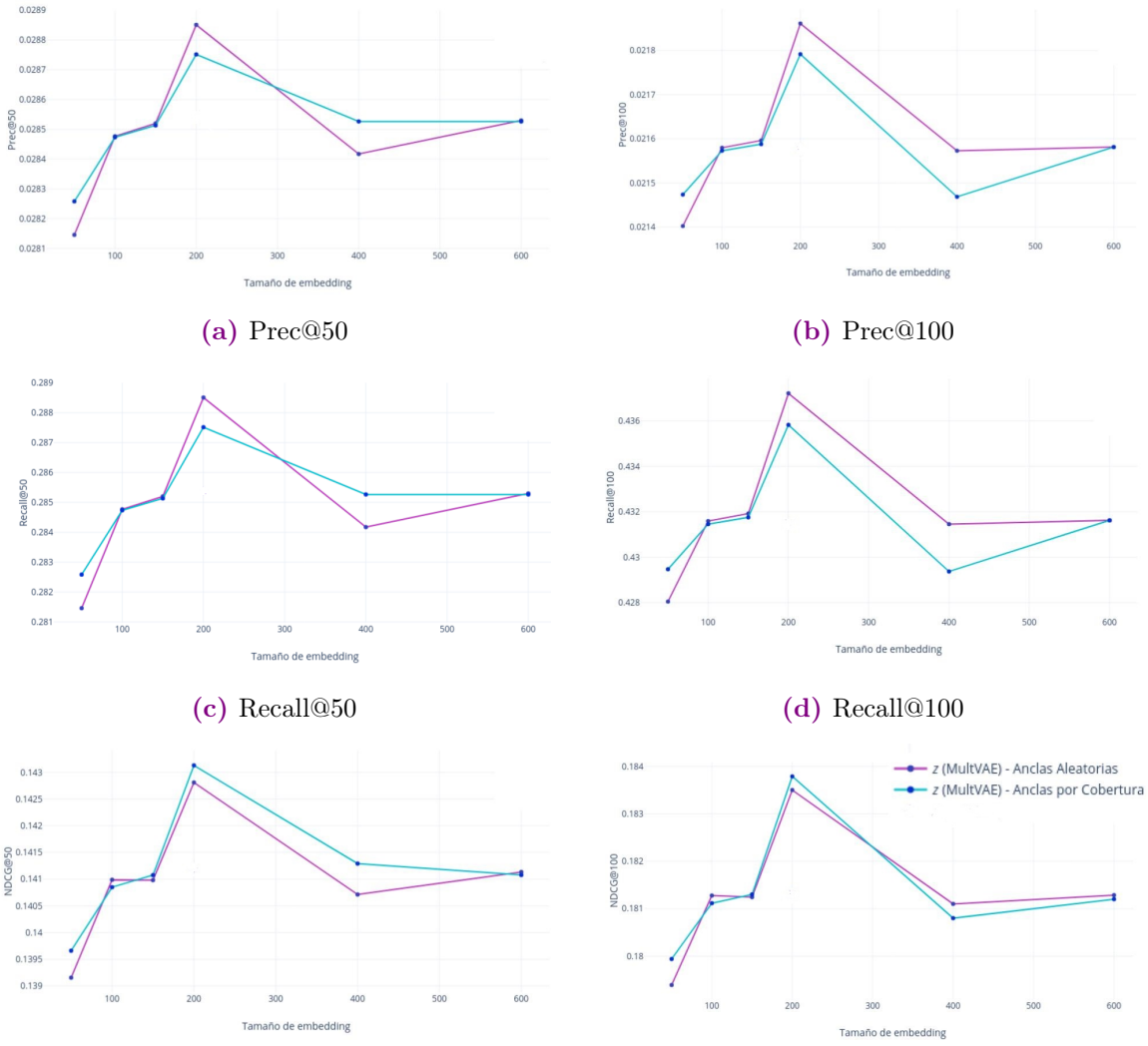
- la dimensión del embedding z,
- el método de selección de usuarios ancla mencionados en la Sección 3.2.2, y
- el número de modelos locales en el entrenamiento de LOCA.

Para esto, se realizan experimentos con diferentes configuraciones con cuatro ejecuciones de LOCA para cada configuración. Mult-VAE es el modelo considerado tanto para los modelos locales como para el global. Se presentan visualizaciones de los resultados como apoyo visual para identificar patrones, tendencias y comparaciones del rendimiento entre los distintos métodos y configuraciones implementadas. En la sección A.2 se presentan las tablas donde se resumen los experimentos realizados.

4.4.1. Resultados

La elección de representaciones latentes tiene un impacto significativo en el desempeño del modelo, en su capacidad de generalización y en realizar recomendaciones relevantes para los usuarios. Para determinar si las representaciones latentes enriquecen

la información capturada por el modelo LOCA se experimenta con representaciones de embeddings z con diferentes dimensiones generadas a partir de un Mult-VAE.



(a) Prec@50 **(b)** Prec@100

(c) Recall@50 **(d)** Recall@100

(e) NDCG@50 **(f)** NDCG@100

Figura 4.4: Desempeño del modelo bajo distintas configuraciones y métricas de evaluación. Las escalas de cada métrica varían entre sí debido a sus diferentes formas de normalización.

La Figura 4.4 muestra el desempeño promedio de 10 corridas en las métricas Prec@R, Recall@R y NDCG@R para $R = 50$ y $R = 100$, al entrenar LOCA con diferentes tamaños de representaciones latentes z generadas por un modelo Mult-VAE. Además, se compara el rendimiento de LOCA al seleccionar los usuarios ancla de manera aleatoria y mediante una estrategia basada en cobertura.

Se observa que las variables latentes z con tamaño 200 obtienen un mejor desempeño en todas las métricas; de hecho, Choi y cols. (2021) entrenan LOCA utilizando embeddings de esta dimensión. Por otro lado, no se evidencia una mejora al emplear la estrategia de cobertura frente a la selección aleatoria de anclas: ambas obtienen resultados muy similares, lo que sugiere que el criterio de cobertura no aporta una ganancia adicional en el rendimiento del modelo.

Otra de las configuraciones utilizadas fue la cantidad de modelos locales en LOCA. Por fines ilustrativos, se muestran los resultados de 4 corridas obtenidos al elegir usuarios ancla aleatorios para el entrenamiento de 2, 3, 4, 6, 8 y 18 modelos locales con anclas aleatorias en la Figura 4.5 y por anclas por cobertura en la Figura 4.6.

Además, se compara el rendimiento de LOCA al entrenarlo con embeddings de diferentes longitudes, es decir, variando la longitud de z. Nuevamente se observa que la representación latente de longitud 200 es la que mejor rendimiento tiene en todas las métricas; sin embargo los resultados entre cada corrida se ven más distantes entre sí que los de longitud 600 o por el embedding propuesto por Choi y cols. (2021).

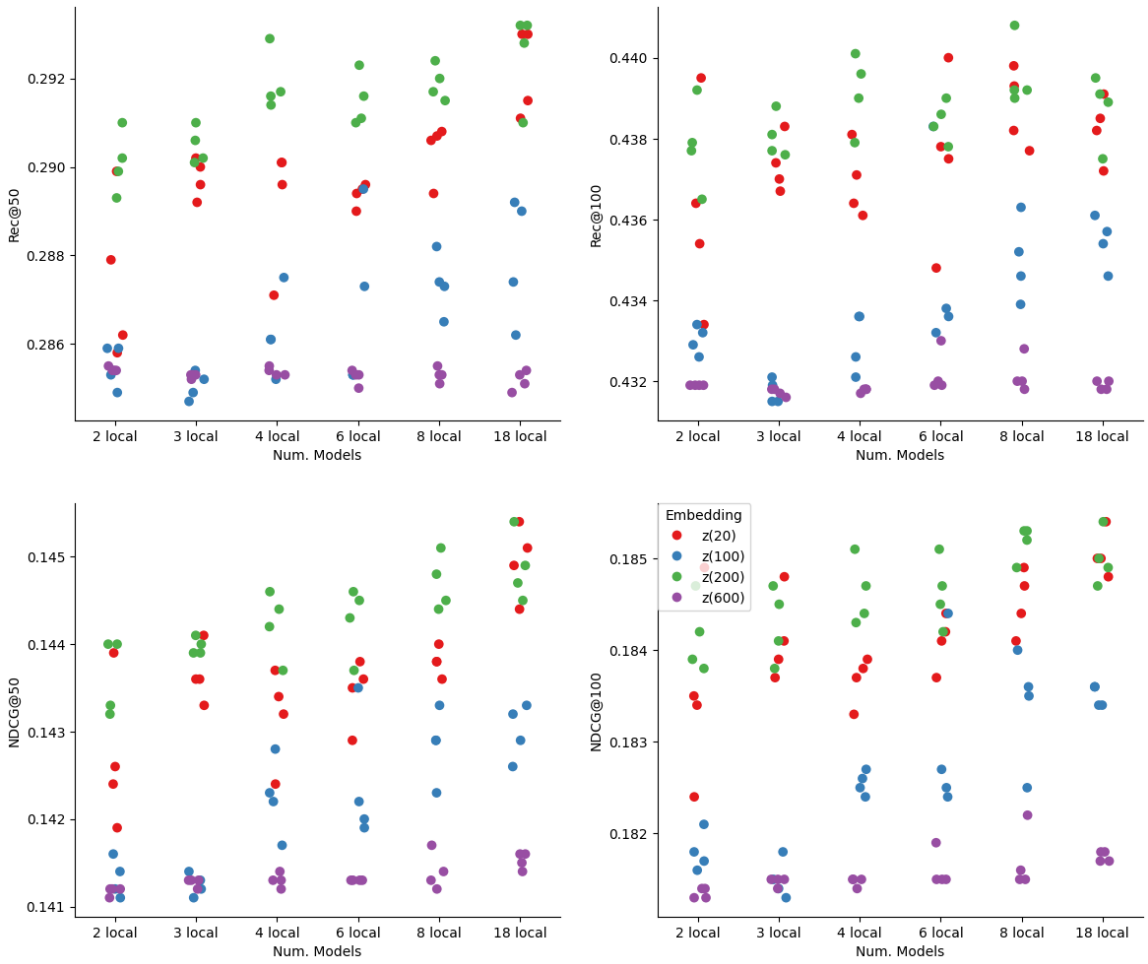


Figura 4.5: Rendimiento de 4 corridas de LOCA considerando tanto diferentes longitudes de z como diferentes números de modelos locales con usuarios ancla escogidos aleatoriamente.

4.4.2. Resumen de resultados experimentales

Los resultados experimentales presentados en las Figuras 4.4, 4.5 y 4.6 permiten extraer conclusiones relevantes sobre las decisiones de configuración del modelo LOCA:

- **Selección de anclas:** No se observa una diferencia significativa en el desempeño del modelo al seleccionar usuarios ancla de manera aleatoria o mediante una estrategia basada en cobertura. Esto sugiere que la cobertura no aporta

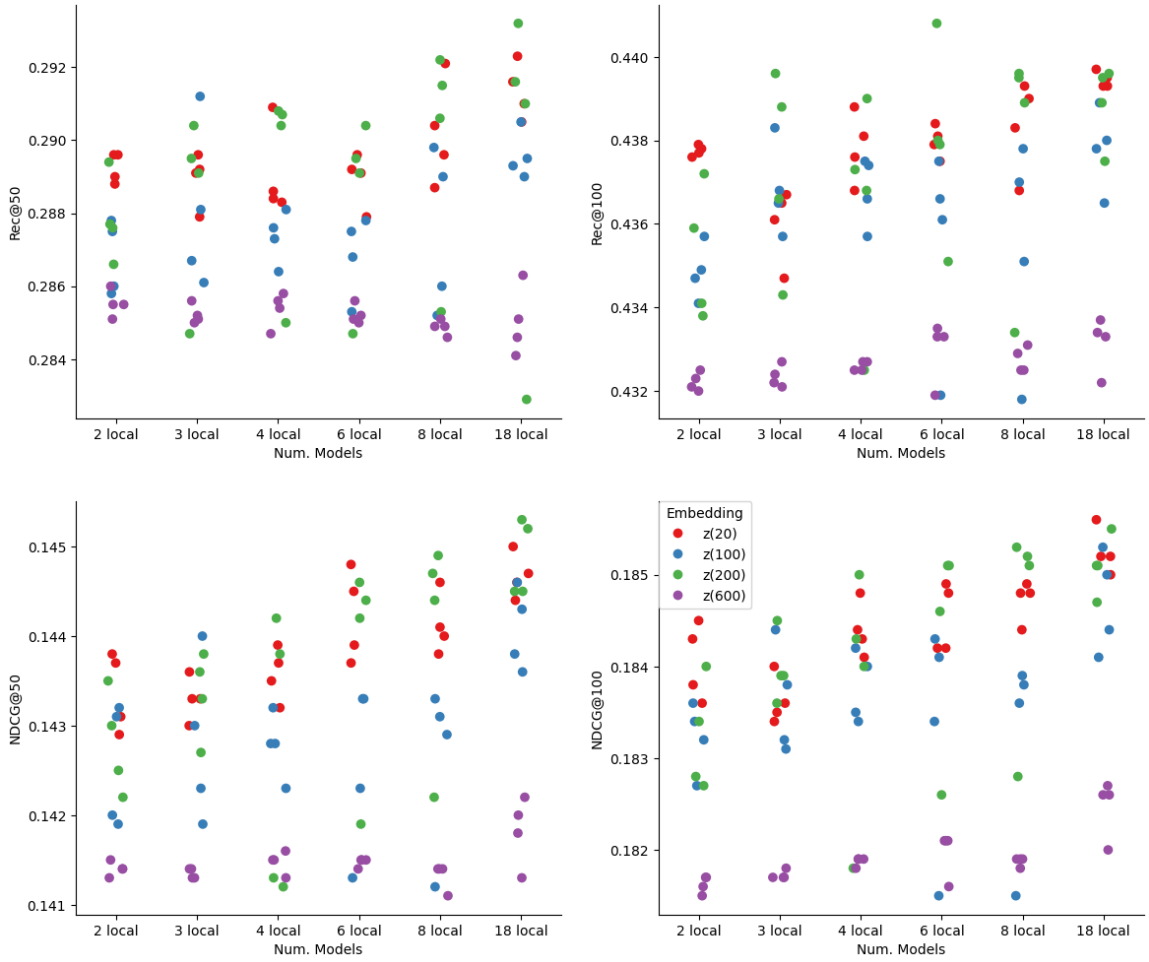


Figura 4.6: Rendimiento de 4 corridas de LOCA considerando tanto diferentes longitudes de z como diferentes números de modelos locales con usuarios ancla escogidos por cobertura.

información adicional relevante para mejorar la calidad de las recomendaciones, lo cual motiva la necesidad de explorar estrategias más efectivas en esta etapa.

- **Dimensión del embedding:** La representación latente z con dimensión 200 obtiene el mejor rendimiento en promedio en todas las métricas evaluadas. Esta configuración es consistente con la utilizada por Choi y cols. (2021). Sin embargo, se observa mayor variabilidad entre ejecuciones con esta dimensión en comparación con otras más grandes como 600.

- **Número de modelos locales:** Aumentar el número de modelos locales mejora la precisión de las recomendaciones. No obstante, esta mejora implica un mayor costo computacional, por lo que se debe buscar un balance entre rendimiento y

eficiencia.

En resumen, si bien la arquitectura del modelo y la dimensionalidad de las representaciones tienen un impacto claro en el desempeño, la estrategia actual de selección de usuarios ancla —ya sea aleatoria o por cobertura— no produce ganancias en el desempeño del modelo.

En el Capítulo 5 se propone una nueva estrategia de selección de anclas basada en agrupamiento sobre una matriz enriquecida, que incorpora no solo las interacciones históricas de los usuarios, sino también características adicionales relacionadas con sus gustos y perfiles. Esta propuesta busca mejorar la calidad de las recomendaciones optimizando la construcción de modelos locales más representativos y especializados.

Capítulo 5

Extensión de LOCA con selección de anclas basada en datos aumentados

En este capítulo se presenta una extensión del modelo LOCA cuyo eje principal es la selección estratégica de anclas. La idea central consiste en identificar y ubicar un conjunto reducido de anclas en posiciones clave dentro del espacio de usuarios, de manera que cada ancla actúe como un representante de una subregión específica. El objetivo es que este conjunto de anclas logre abarcar de forma adecuada la heterogeneidad de los datos, maximizando la diversidad cubierta y garantizando que las distintas subcomunidades de usuarios estén bien representadas.

Para ello, se incorporan datos aumentados que enriquecen la caracterización de usuarios e ítems, lo cual permite guiar el proceso de selección de anclas hacia opciones más informadas y robustas. De esta forma, cada modelo local asociado a un ancla se especializa en capturar patrones particulares de su subregión, contribuyendo a una segmentación más precisa del espacio latente.

Finalmente, se presentan los resultados experimentales de los distintos enfoques de selección propuestos y se analiza comparativamente su impacto en el desempeño del modelo LOCA, destacando cómo la elección de anclas bien distribuidas y diversas

puede mejorar la calidad global de las predicciones.

5.1. Construcción de nuevas variables

Para cada usuario, tenemos el género de cada pelicula con la que interactuó. Para aprovechar esta información pero sin aumentar de manera substancial la complejiad computacional, proponemos usar resúmenes que contienen los 7 géneros más frecuentes en el conjunto de datos: Drama, Comedia, Terror, Romance, Documental, Suspenso y Acción, ver Tabla 5.1. Estas variables permiten obtener información relevante acerca del tipo de películas con las que cada usuario ha interactuado. No obstante, es importante señalar que estas variables únicamente nos brindan información sobre la interacción del usuario con determinados géneros, pero no nos proporcionan información acerca del nivel de interés del usuario por dichos géneros.

Para recopilar esta información, se han generado variables suplementarias que indican las calificaciones totales que los usuarios han otorgado a los géneros más frecuentes. Por ejemplo, el usuario $u = 5$ ha visto 16 películas infantiles y 17 de aventura, con calificaciones totales de 42 y 55, respectivamente. Esta diferencia sugiere una preferencia hacia el género de aventura. Como las calificaciones pueden variar por género, calculamos la calificación total por género para cada usuario.

Además, consideramos la cantidad total de películas vistas por usuario. Esta variable nos proporciona información útil sobre la actividad de cada usuario en la plataforma, permitiendo comparar la actividad entre usuarios, identificar a los más activos y determinar los periodos de mayor actividad. Sin embargo, no proporciona información sobre los intereses específicos del usuario.

Finalmente, proponemos incluir una variable discretizada de edad como información adicional de los usuarios. Para ello, se definen tres categorías: $E_1 :=$ “Menor a 18”, $E_2 :=$ “18–50” y $E_3 :=$ “Mayor a 50”. Sin embargo, al analizar los intereses e inter-

Tabla 5.1: Conteo de películas por género

Género	Películas
Drama	843
Comedia	521
Terror	178
Comedia|Drama	162
Comedia|Romance	142
Drama|Romance	134
Documental	116
Suspenso	101
Acción	65
Drama|Suspenso	63
Acción|Suspenso	48
Infantil|Comedia	47
Crimen|Drama	44
Drama|Guerra	43
Romance	40
Acción|Drama	39
Animación|Infantil	35
Comedia|Drama|Romance	34
Western	33
Terror|Ciencia ficción	33

acciones en el conjunto de datos MovieLens 1M, no se observan diferencias en las preferencias entre los distintos grupos de edad. En particular, todos los grupos muestran una fuerte inclinación hacia los géneros de Comedia y Drama, como se observa en la Figura 5.1.

Una vez que se eligieron las variables representantes del conjunto de datos, se mapean los conjuntos de datos de calificaciones, películas y usuarios, obteniendo una

Tabla 5.2: Variables representantes calificaciones, usuarios y películas de los primeros 4 usuarios. $\overline{R}$ representa el promedio de calificaciones. Películas Totales es la suma del total de películas observadas para cada usuario. Edad tiene 3 categorías y género 2 categorías.

Animación	...	**Drama**	$\overline{R}$ **Animación**	...	$\overline{R}$ **Drama**	**Películas Totales**	**Edad**	**Género**
3	...	26	4.67	...	4.19	53	E_1	F
6	...	61	3.67	...	3.7	129	E_3	M
0	...	19	0	...	3.58	51	E_2	M
0	...	9	0	...	4.11	21	E_2	M

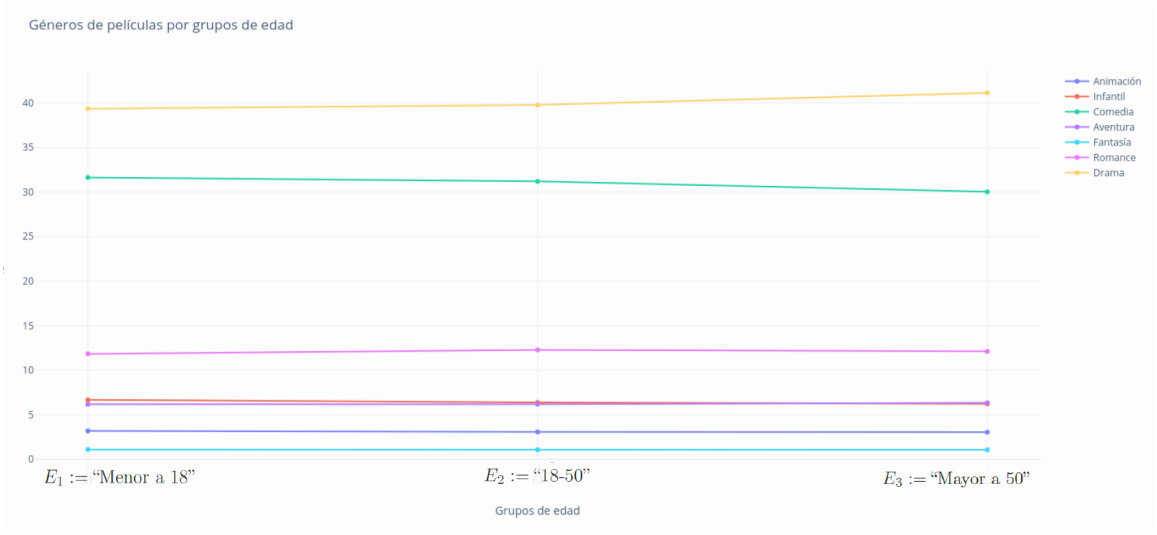


Figura 5.1: Distribución de género de películas para los grupos de edad E_1, E_2 y E_3, donde E_1 := "Menor a 18", E_2 := "18-50" y E_3 := "Mayor a 50"

matriz nueva de datos aumentados representada en la Tabla 5.2. Para fines experimentales, se denota a la matriz de características y preferencias de los usuarios por $\mathbb{M}^{U\times C_n}$, donde cada entrada m_{uc} representa el atributo del usuario $u \in \{1, 2, ..., U\}$ con la característica $c \in \{C_1, ..., C_n\}$ con $n = 17$, donde las variables de características son:

- $C_1, ..., C_7$:=cantidad de películas de animación, infantiles, comedia, aventura, fantasía, romance, drama, respectivamente.

- $C_8, ..., C_{14}$:=promedio de calificaciones de películas de animación, infantiles, comedia, aventura, fantasía, romance, drama

- C_{15} := Películas vistas totales.

- C_{16} := Edad.

- C_{17} := Género.

5.2. Selección de usuarios ancla

Dado nuestro interés en obtener representantes que reflejen la diversidad de patrones de comportamiento entre los usuarios, esta sección explora diferentes enfoques para seleccionar anclas como representantes de cada clúster de usuarios, obtenidos mediante agrupamientos sobre la matriz de datos aumentados. Para este fin, se agrupa sobre una representación en menor dimensión de la matriz $\mathbb{M}^{U \times C_n}$, la cual contiene un resumen de las calificaciones de películas y características de los usuarios (ver Sección 3.2.2).

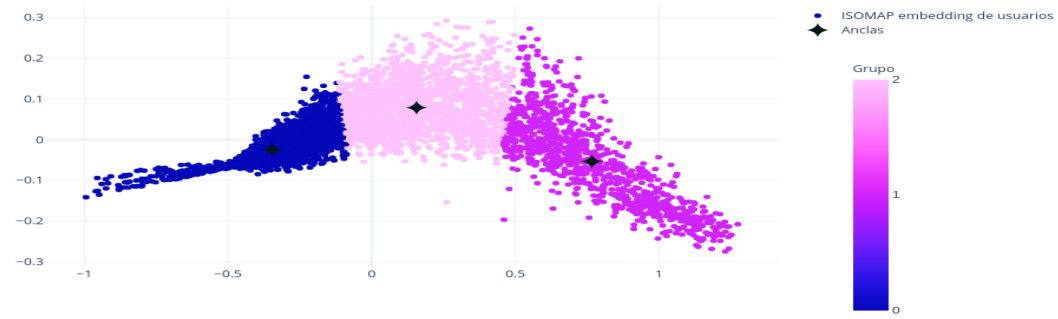
Para la reducción de dimensión, y con el fin de mejorar la separación entre grupos, eliminar ruido y facilitar la visualización, se aplicaron técnicas no lineales de reducción de dimensionalidad sobre la matriz $\mathbb{M}^{U \times C_n}$ antes de realizar el agrupamiento. Las técnicas empleadas fueron *ISOMAP*, *UMAP* y *t-SNE*, las cuales están diseñadas para preservar estructuras globales o locales del espacio original de datos.

ISOMAP busca mantener las distancias geodésicas entre pares de puntos, proporcionando una representación global coherente del espacio de usuarios (Xia y cols., 2021). Por otro lado, UMAP y t-SNE se enfocan en preservar relaciones de vecindad local, lo cual es útil para capturar agrupamientos naturales de usuarios en espacios latentes. Según Xia y cols. (2021), tanto UMAP como t-SNE son ampliamente utilizados en tareas de visualización y exploración de clústeres debido a su efectividad al separar grupos densos. Para mayor detalles sobre estos métodos de reducción de dimensión ver el Anexo A.1.2.

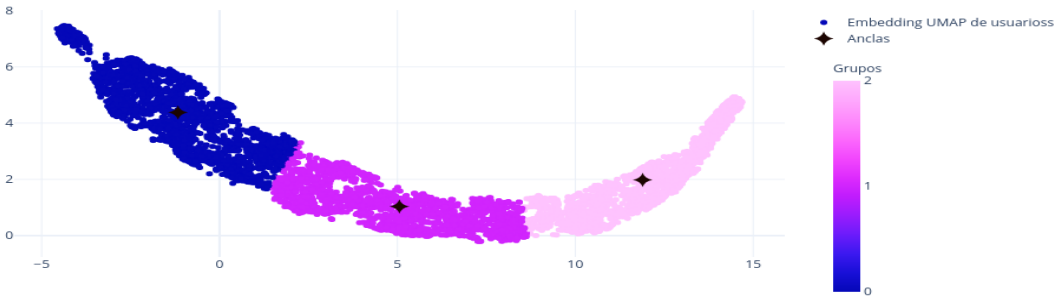
Agrupamiento sobre espacio reducido de datos originales

La Figura 5.2 muestra los resultados del agrupamiento aplicado mediante K-Means sobre los espacios reducidos a dos dimensiones usando cada una de las tres técnicas

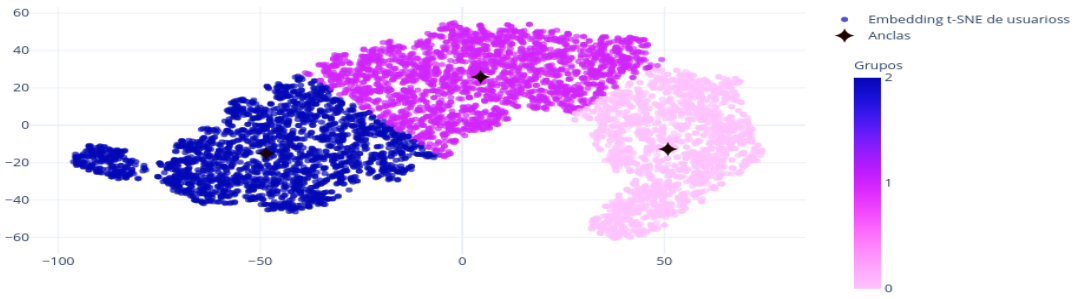
mencionadas. Las cruces negras marcan los usuarios ancla seleccionados como representantes de cada grupo.



(a) Agrupamiento de usuarios mediante ISOMAP y sus correspondientes usuarios ancla.



(b) Agrupamiento de usuarios mediante UMAP y sus correspondientes usuarios ancla.



(c) Agrupamiento de usuarios mediante t-SNE y sus correspondientes usuarios ancla.

Figura 5.2: Resultado de agrupamiento por K-Means en espacios reducidos a 2D

Como complemento, en la Figura 5.3 se presentan los resultados del agrupamiento jerárquico aplicado sobre los mismos espacios latentes generados por ISOMAP, UMAP y t-SNE. Se observa que los grupos resultantes son similares a los obtenidos por K-Means: están bien separados entre sí, aunque mantienen cierta cercanía en el espacio reducido, lo cual es esperable dado que los datos representan preferencias en un dominio compartido.

Finalmente, para interpretar las principales diferencias entre los grupos generados por las técnicas de reducción y agrupamiento, se utilizaron diagramas de caja. Estos permiten observar la distribución de las variables originales en $\mathbb{M}^{U\times C_n}$ para cada grupo de usuarios. En la Figura 5.4, se destacan diferencias importantes en variables como la cantidad de películas de comedia y drama vistas, el total de películas, y la edad de los usuarios. Estas características ayudan a entender los patrones subyacentes que definen cada grupo y justifican la selección de los usuarios ancla.

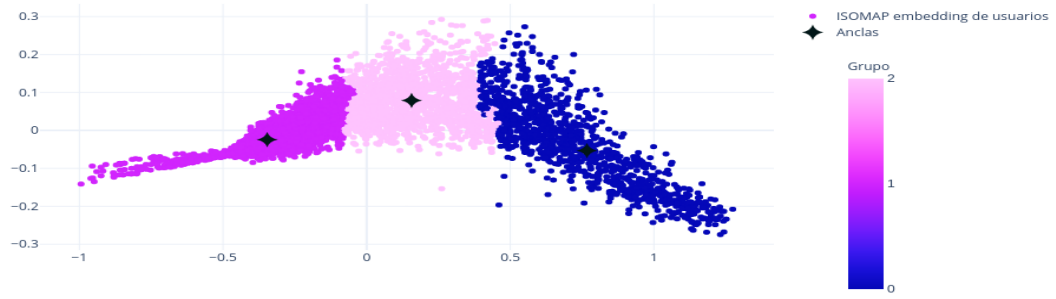
Agrupamiento sobre espacio reducido de datos normalizados

Como enfoque complementario, se implementó una técnica de normalización individual por subconjuntos de variables, previa a la reducción de dimensionalidad. Esta estrategia busca igualar las escalas de distintos tipos de datos (calificaciones, características de películas e información de usuarios), con el objetivo de reducir sesgos y permitir una comparación más justa entre subconjuntos heterogéneos.

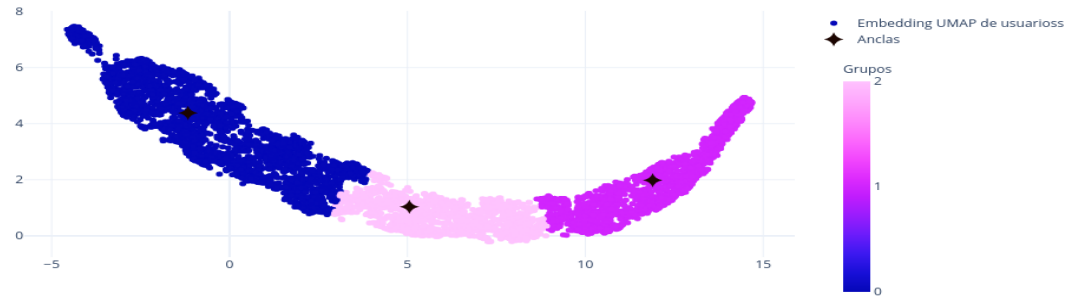
Para ello, la matriz aumentada $\mathbb{T} \in \mathbb{R}^{U\times N}$ fue descompuesta en tres subconjuntos:

$$\mathbb{T} = [\mathbb{X} \quad \mathbb{M}_1 \quad \mathbb{M}_2],$$

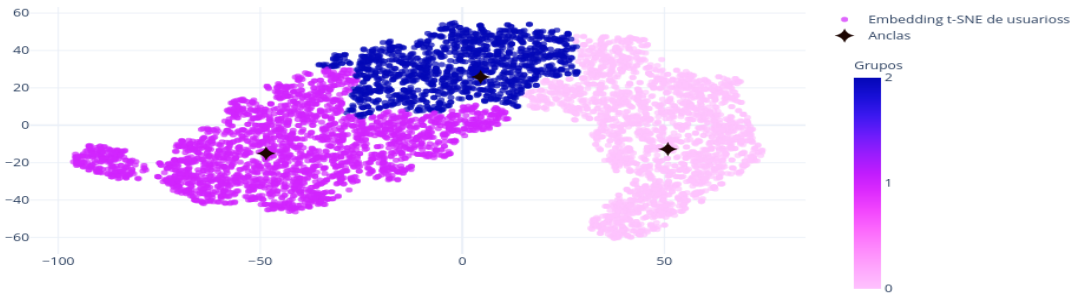
donde $\mathbb{X} \in \mathbb{R}^{U\times n_1}$: matriz de calificaciones por usuario, $\mathbb{M}_1 \in \mathbb{R}^{U\times n_2}$ representa las características de las películas (e.g., géneros, vistas), y $\mathbb{M}_2 \in \mathbb{R}^{U\times n_3}$ representa la información demográfica de los usuarios.



(a) Agrupamiento jerárquico en embedding generado por ISOMAP.



(b) Agrupamiento jerárquico en embedding generado por UMAP.



(c) Agrupamiento jerárquico en embedding generado por t-SNE.

Figura 5.3: Resultado de agrupamiento jerárquico en espacios reducidos

Cada subconjunto fue normalizado de forma independiente. Para ello, se seleccionaron aleatoriamente K muestras de cada conjunto y se calcularon las distancias

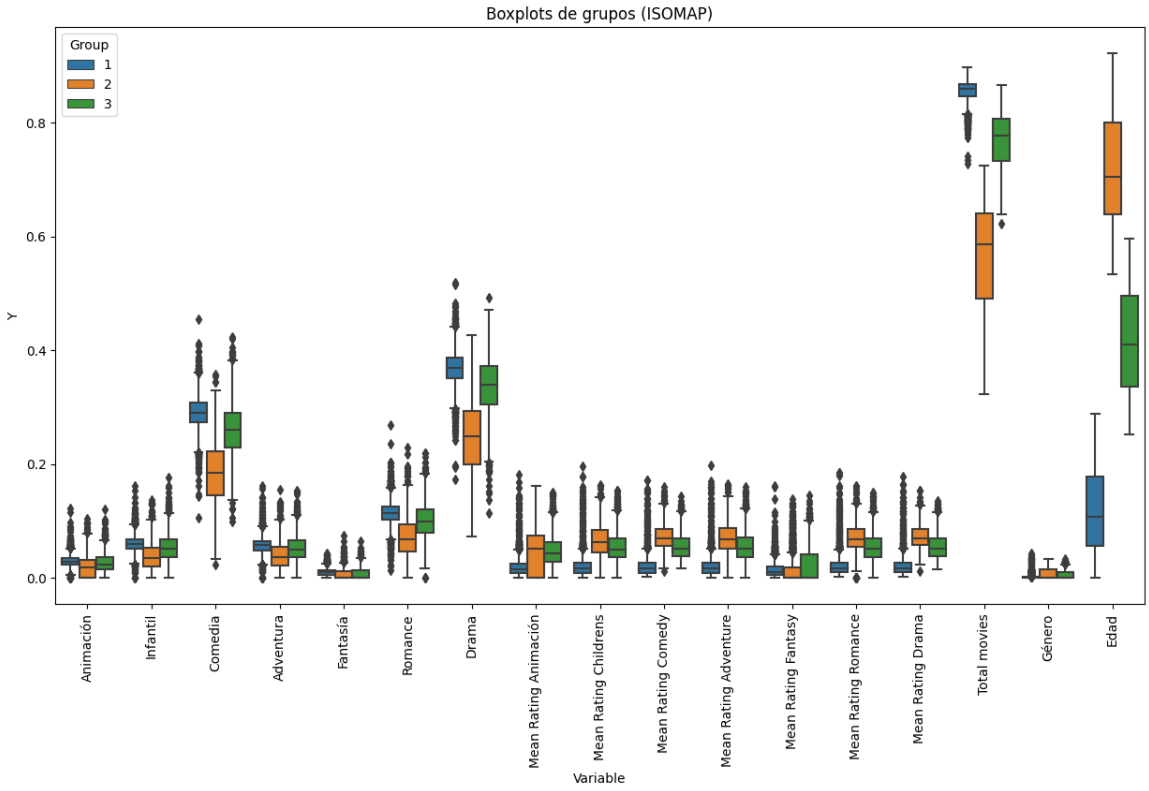


Figura 5.4: Boxplots por grupo en función de variables de la matriz aumentada (embedding de ISOMAP)

euclidianas entre todos los pares:

$$d_{ij} = \sqrt{\sum_{k=1}^{C} (x_{ik} - x_{jk})^2},$$

donde $x_i, x_j \in \mathbb{R}^C$ son dos vectores muestreados, y C es la cantidad de variables en el subconjunto correspondiente.

A partir de este conjunto de distancias se obtuvieron dos medidas de dispersión para cada subconjunto:

- La media de distancias:

$$\mu = \frac{1}{K(K-1)/2} \sum_{i=1}^{K} \sum_{j=i+1}^{K} d_{ij},$$

- La mediana m, definida como el valor central de la lista ordenada de distancias d_{ij}.

Estas medidas se utilizaron como centro de normalización. Luego, cada elemento x del subconjunto fue transformado según:

$$\hat{x}_\mu = \frac{x - \mu}{\text{std}(\mathbb{X})}, \qquad \hat{x}_m = \frac{x - m}{\text{MAD}(\mathbb{X})},$$

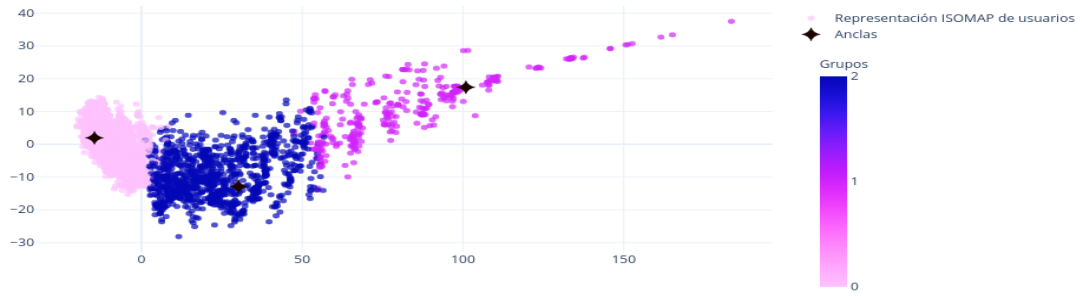
donde std$(\cdot)$ representa la desviación estándar, y MAD$(\cdot)$ la desviación absoluta mediana, calculadas sobre el conjunto completo correspondiente.

Una vez normalizados todos los subconjuntos, se construyeron las matrices concatenadas normalizadas:

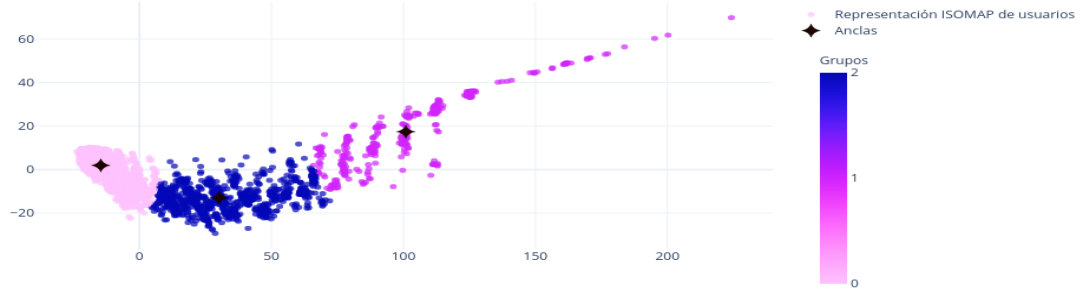
$$\hat{\mathbb{T}}_\mu = \begin{bmatrix} \hat{\mathbb{X}}_\mu & \hat{\mathbb{M}}_{1\mu} & \hat{\mathbb{M}}_{2\mu} \end{bmatrix}, \qquad \hat{\mathbb{T}}_m = \begin{bmatrix} \hat{\mathbb{X}}_m & \hat{\mathbb{M}}_{1m} & \hat{\mathbb{M}}_{2m} \end{bmatrix}.$$

Posteriormente, se aplicaron técnicas de reducción de dimensionalidad (ISOMAP, UMAP y t-SNE) sobre cada matriz normalizada, y se realizó el agrupamiento por K-Means en los espacios proyectados. Los resultados se presentan en las Figuras 5.5, 5.6 y 5.7.

Los agrupamientos obtenidos presentan una mejor separación visual entre grupos, particularmente en el caso de UMAP. Esto sugiere que la normalización individual ayuda a preservar la estructura relativa dentro de cada tipo de variable, reduciendo la dominancia de atributos con mayor varianza o escala. En la Figura 5.8 se observa que las variables más influyentes para diferenciar los grupos siguen siendo el total de películas vistas, los géneros más comunes (comedia, drama, aventura) y la edad del usuario.



(a) Agrupamiento resultante de representación ISOMAP de normalización mediante media de distancias de grupos de variables y sus correspondientes usuarios ancla.

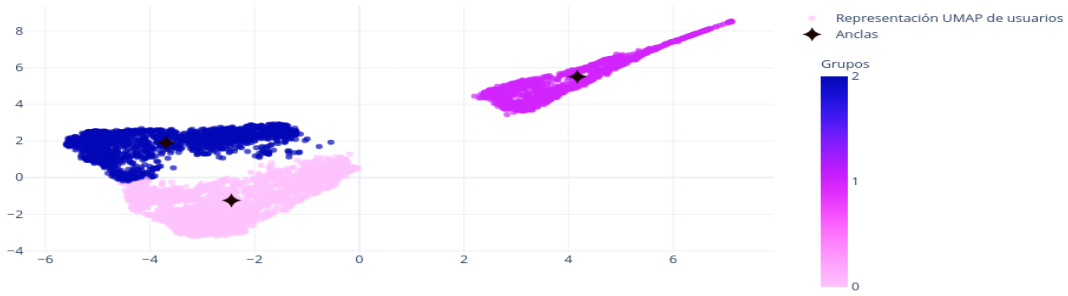


(b) Agrupamiento resultante de representación ISOMAP sobre normalización mediante mediana de distancias de grupos de variables y sus correspondientes usuarios ancla.

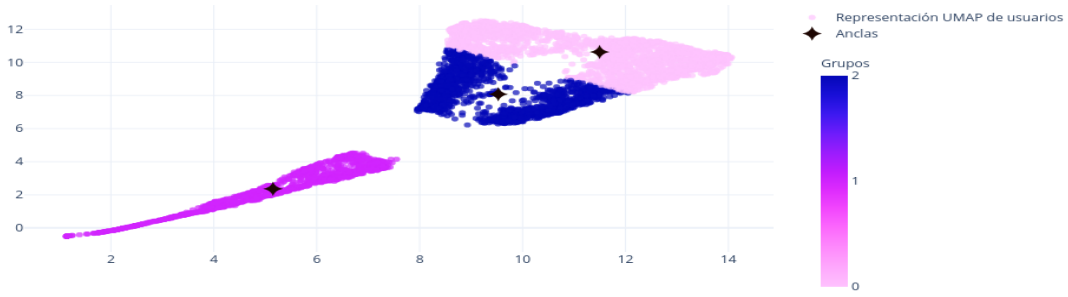
Figura 5.5: Resultado de agrupamiento de representación ISOMAP sobre usuarios normalizados mediante media y mediana de grupos de variables

5.3. Resultados

Los experimentos muestran que la manera en que se seleccionan las anclas tiene un impacto directo en el rendimiento del modelo LOCA. En primer lugar, observamos que el método de selección por cobertura no aporta mejoras sustanciales respecto a la elección aleatoria: en la práctica, ambos enfoques ofrecen resultados muy similares en todas las métricas de evaluación, lo que sugiere que este criterio es deficiente como



(a) Agrupamiento resultante de representación UMAP sobre normalización mediante media de distancias de grupos de variables y sus correspondientes usuarios ancla.



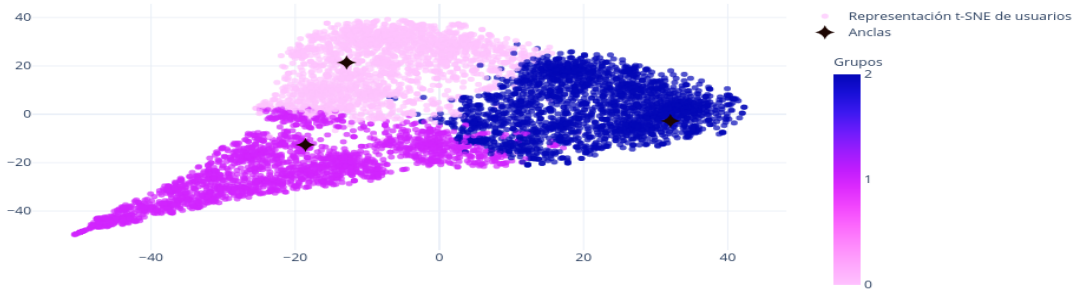
(b) Agrupamiento resultante de representación UMAP sobre normalización mediante mediana de distancias de grupos de variables y sus correspondientes usuarios ancla.

Figura 5.6: Resultado de agrupamiento de representación UMAP sobre usuarios normalizados mediante media y mediana de grupos de variables

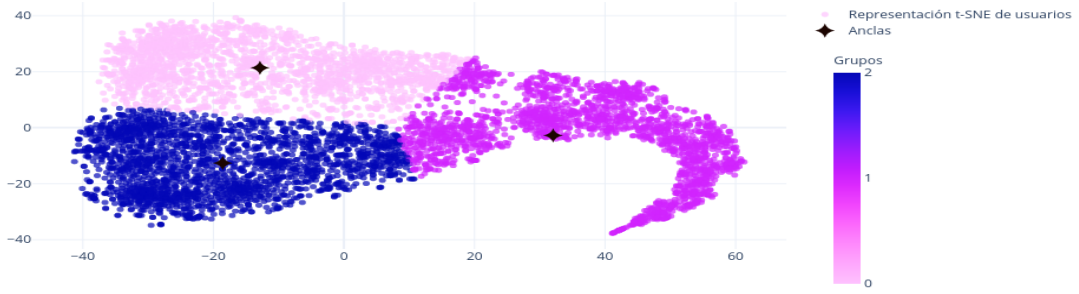
estrategia para capturar la heterogeneidad de los datos.

En contraste, al elegir las anclas de manera estratégica, incorporando información adicional y técnicas de reducción de dimensionalidad, se logran ganancias consistentes en el desempeño. En particular, los enfoques que introducen covariables en el modelo bayesiano representan un camino alternativo, ya que permiten que los modelos locales capturen con mayor precisión la diversidad en las preferencias y patrones de los usuarios.

En la Tabla 5.3 se presentan los promedios obtenidos sobre diez corridas indepen-



(a) Agrupamiento resultante de representación t-SNE sobre normalización mediante media de distancias de grupos de variables y sus correspondientes usuarios ancla.



(b) Agrupamiento resultante de representación t-SNE sobre normalización mediante mediana de distancias de grupos de variables y sus correspondientes usuarios ancla.

Figura 5.7: Resultado de agrupamiento de representación t-SNE sobre usuarios normalizados mediante media y mediana de grupos de variables

dientes de cada configuración experimental, mostrando el contraste entre los métodos de referencia y las propuestas basadas en embeddings enriquecidos, agrupamiento, normalización y reducción de dimensionalidad.

En cuanto al detalle de los experimentos:

- **Experimentos 1 y 2**: corresponden al baseline propuesto por Choi y cols. (2021), con embeddings de dimensión 200 y selección de anclas de manera aleatoria y por cobertura, respectivamente. Como se aprecia en la Tabla 5.3, ambas estrategias producen resultados prácticamente idénticos en todas las métricas, lo

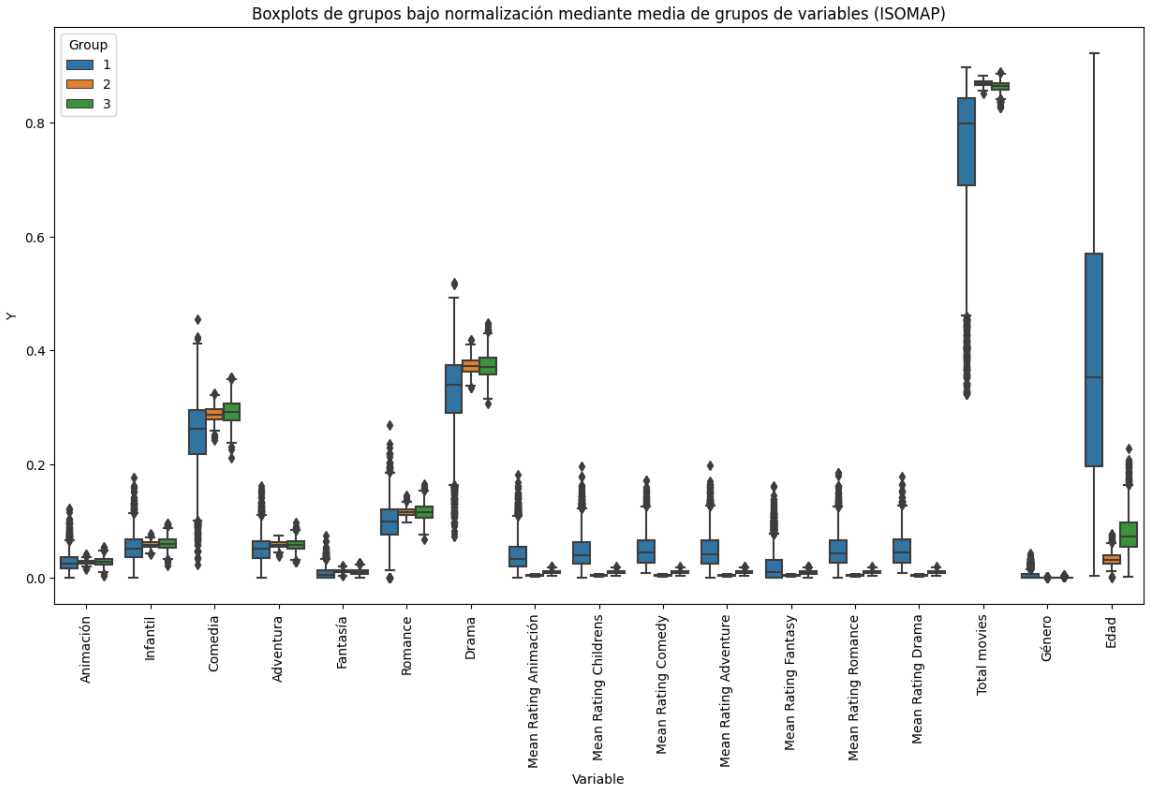


Figura 5.8: Boxplots de grupos, obtenidos después de normalización mediante medias de conjuntos de datos, por variables sobre representación de usuarios de ISOMAP

cual refuerza la idea de que el criterio de cobertura no es superior a la selección aleatoria.

- **Experimentos 3 y 4**: emplean como embedding la variable latente z_{200} generada por un modelo MultVAE, nuevamente con selección aleatoria y por cobertura de anclas. Estos resultados muestran una ligera pero consistente mejora frente al baseline de Choi y cols. (2021), lo que indica que el uso de embeddings más expresivos sí aporta al rendimiento.

- **Experimentos 5 al 7**: introducen la selección de anclas por agrupamiento sobre la matriz de características y preferencias de usuarios $\mathbb{M}^{U \times C_n}$ (ver Sección 5.1). Para este agrupamiento, se aplicaron métodos de reducción de dimensionalidad como ISOMAP, UMAP y t-SNE. Dentro de este bloque, destacan los experimentos 6 y 7, donde UMAP y t-SNE alcanzan mejores valores en las métricas de evaluación, evidenciando que estas técnicas permiten encontrar anclas

más representativas de la diversidad de los usuarios.

- **Experimentos 8 al 13**: incorporan además normalización de los grupos de variables de $\mathbb{M}^{U \times C_n}$ (ver Sección 5.2), considerando tanto la media como la mediana. Aunque el experimento 8 (ISOMAP con normalización por media) no supera al baseline, los demás muestran mejoras, particularmente el experimento 11 (ISOMAP con normalización por mediana) y el experimento 12 (UMAP con normalización por mediana), que consiguen desempeños sólidos en todas las métricas.

La mayoría de los experimentos propuestos superan al baseline en distintas métricas de evaluación, destacando en especial los que incorporan UMAP y t-SNE. Cabe señalar que el experimento 8 es la única excepción, ya que no logra mejorar el rendimiento respecto al baseline, mientras que el resto de configuraciones presentan incrementos, aunque de distinta magnitud, en precisión, cobertura y NDCG.

Tabla 5.3: Resultados sobre 10 repeticiones de cada experimento, en los que incluyen la propuesta del baseline y de la propuesta

Experimento	Embedding	Anchors	Dim. Reduction	Normalización	Métrica					
					Prec@50	Prec@100	Recall@50	Recall@100	NDCG@50	NDCG@100
1	Baseline	Aleatorios	-	-	0.02853	0.02158	0.28527	0.43161	0.14108	0.18123
2	Baseline	Cobertura	-	-	0.02854	0.02158	0.28536	0.43152	0.1412	0.18131
3	z_{200}	Aleatorios	-	-	0.02891	0.02184	0.28906	0.43672	0.14335	0.18386
4	z_{200}	Cobertura	-	-	0.02875	0.02174	0.28754	0.43474	0.14253	0.18289
5	z_{200}	Agrupamiento	ISOMAP	-	0.02894	0.02187	0.2894	0.43736	0.14343	0.18402
6	z_{200}	Agrupamiento	UMAP	-	0.02894	**0.02188**	0.2894	**0.43762**	0.14332	0.18398
7	z_{200}	Agrupamiento	t-SNE	-	0.02896	0.02185	0.28957	0.43698	**0.14385**	**0.18429**
8	z_{200}	Agrupamiento	ISOMAP	Media	0.02847	0.02148	0.28469	0.42959	0.14174	0.18149
9	z_{200}	Agrupamiento	UMAP	Media	0.02894	0.02187	0.28941	0.43748	0.14332	0.18393
10	z_{200}	Agrupamiento	t-SNE	Media	0.02878	0.02181	0.28779	0.43622	0.14296	0.18369
11	z_{200}	Agrupamiento	ISOMAP	Mediana	0.02881	0.02175	0.28808	0.43509	0.14287	0.18317
12	z_{200}	Agrupamiento	UMAP	Mediana	**0.02904**	**0.02188**	**0.29039**	0.43758	0.14354	0.18394
13	z_{200}	Agrupamiento	t-SNE	Mediana	0.02888	0.02179	0.28884	0.43579	0.1433	0.18362

Recordemos que elegimos considerar tres modelos locales para el entrenamiento de LOCA, dado que este número logra un balance adecuado entre desempeño y costo computacional (ver Figura 4.5).

Figura 5.9: Resultados de 13 experimentos propuestos. Ver tabla 5.3 para mayor detalle de la configuración de cada experimento

Capítulo 6

Conclusiones y trabajo futuro

Los experimentos realizados en esta tesis mostraron que la inclusión de información adicional a la matriz de interacción para la selección de anclas permite construir modelos locales más representativos y, en consecuencia, mejorar la calidad de las recomendaciones. En particular, se observó que seleccionar anclas de manera estratégica —a través de técnicas de agrupamiento y reducción de dimensionalidad sobre la matriz de características de usuarios $\mathbb{M}$— conduce a mejoras sustanciales en comparación con enfoques más simples como la selección aleatoria o por cobertura. Esto confirma la hipótesis inicial de que ubicar anclas en posiciones clave dentro del espacio de usuarios facilita una mejor cobertura de la heterogeneidad de las preferencias.

El modelo MultVAE sirvió como base para la construcción de embeddings latentes de los usuarios, los cuales demostraron ser efectivos para capturar patrones no lineales en las interacciones. Sin embargo, el verdadero valor añadido provino de la integración de estos embeddings con el enfoque local de LOCA, que permitió explotar la diversidad de subregiones del espacio latente mediante la combinación de modelos locales especializados. La evidencia experimental mostró que LOCA, al aprovechar anclas estratégicamente seleccionadas, supera al baseline en la mayoría de configuraciones evaluadas.

Otro hallazgo relevante es que el modelo LOCA muestra baja sensibilidad a parámetros como la dimensionalidad del embedding z, obteniéndose los mejores resultados con $z = 200$. Asimismo, se corroboró que no existe una diferencia significativa entre seleccionar anclas de manera aleatoria o por cobertura, lo cual refuerza la necesidad de utilizar criterios más informados basados en covariables o agrupamientos. Dentro de los métodos de reducción de dimensionalidad probados, UMAP y t-SNE destacaron por su capacidad para generar anclas más representativas, mientras que ISOMAP ofreció un desempeño más estable, aunque menos competitivo en métricas de precisión y NDCG.

En términos prácticos, uno de los principales retos enfrentados fue la transformación de los datos. Fue necesario mapear y unificar distintos conjuntos de información —los ratings de películas, las características de los ítems y los datos demográficos de los usuarios— en una representación común. Este proceso implicó no solo diseñar una estrategia adecuada para incorporar toda esa información de manera consistente, sino también resumirla para que resultara útil en la caracterización de grupos de usuarios y, en consecuencia, en la selección de anclas más informativas.

A ello se suma la limitación computacional: con los recursos disponibles no fue posible llevar a cabo experimentos en conjuntos de datos de más de un millón de observaciones, lo que restringió la evaluación del modelo en escenarios de mayor escala. Por ello, un trabajo futuro natural es la implementación de LOCA en entornos de cómputo distribuido, tales como la nube o clusters de alto rendimiento, para habilitar el procesamiento de volúmenes de datos más grandes y acercar el modelo a aplicaciones reales en sistemas de recomendación de gran escala.

Adicionalmente, se abren varias líneas de investigación para aprovechar mejor la flexibilidad del marco bayesiano utilizado. Una posibilidad interesante consiste en explorar distribuciones alternativas a las empleadas en esta tesis, con el fin de capturar mejor la variabilidad e incertidumbre en los datos de interacción. Este enfoque permitiría avanzar hacia modelos con un carácter menos computacional y más estadístico,

ampliando el potencial explicativo y predictivo de LOCA.

En conclusión, esta tesis demostró que la combinación de embeddings generativos (MultVAE) con un esquema local (LOCA) y una selección estratégica de anclas es un camino prometedor para mejorar la calidad de los sistemas de recomendación. Aunque aún existen retos computacionales y metodológicos por resolver, los resultados obtenidos sientan una base sólida para el desarrollo de futuras extensiones y aplicaciones de este modelo en escenarios de datos masivos y de alta heterogeneidad.

Referencias

Auto-encoder: What is it? and what is it used for? (part 1). (2023). Descargado Marzo 18, 2002, de https://www.tensorflow.org/datasets/catalog/mnist

Billsus, D., Pazzani, M. J., y cols. (1998). Learning collaborative information filters. En *Icml* (Vol. 98, pp. 46–54).

Bishop, C. M., y Bishop, H. (2024). Autoencoders. En C. M. Bishop y H. Bishop (Eds.), *Deep learning* (pp. 563–579). Cham: Springer. Descargado de https://doi.org/10.1007/978-3-031-45468-4_19 doi: 10.1007/978-3-031-45468-4_19

Bottou, L., Curtis, F. E., y Nocedal, J. (2018). Optimization methods for large-scale machine learning. *SIAM Review*, *60*(2), 223–311.

Choi, M., Jeong, Y., Lee, J., y Lee, J. (2021). Local collaborative autoencoders. En *Proceedings of the 14th acm international conference on web search and data mining* (pp. 734–742).

Christakopoulou, E., y Karypis, G. (2016). Local item-item models for top-n recommendation. En *Proceedings of the 10th acm conference on recommender systems* (pp. 67–74).

Christakopoulou, E., y Karypis, G. (2018). Local latent space models for top-n recommendation. En *Proceedings of the 24th acm sigkdd international conference on knowledge discovery & data mining* (pp. 1235–1243).

Dueck, D., Frey, B., Dueck, D., y Frey, B. J. (2004). Probabilistic sparse matrix factorization. *University of Toronto technical report PSI-2004-23*.

Evidently AI team. (2024). *Ndcg metric – ranking metrics for recommender systems.* https://www.evidentlyai.com/ranking-metrics/ndcg-metric. (Consultado el 6 de octubre de 2024)

GeeksforGeeks. (2024). *Umap - uniform manifold approximation and projection.* Descargado de https://www.geeksforgeeks.org/machine-learning/umap-uniform-manifold-approximation-and-projection/ (Accessed: 2025-08-13)

Gunawardana, A., y Shani, G. (2009). A survey of accuracy evaluation metrics of recommendation tasks. En *Journal of machine learning research* (Vol. 10, pp. 2935–2962).

Harper, F. M., y Konstan, J. A. (2015). The movielens datasets: History and context. *Acm transactions on interactive intelligent systems (tiis)*, *5*(4), 1–19.

Herlocker, J. L., Konstan, J. A., Terveen, L. G., y Riedl, J. T. (2004). Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS)*, *22*(1), 5–53.

Kingma, D. P., y Welling, M. (2014). *Auto-encoding variational bayes.*

Kingma, D. P., Welling, M., y cols. (2019). An introduction to variational autoencoders. *Foundations and Trends® in Machine Learning*, *12*(4), 307–392.

Koren, Y., Bell, R., y Volinsky, C. (2009a). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30-37. doi: 10.1109/MC.2009.263

Koren, Y., Bell, R., y Volinsky, C. (2009b, agosto). Matrix factorization techniques for recommender systems. *Computer*, *42*(8), 30–37. Descargado de http://dx.doi.org/10.1109/MC.2009.263 doi: 10.1109/mc.2009.263

Lang, K. (1995). Newsweeder: Learning to filter netnews. En *Machine learning proceedings 1995* (pp. 331–339). Elsevier.

Lee, J., Bengio, S., Kim, S., Lebanon, G., y Singer, Y. (2014). Local collaborative ranking. En *Proceedings of the 23rd international conference on world wide web* (pp. 85–96).

Lee, J., Kim, S., Lebanon, G., y Singer, Y. (2013). Local low-rank matrix approximation. En *International conference on machine learning* (pp. 82–90).

Lee, J., Kim, S., Lebanon, G., Singer, Y., y Bengio, S. (2016). Llorma: Local low-rank matrix approximation.

Lepping, J. (2018). Wiley interdisciplinary reviews: Data mining and knowledge discovery. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*.

Li, X., Cheung, M., y She, J. (2016). Connection discovery using shared images by gaussian relational topic model. En *2016 ieee international conference on big data (big data)* (pp. 931–936).

Liang, D., Krishnan, R. G., Hoffman, M. D., y Jebara, T. (2018). *Variational autoencoders for collaborative filtering.*

López, V. F. (2013). *Técnicas eficientes para la recomendación de productos basadas en filtrado colaborativo* (Tesis Doctoral no publicada). Universidade da Coruña.

Manning, C. D., Raghavan, P., y Schütze, H. (2008). *Introduction to information retrieval*. Cambridge university press.

McInnes, L., Healy, J., y Melville, J. (2018). Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*. Descargado de https://arxiv.org/abs/1802.03426

McQueen, J. (1967). Some methods for classification and analysis of multivariate observations. En *Proc. fifth berkeley symposium on mathematical statistics and probability, 1967* (pp. 281–297).

Mnih, A., y Gregor, K. (2014). Neural variational inference and learning in belief networks. En *Proceedings of the 31st international conference on machine learning.*

Mnih, A., y Salakhutdinov, R. R. (2007). Probabilistic matrix factorization. *Advances in neural information processing systems*, *20*.

Mnist. (2023). Descargado Marzo 18, 2002, de https://www.tensorflow.org/datasets/catalog/mnist

Nadaraya, E. A. (1964). On estimating regression. *Theory of Probability & Its Applications*, *9*(1), 141–142.

Pazzani, M., y Billsus, D. (1997). Learning and revising user profiles: The identifica-

tion of interesting web sites. *Machine learning*, *27*(3), 313–331.

Popkes, A.-L. (2019). *Kullback-leibler divergence.* Descargado 2024-04-25, de https://alpopkes.com/posts/machine_learning/kl_divergence/

Rendle, S. (2010). Factorization machines. En *2010 ieee international conference on data mining* (p. 995-1000). doi: 10.1109/ICDM.2010.127

Rennie, J. D., y Srebro, N. (2005). Fast maximum margin matrix factorization for collaborative prediction. En *Proceedings of the 22nd international conference on machine learning* (pp. 713–719).

Rezende, D. J., Mohamed, S., y Wierstra, D. (2014a). Stochastic backpropagation and approximate inference in deep generative models. En *Proceedings of the 31st international conference on machine learning.*

Rezende, D. J., Mohamed, S., y Wierstra, D. (2014b, 22–24 Jun). Stochastic backpropagation and approximate inference in deep generative models. En E. P. Xing y T. Jebara (Eds.), *Proceedings of the 31st international conference on machine learning* (Vol. 32, pp. 1278–1286). Bejing, China: PMLR. Descargado de https://proceedings.mlr.press/v32/rezende14.html

Tipping, M. E., y Bishop, C. M. (1999). Probabilistic principal component analysis. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, *61*(3), 611–622.

Van der Maaten, L., y Hinton, G. (2008). Visualizing data using t-sne. *Journal of machine learning research*, *9*(11).

Wang, C., y Blei, D. M. (2011). Collaborative topic modeling for recommending scientific articles. En *Proceedings of the 17th acm sigkdd international conference on knowledge discovery and data mining* (pp. 448–456).

Wang, K., Peng, H., Jin, Y., Sha, C., y Wang, X. (2016). Local weighted matrix factorization for top-n recommendation with implicit feedback. *Data Science and Engineering*, *1*, 252–264.

Watson, G. S. (1964). Smooth regression analysis. *Sankhyā: The Indian Journal of Statistics, Series A*, 359–372.

Welling, M. (2020). *Integrating generative modeling into deep learning.* Keynote

presented at. Descargado de https://vimeo.com/396903010 (22 February 2020, Valletta, Malta)

Xia, J., Zhang, Y., Song, J., Chen, Y., Wang, Y., y Liu, S. (2021). Revisiting dimensionality reduction techniques for visual cluster analysis: An empirical study. *IEEE Transactions on Visualization and Computer Graphics*, *28*(1), 529–539.

Yang, X. (2007). Understanding the variational lower bound. *Institute for Advanced Computer Studies*.

Apéndice A

Anexos

A.1. Capítulo 4.

A.1.1. Redes Neuronales

Las redes neuronales (RN), en general, son un tipo de modelo matemático y computacional inspirado en el funcionamiento del cerebro humano. También se les conoce como redes neuronales artificiales (RNA) debido a su naturaleza artificial y la similitud con las redes de neuronas biológicas.

Las redes neuronales consisten en unidades interconectadas llamadas neuronas artificiales o nodos, organizadas en capas, y se utilizan para resolver problemas de aprendizaje automático y reconocimiento de patrones. Una vertiente de las RN son las redes neuronales profundas (RNP), adoptan una arquitectura con varias capas ocultas entre las capas de entrada y salida, las cuales contienen capas de activación no lineales y, por lo tanto, son generalmente más potentes que los modelos tradicionales; sin embargo, los modelos de sistemas de recomendación que usan redes neuronales suelen emplear redes poco profundas debido a la escasez de datos de entrenamiento.

A.1.2. Técnicas de reducción de dimensión

Las técnicas de reducción de dimensionalidad son ampliamente utilizadas en agrupamiento para visualizar y analizar datos complejos en espacios de menor dimensión. Cada método tiene sus propias ventajas y enfoques, y la elección depende de las características de los datos y los objetivos del análisis. En particular, en este trabajo se consideran los métodos ISOMAP, UMAP y t-SNE.

ISOMAP (Isometric Mapping) es un algoritmo de reducción de dimensionalidad no lineal que busca preservar la geometría intrínseca de los datos al proyectarlos en un espacio de menor dimensión. Su fundamento radica en aproximar las distancias geodésicas sobre la variedad original mediante un grafo de vecinos más cercanos, y luego aplicar un escalado multidimensional clásico sobre la matriz de distancias resultante. De esta forma, ISOMAP captura relaciones globales en la estructura de los datos, lo que lo hace adecuado para representar manifolds no lineales en dimensiones reducidas (Lepping, 2018).

Por otro lado, *UMAP* (Uniform Manifold Approximation and Projection) es una técnica de reducción de dimensionalidad que combina principios de topología algebraica y teoría de grafos para modelar la estructura de los datos como un conjunto simplicial difuso. El algoritmo asume que los datos se encuentran distribuidos sobre una variedad riemanniana de baja dimensión y construye un grafo de proximidad en el espacio original, cuya estructura se preserva en la proyección mediante la optimización de una función de costo. En comparación con otros métodos, UMAP ofrece una mayor escalabilidad, conserva tanto la estructura local como global y resulta útil no solo para visualización, sino también como paso previo en tareas de aprendizaje automático (GeeksforGeeks, 2024; McInnes, Healy, y Melville, 2018).

Finalmente, *t-SNE* (t-Distributed Stochastic Neighbor Embedding) es un algoritmo diseñado principalmente para visualización de datos en dos o tres dimensiones. Su enfoque consiste en modelar las similitudes entre pares de puntos en el espacio

de alta dimensión como probabilidades, y luego encontrar una representación de baja dimensión en la que dichas probabilidades se conserven lo mejor posible. Una característica distintiva de t-SNE es su capacidad para resaltar la estructura local, revelando agrupamientos y subcomunidades en los datos, aunque a costa de perder información global y con un alto costo computacional (Van der Maaten y Hinton, 2008).

A.1.3. K-Medias

K-means es un método de agrupamiento no jerárquico que busca particionar un conjunto de datos en K grupos predefinidos. Cada grupo se representa por su centroide, que es el punto medio de todos los puntos pertenecientes a ese grupo (McQueen, 1967). Dado un conjunto de datos $X = \{x_1, x_2, ..., x_n\}$, donde cada punto x_i pertenece a un espacio vectorial de dimensión d, y el número de grupos K. Se define el conjunto de centroides $C = \{c_1, c_2, ..., c_K\}$, donde cada centroide c_k también es un vector de dimensión d, el objetivo es

$$\text{minimizar: } \sum_{i=1}^{n} \sum_{j=1}^{K} \|x_i - c_j\|^2 ,$$

donde n es el número total de puntos en el conjunto de datos K es el número de grupos predefinidos, x_i representa el punto i en el conjunto de datos y c_j representa el centroide del grupo j. La función $\|x_i - c_j\|^2$ representa la distancia euclidiana al cuadrado entre el punto x_i y el centroide c_j.

En otras palabras, el objetivo del algoritmo de K-means es encontrar los centroides $C = \{c_1, c_2, ..., c_K\}$ que minimicen la suma de las distancias al cuadrado mencionada anteriormente, asignando cada punto x_i al centroide c_j más cercano.

A.1.4. Agrupamiento Jerárquico

El agrupamiento jerárquico es un método que crea una estructura de agrupamiento en forma de árbol o dendrograma, donde los grupos se fusionan o dividen gradualmente según su similitud. Un principal enfoque en el agrupamiento jerárquico es el aglomerativo. El enfoque aglomerativo comienza con cada punto como un grupo individual y fusiona los grupos más similares en cada iteración, hasta formar un único grupo final que contiene todos los puntos. La similitud entre grupos se mide utilizando una métrica de distancia, como la distancia euclidiana. Dado un conjunto de datos $X = \{x_1, x_2, ..., x_n\}$, donde cada punto x_i pertenece a un espacio vectorial de dimensión d. Se define una matriz de distancias D de tamaño $n \times n$, donde cada elemento $D(s,t)$ representa la distancia entre los puntos c_s y c_t.

$$D = \text{dist}(s,t) = \|c_s - c_t\|_2$$

donde c_s y c_t son los centroides de los grupos s y t, respectivamente. Cuando dos grupos s y t se combinan en un nuevo grupo u, el nuevo centroide se calcula sobre todos los objetos originales en los grupos s y t. La distancia Entonces, la distancia se convierte en la distancia euclidiana entre el centroide de u y el centroide de un grupo restante v en el bosque. Esto también se conoce como el algoritmo UPGMC, para mayor detalle ver .

Resumiendo, el resultado del agrupamiento jerárquico es una estructura jerárquica en forma de árbol o dendrograma, que puede ser cortado en diferentes niveles para obtener diferentes particiones de los datos.

A.2. Tablas

Tabla A.1: Experimentación con diferentes embeddings (Emb) de usuarios.

Emb.	Emb. Epochs	Size	N. usuarios por Grupo			Resultados			
			1	2	3	Recall@50	*Recall@100*	NDCG@50	NDCG@100
Baseline	-	200	5970	5809	5834	0.2847	0.4303	0.1412	0.1811
z	100	100	6038	6031	5972	0.2884	**0.4366**	0.1427	0.1834
z	100	200	6040	2460	6036	**0.2894**	0.4363	**0.1438**	**0.1841**
z	100	600	5906	5975	5722	0.2856	0.4323	0.1416	0.1818
z	1000	100	5402	5068	5349	0.2854	0.4319	0.1413	0.1815
z	1000	200	4942	4752	4174	0.2854	0.4320	0.1411	0.1813
z	1000	600	2782	429	1801	0.2855	0.4319	0.1412	0.1813

Tabla A.2: Resultados del uso de 2 modelos locales con entrenamiento de embedding z de 100 épocas. Time representa el tiempo de entrenamiento del modelo con diferentes embeddings (Emb.) de los usuarios.

Emb: Baseline

Run	Grupos		Resultados				
	1	2	Recall@50	Recall@100	NDCG@50	NDCG@100	Time
1	5970	5809	0.2843	0.4304	0.1409	0.181	36.2
2	5970	5809	0.2844	0.43	0.1409	0.1808	**32**
3	5970	5809	**0.2849**	**0.4322**	0.1409	**0.1813**	46.6
4	5970	5809	0.2846	0.4304	**0.1411**	0.1811	52.9

Emb: z **Embedding size: 100**

Run	Grupos		Resultados				
	1	2	Recall@50	Recall@100	NDCG@50	NDCG@100	Time
1	6038	6031	0.2858	**0.4357**	0.1420	0.1832	284.9
2	6038	6031	**0.2878**	0.4341	**0.1432**	0.1834	**280**
3	6038	6031	0.2875	0.4349	0.1431	**0.1836**	334.2
4	6038	6031	0.2860	0.4347	0.1419	0.1827	309.4

Emb: z **Embedding size: 200**

Run	Grupos		Resultados				
	1	2	Recall@50	Recall@100	NDCG@50	NDCG@100	Time
1	6040	2460	**0.2894**	**0.4372**	**0.1435**	**0.1840**	441.1
2	6040	2460	0.2877	0.4341	0.1425	0.1827	386.3
3	6040	2460	0.2876	0.4359	0.1422	0.1828	**362**
4	6040	2460	0.2866	0.4338	0.1430	0.1834	414.2

Emb: z **Embedding size: 600**

Run	Grupos		Resultados				
	1	2	Recall@50	Recall@100	NDCG@50	NDCG@100	Time
1	5906	5975	0.2855	0.4320	0.1413	0.1815	38
2	5906	5975	0.2855	0.4323	0.1414	0.1816	49.5
3	5906	5975	**0.2860**	**0.4325**	**0.1415**	**0.1817**	47.1

Continuación de tabla A.2

4	5906	5975	0.2851	0.4321	0.1414	**0.1817**	**35.3**

Tabla A.3: Resultados del uso de 3 modelos locales con entrenamiento de embedding z de 100 épocas. Time representa el tiempo de entrenamiento del modelo con diferentes embeddings (Emb.) de los usuarios.

Emb: Baseline	Baseline							
Run	**Grupos**			**Resultados**				
	1	**2**	**3**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	5970	5809	5834	0.2844	0.4307	0.1412	0.1813	54
2	5970	5809	5834	0.2844	0.43	0.1409	0.1808	**32**
3	5970	5809	5834	**0.2856**	**0.4317**	**0.1415**	**0.1815**	78.1
4	5970	5809	5834	0.2846	0.4304	0.1411	0.1811	52.9
Emb: z	**Embedding size: 100**							
Run	**Grupos**			**Resultados**			**Resultados**	
	1	**2**	**3**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	6038	6031	5972	0.2861	0.4365	0.1419	0.1831	**279.8**
2	6038	6031	5972	0.2867	0.4357	0.1423	0.1832	338.9
3	6038	6031	5972	0.2881	0.4368	0.1430	0.1838	295.2
4	6038	6031	5972	**0.2912**	**0.4383**	**0.1440**	**0.1844**	587
Emb: z	**Embedding size: 200**							
Run	**Grupos**			**Resultados**	**Resultados**			
	1	**2**	**3**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	6040	2460	6036	0.2891	**0.4396**	0.1427	0.1839	**471.5**
2	6040	2460	6036	0.2904	0.4388	**0.1438**	**0.1845**	506.4
3	6040	2460	6036	**0.2895**	0.4366	0.1433	0.1836	480.2
4	6040	2460	6036	0.2847	0.4343	0.1436	0.1839	488
Emb: z	**Embedding size: 600**							
Run	**Grupos**			**Resultados**	**Resultados**			
	1	**2**	**3**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	5906	5975	5722	0.2852	**0.4327**	0.1413	0.1817	**44.3**
2	5906	5975	5722	0.2850	0.4321	**0.1414**	**0.1818**	46.7
3	5906	5975	5722	**0.2856**	0.4322	**0.1414**	0.1817	48.7
4	5906	5975	5722	0.2851	0.4324	0.1413	0.1817	52

Tabla A.4: Resultados del uso de 4 modelos locales con entrenamiento de embedding z de 100 épocas. Time representa el tiempo de entrenamiento del modelo con diferentes embeddings (Emb.) de los usuarios.

Emb: Baseline									
Run	**Grupos**				**Resultados**				
	1	**2**	**3**	**4**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	5970	5809	5834	5937	**0.2859**	0.4323	**0.1416**	0.1817	84.6
2	5970	5809	5834	5937	0.2851	0.4322	**0.1416**	0.1819	86.3
3	5970	5809	5834	5937	0.2847	0.4300	0.1413	0.1812	**58.6**
4	5970	5809	5834	5937	0.2856	**0.4327**	**0.1416**	**0.1820**	82.7
Emb: z	**Embedding size: 100**								
Run	**Grupos**				**Resultados**				
	1	**2**	**3**	**4**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	6038	6031	5972	6023	0.2876	0.4357	0.1428	0.1834	312.6
2	6038	6031	5972	6023	**0.2881**	0.4374	**0.1432**	**0.1842**	324.5
3	6038	6031	5972	6023	0.2864	0.4366	0.1423	0.1835	**311.5**
4	6038	6031	5972	6023	0.2873	**0.4375**	0.1428	0.1840	327.2
Emb: z	**Embedding size: 200**								
Run	**Grupos**				**Resultados**				
	1	**2**	**3**	**4**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	6040	2460	6036	6036	0.2904	**0.4390**	0.1412	**0.1850**	**569.4**
2	6040	2460	6036	6036	**0.2908**	0.4368	**0.1442**	0.1843	574.7
3	6040	2460	6036	6036	0.2907	0.4373	0.1438	0.1840	642.2
4	6040	2460	6036	6036	0.2850	0.4325	0.1413	0.1818	60.4
Emb: z	**Embedding size: 600**								
Run	**Grupos**				**Resultados**				
	1	**2**	**3**	**4**	**Recall@50**	**Recall@100**	**NDCG@50**	**NDCG@100**	**Time**
1	5906	5975	5722	6019	0.2856	**0.4327**	0.1415	0.1819	68.6
2	5906	5975	5722	6019	0.2854	0.4325	**0.1416**	0.1819	62.3
3	5906	5975	5722	6019	**0.2858**	**0.4327**	0.1415	0.1818	65.4
4	5906	5975	5722	6019	0.2847	0.4325	0.1413	0.1819	**58.7**