UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

Arquitectura para la Predicción de *Multi-criteria Ratings* en Sistemas de Recomendación

.

Juan Pedro Otavalo Maigua

Ingeniería en Ciencias de la Computación

Trabajo de fin de carrera presentado como requisito
para la obtención del título de
Ingeniero en Ciencias de la Computación

Quito, 14 de mayo de 2021

UNIVERSIDAD SAN FRANCISCO DE QUITO USFQ

Colegio de Ciencias e Ingenierías

HOJA DE CALIFICACIÓN
DE TRABAJO DE FIN DE CARRERA

Arquitectura para la Predicción de *Multicriteria Ratings* en Sistemas de Recomendación

Juan Pedro Otavalo Maigua

Nombre del profesor, Título académico **Daniel Riofrio, PhD**

Quito, 14 de mayo de 2021

© DERECHOS DE AUTOR

Nombres y apellidos: Juan Pedro Otavalo Maigua

Código: 00117882

Cédula de identidad: 1003982988

Lugar y fecha: Quito, 14 de mayo de 2021

ACLARACIÓN PARA PUBLICACIÓN

Nota: El presente trabajo, en su totalidad o cualquiera de sus partes, no debe ser considerado como una publicación, incluso a pesar de estar disponible sin restricciones a través de un repositorio institucional. Esta declaración se alinea con las prácticas y recomendaciones presentadas por el Committee on Publication Ethics COPE descritas por Barbour et al. (2017) Discussion document on best practice for issues around theses publishing, disponible en http://bit.ly/COPETheses.

UNPUBLISHED DOCUMENT

Note: The following capstone project is available through Universidad San Francisco de Quito USFQ institutional repository. Nonetheless, this project – in whole or in part – should not be considered a publication. This statement follows the recommendations presented by the Committee on Publication Ethics COPE described by Barbour et al. (2017) Discussion document on best practice for issues around theses publishing available on http://bit.ly/COPETheses.

RESUMEN

La correcta predicción de los *multi-criteria rating* en los sistemas de recomendación ayuda a mejores predicciones de *overall rating* lo cual produce un mejor sistema de recomendación. La arquitectura propuesta en este texto ayuda a predecir los *multi-criteria rating* mediante tres procesos: *encoding data*, preprocesamiento de datos y predicción de *multicriteria rating* usando *multi-layer perceptron*. *Encoding data* ayuda a transformar los datos categóricos (usuarios e ítems) a datos numéricos. El preprocesamiento de datos mediante *autoencoder* elimina características innecesarias del anterior proceso. Se usa *multi-layer perceptron* para predecir los *multi-criteria rating*. Esta arquitectura ha sido probada en un *data-set* que contiene 1000 usuarios demostrando tener buenos resultados en la predicción de los *multi-criteria rating*.

Palabras clave: sistema de recomendación, *multi-criteria rating, autoencoder, multi-layer perceptron, one hot encoding.*

ABSTRACT

The correct prediction of the multi-criteria rating in recommendation systems helps to obtain better predictions of the overall rating, which produces a better recommendation system. The architecture proposed in this text helps to predict multi-criteria rating through three processes: encoding, data pre-processing, and multi-criteria classification prediction using multi-layer perceptron. Encoding helps to transform categorical data (users and items) into numerical data. Data pre-processing by autoencoder delete unnecessary features from the previous process. Multi-layer perceptron is used to predict multi-criteria rating. This architecture has been tested don a data set containing 1000 users showing good results in the prediction of multi-criteria rating.

Key words: recommendation system, multi-criteria rating, autoencoder, multi-layer perceptron, one hot encoding.

TABLA DE CONTENIDO

Introducción **10**

Desarrollo del Tema **12**

Estado del Arte **12**

Materiales y Métodos **14**

Base de datos Experimental 14

Modelos 15

Propuesta experimental 16

Entrenamiento y pruebas 19

Discusión y resultados 23

Conclusiones **24**

Referencias bibliográficas **25**

ÍNDICE DE TABLAS

Tabla 1.- Datos estadísticos de movie data-set 14
Tabla 2.- Datos multi-criteria rating de movie data-set 14
Tabla 3.- Datos estadísticos de 1000 usuarios de movie data-set 15
Tabla 4.- Datos multi-criteria rating de 1000 usuarios de movie data-set 15
Tabla 5.- Dimensión del vector one hot encoding concatenado 17
Tabla 6.- Modelos de autoencoder para el bottleneck de 10% 20
Tabla 7.- Modelos de multi-layer perceptron para el bottleneck de 10%..... 20
Tabla 8.- Modelos de autoencoder para el bottleneck de 25% 21
Tabla 9.- Modelos de multilayer perceptron para el bottleneck de 25%..... 21
Tabla 10.- Modelos de autoencoder para el bottleneck de 50% 22
Tabla 11.- Modelos de multilayer perceptron para el bottleneck de 50%..... 22
Tabla 12.- Muestra de los mejores resultados con la arquitectura propuesta 23

ÍNDICE DE FIGURAS

Figura 1.- Matriz de Factorización Multi-criteria Rating 11
Figura 2.- Arquitectura presentada propuesta 16
Figura 3.- Matriz de factorización con one hot encoding 17
Figura 4.- Arquitectura autoencoder 18

INTRODUCCIÓN

Los sistemas de recomendación ayudan a que los usuarios de empresas que ofrecen productos o servicios como *Netflix, Google, Amazon* o *Spotify* puedan recibir recomendaciones de los productos/ítems que ofrecen. Según Nassar, Jafar y Rahhal (2019) existen 3 tipos de técnicas para la creación de sistemas de recomendación: Filtración Colaborativa (*Colaborative Filtering*), Filtro Basado en contenido (*Content Based Filtering*) y Filtro Basado en las dos anteriores (*Hibrid Based Filtering*). *Collaboraive Filtering* recomienda ítems que a usuarios con similares preferencias les ha gustado en el pasado mientras que *Content Based Filtering* recomienda ítems similares a los que le gustó a usuarios en el pasado y finalmente *Hybrid Based Filtering* combina los métodos de *Content Based Filtering* y *Collaborative Filtering* (Adomavicius & Kwon, 2007)

La mayoría de los sistemas de recomendación, sin importar su tipo, están basados en sistemas de calificación de ítems. A esta calificación de ítem se lo conoce como evaluación general (*overall rating*). Cada ítem tiene un *overall rating* que es dado por un usuario y que refleja el gusto o disgusto por el ítem. Mientras mayor sea el *overall rating* que tenga un ítem, mayor será la probabilidad de que ese ítem sea recomendado al usuario debido a que su gusto por el ítem es alto. Sin embargo, muchas veces este *rating* no refleja los factores por los cuales a un usuario le gusta el ítem, por ejemplo: a un usuario le gusta una canción por el ritmo de los tambores, la tonalidad de la guitarra y el sonido del bajo.

Todos estos criterios son conocidos como *multi-criteria ratings* y no eximen la existencia del *overall rating* para cada ítem. Por el contrario, según Adomavicius y Kwon (2007) los *overall ratings* son dependientes de los *multi-criteria ratings.* De la misma forma Nassar, Jafar y Rahhal (2020) dicen que los sistemas de recomendación que utilizan *multi-*

criteria ratings son más precisos porque brindan a los usuarios la posibilidad de dar múltiples *ratings* para un ítem, lo que significa una mejor representación de la preferencia hacia un ítem.

Los datos de los ítems, usuarios y *ratings* se los suelen representar o almacenar en una matriz de factorización como se muestra en la Figura 1 donde se muestra en números grandes los *overall ratings* y en números más pequeños los *multi-criteria ratings* (también conocidos como vectores latentes que pueden tener *ratings* en ítems y en usuarios). También se puede observar que existen campos de matriz de la figura 1 que contienen el signo de interrogación. Estos datos con signo de interrogación son los ítems a los cuales los usuarios aún no le han dado un *rating.* Jian, Chen, Zhou y Zouyin (2017) denominan a este problema de tener datos vacíos o desconocidos en la matriz de factorización como *cold-start problem.*

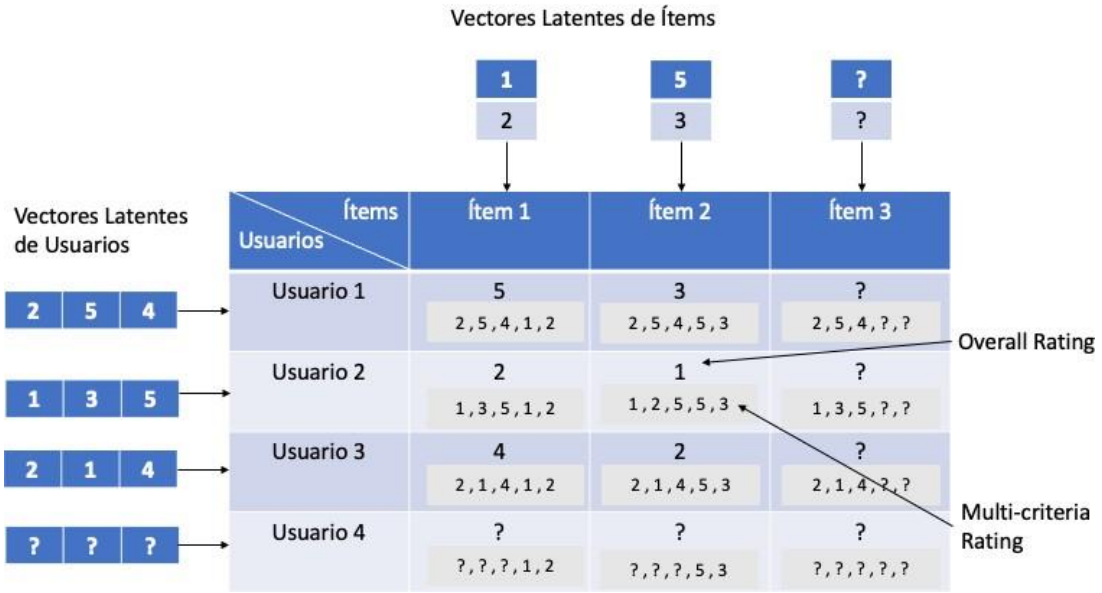

Figura 1.- Matriz de Factorización Multi-criteria Rating

En este texto se presenta una arquitectura que soluciona este problema mediante el uso de *Collaborative Filtering model based* junto con *multi-criteria ratings.* Enfocado únicamente en la predicción de los ratings desconocidos de *multi-criteria rating.* Esta arquitectura presenta las siguientes contribuciones:

1. Uso de *One Hot Encoding* para representar datos categóricos como los nombres de los usuarios e ítems.
2. Preprocesamiento del conjunto de datos (*data-set*) realizando reducción de dimensionalidad (*dimensionality reduction*) de los vectores *One Hot Encoding* mediante el uso de la red neuronal: *autoencoder*.
3. Comprobar la factibilidad de los dos procesos antes mencionados para la predicción de los *multi-criteria rating.*

Este texto está distribuido de la siguiente manera: en la sección de desarrollo del tema se muestra el estado del arte, en materiales y métodos se describe el *data-set* usado para probar la arquitectura que se está proponiendo junto con la descripción de las etapas que constituyen la arquitectura. Finalmente, se tiene la sección de los resultados junto con las conclusiones y el trabajo futuro.

DESARROLLO DEL TEMA

Estado del Arte

El estado del arte divide a Collaborative Filtering en dos enfoques: Basado en la Vecindad (*neighborhood-based*) y Basado en el Modelo (*model-based*). *Neighborhood-based* se enfoca en encontrar la similaridad entre los usuarios basado en los ratings que se hayan dado en el pasado. Esta similaridad se la puede medir según Adomavicius y Kwon (2007) por medio del cálculo de distancias como el *Manhattan Distance*, *Euclidean Distance* u otros. *Model-based* está enfocado en el entrenamiento de un modelo que permita predecir los *ratings* para nuevos usuarios.

En los últimos años se han presentado varias arquitecturas de *Machine Learning* y *Deep Learning* que aplican *collaborative-filtering* con *multi-criteria rating* y son *model-based*. Nassar, Jafar y Rahhal (2020) han presentado dos arquitecturas: *Deep multi-criteria*

collaborative filtering y *Multi-criteria collaborative filtering recommender by fusing deep neural network and matrix factorization* que según los autores son los primeros intentos de usar *Deep Learning* en sistemas de recomendación que usan *multi-criteria rating.* En los dos modelos anteriormente mencionados se usan *Multi-layer Perceptron* y *Deep Learning* para entrenar modelos que predigan los *multi-criteria ratings.*

Mientras que Bobadilla, Alonso y Hernando han presentado una arquitectura que usa *Deep Learning* no solo para predecir *multi-criteria ratings* sino también para conocer la confiabilidad de las predicciones obtenidas. Esto debido a que según Bobadilla, Alonso y Hernando (2020) las predicciones de los ratings pueden ser más confiables si el porcentaje de error en la predicción es muy bajo ayudando a seleccionar de mejor manera el ítem que se debe recomendar a un usuario.

Li y Tuzhlin (2019) plantean en su arquitectura de sistema de recomendación la toma de los *reviews* como un elemento fundamental para predecir el ítem que se debe recomendar. Para esto utilizan la arquitectura presentada por Sutskever y Cho de forma conjunta. Según Sutskever, Vinyals y Le (2014) se puede usar una red neuronal de tipo *autoencoder* para tranducir una palabra de Ingles a Frances, en la arquitectura de Li y Tuzhlin se utiliza este concepto para realizar un análisis semántico de las *reviews*. Para la representación de las palabras se ha usado la arquitectura presentada por Cho, Merriender, Gulcehre, et al (2014) en la que los autores plantean la transformación de datos categóricos a numéricos mediante el uso de la red neuronal del tipo Recurrent Neural Network (RNN). En la arquitectura planteada por Li y Tuzhlin se basa la recomendación en los *multi-criteria rating* y no en el *overall rating* como normalmente funcionan los sistemas de recomendación.

En el estado del arte se ha encontrado que una de las arquitecturas que ha servido de inspiración para diversos autores que aplican *collaborative filtering* por medio de *multi-layer perceptron* ha sido *Neural Collaborative Filtering.* He, Liao, Zhang, Nie y Hu (2017) dicen

que esta arquitectura en realidad es un *framework*. Ellos plantean en su *paper* una arquitectura que toma como entrada (*input*) vectores con valores de unos y ceros. Estas son las representaciones de los usuarios y los ítems que sirven como *input* para un *multi-layer perceptron* que predice un solo *rating* cuyo valor es binario.

Materiales y Métodos

Base de datos Experimental

El *data-set* con el que se probó la arquitectura presentada es el siguiente: *Movie Data-set.* Este conjunto de datos contiene 4 *multi-criteria ratings* y un *overall rating*. Tanto el *overall rating* como el *multi-criteria rating* tiene valores en el rango del 1 al 13. En la tabla 1 se puede observar los datos estadísticos del *data-set* y en la tabla 2 se muestra la cantidad de *ratings* de valor 1 que recibió el *multi-criteria rating* con nombre Criterio 1. Estos datos se muestran para todos los *ratings* del 1 al 12 y para todos los *multi-criteria.*

Descripción Campo	Valor
Número usuarios	6078
Número películas	976
Número Ratings	62.156
Dispersión de datos	98.95%

Tabla 1.- Datos estadísticos de movie data-set

Ratings	1	2	3	4	5	6	7	8	9	10	11	12	13
Criterio 1	3325	1046	1581	1151	1754	3018	2297	3462	6544	6582	5711	12187	12498
Criterio 2	2567	796	1133	861	1375	2729	2038	3475	7360	6743	6163	12330	14586
Criterio 3	3300	1067	1619	1113	1672	3073	2181	3477	7413	6328	5781	12345	12787
Criterio 4	2533	655	1063	758	1163	2654	1802	3135	7156	6365	5794	12004	17074
General (*overall*)	3395	1340	1522	1329	2051	2428	2428	3251	5586	7006	6702	12153	12904

Tabla 2.- Datos multi-criteria rating de movie data-set

Debido a problemas de paralelización en GPU para probar los distintos modelos de *autoencoders* lo cual trajo consigo problemas en el tiempo de ejecución de la arquitectura se tomó la decisión de usar la información de 1000 usuarios de todo el *data-set* descrito para

probar el modelo presentado. A continuación, se muestra los datos estadísticos y los datos de *multi-criteria rating* del *data-set* para 1000 usuarios.

Descripción campo	Valor
Número usuarios	1000
Número películas	968
Número Ratings	100085

Tabla 3.- Datos estadísticos de 1000 usuarios de movie data-set

Ratings	1	2	3	4	5	6	7	8	9	10	11	12	13
Criterio 1	569	129	241	160	284	459	354	521	1014	963	979	2019	2393
Criterio 2	411	113	169	109	214	418	314	545	1109	1084	1032	1985	2582
Criterio 3	533	148	243	138	227	488	319	519	1134	1012	976	2087	2270
Criterio 4	434	93	152	99	171	411	230	460	1104	98	945	2005	3002
General (*overall*)	566	192	209	198	308	374	375	492	865	1096	1120	2048	2242

Tabla 4.- Datos multi-criteria rating de 1000 usuarios de movie data-set

Modelos

El modelo de arquitectura presentado se lo puede observar en la figura 2. En este gráfico se muestran tres procesos que forman parte de la arquitectura:

1. **Codificador (*Encoder*):** Se identifica con el color plomo y está enfocado en transformar los datos categóricos como el usuario id y el ítem id en datos numéricos. Esto es debido a que los nombres de los usuarios y de los ítems son categóricos en el *data-set* que se está probando el modelo.
2. **Preprocesamiento de datos (Data pre processing):** Se identifica con el color tomate y está enfocado en reducir la dimensionalidad de los datos numéricos del primer proceso y normalizarlos.
3. **Predicción de *multi-criteria rating* (*Multi-criteria rating prediction*):** Se identifica con el color verde y está enfocado en predecir los *multi-criteria ratings* tomando el resultado obtenido en el segundo proceso.

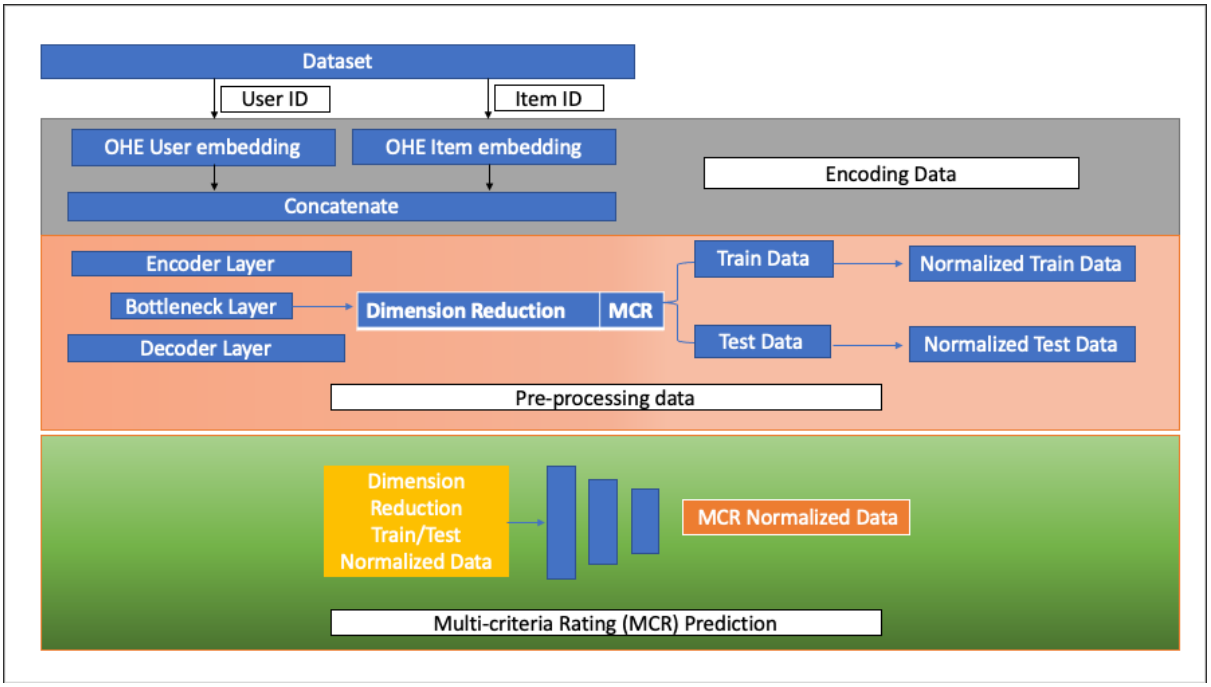

Figura 2.- Arquitectura presentada propuesta

Propuesta experimental

En esta sección se va a describir a mayor detalle los algoritmos usados para cada uno de los procesos antes mencionados en la sección de modelos.

Encoder: Dentro de la arquitectura los datos más importantes son el usuario id y el ítem id. Estos datos suelen ser de tipo categóricos. Debido a que estos datos son categóricos no pueden ser usados como datos de entrada para la red neuronal de tipo *autoencoder*. Según Potdar, Pardawala y Pai (2017), *One hot encoding* es la técnica de *encoding* más usada debido a que transforma las variables categóricas en observaciones que indican la presencia (1) o ausencia (0) de una variable categórica. Además de su uso en tareas como la calificación en los que Potdar, Pardawala y Pai demuestran que se obtiene buenos resultados.

Este es el enfoque que se tomó para transformar las variables categóricas que se tenían en el *data-set*: usuario id e ítem id. El primer proceso que se realizó fue transformar la columna de usuario id en vectores de *one hot encoding*. Lo mismo se realizó con la columna de ítem id. Cada uno de estos vectores (usuario id e ítem id) de *one hot encoding* se concatenaron

respectivamente. Formando un solo vector de *one hot encoding* que represente al usuario y al ítem, como se muestra en la figura 3. Este vector de *one hot encoding* es único para cada usuario e ítem.

Ítems / Usuarios	Ítem 1	Ítem 2	Ítem 3
Usuario 1	0,0,1,0,1,0...	1,0,1,0,1,1...	0,0,1,1,1,1...
Usuario 2	1,0,1,0,1,0...	0,0,1,1,1,0...	0,1,1,0,1,1...
Usuario 3	0,0,1,0,1,1...	0,0,1,0,1,0...	0,1,1,1,1,1...
Usuario 4	0,0,0,1,1,0...	1,0,0,0,1,0...	0,0,0,0,1,0...

Figura 3.- Matriz de factorización con one hot encoding

Data preprocessing: Los vectores concatenados del proceso anterior se toman como *input* para la red neuronal de tipo *autoencoder*. Las dimensiones del vector concatenado pueden variar dependiendo de la cantidad de usuarios. En la Tabla 7 se muestra la dimensión del vector *one hot encoding* concatenado.

Usuarios	**Dimensión**
1000	1882

Tabla 5.- Dimensión del vector one hot encoding concatenado

Como se puede observar en la tabla 7 la dimensión de *one hot encoding* de los vectores concatenados va en aumento cuando el número de usuarios aumenta. Según Miao y Niu (2016) el aumento de la dimensión puede generar demasiado uso de memoria, la presencia de datos que generan ruido y datos redundantes. Además, Verleysen, Francois, Simon y Wertz (2003) han demostrado que la alta dimensionalidad de los datos en redes neuronales puede afectar en el rendimiento de la red neuronal que se usa para clasificar imágenes.

Debido a estos problemas se planteó una reducción de dimensionalidad a los vectores *one hot encoding* que fueron concatenados en el proceso anterior. Para esto se usó una red neuronal de tipo *autoencoder* que tiene dos capas ocultas (*hidden layers*) en el encoder y dos capas ocultas (*hidden layers*) en el decoder. El *bottleneck* es la capa donde se obtiene el vector de *one hot encoding* con dimensionalidad reducida. La arquitectura del *autoencoder* se muestra en la figura 4.

Estos vectores con dimensión reducida fueron unidos junto con sus respectivos 4 *multi-criteria rating*. Debido a que cada vector representa a un usuario y un ítem que tiene *multi-criteria rating*. La unión de estos datos hizo que el rango de valores presentes entre cada columna fuera distinto debido a que en la tarea de reducción de dimensionalidad los valores de cada vector variaban demasiado entre las columnas. Para solucionar este problema se realizó la normalización de los datos después de separar el conjunto de datos, en este caso vector de dimensión reducida y *multi-criteria rating*, en conjunto de entrenamiento (*train data*) y conjunto de pruebas (*test data*).

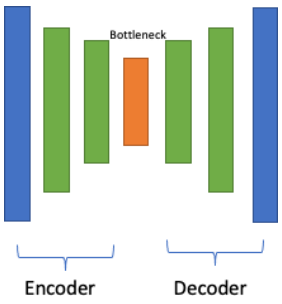

Figura 4.- Arquitectura autoencoder

Se realizó una normalización MinMax que según Singh y Singh (2020) ayuda a tener un límite superior e inferior en el conjunto de datos. Para esto lo que se hizo fue re escalar los valores del conjunto de datos, en este caso vector de dimensión reducida y multi-criteria rating, entre 0 y 1.

Predicción de multi-criteria rating: el conjunto de entrenamiento y prueba normalizado del proceso anterior fue usado en este proceso como *input/output* para el entrenamiento de la red neural *multi-layer perceptron.* Esta red neuronal fue la encargada de predecir los *multi-criteria rating* teniendo como input los vectores *one hot encoding* que se redujeron la dimensionalidad y fueron normalizados en el proceso anterior.

Para la implementación de la arquitectura se utilizaron las siguientes librerías. Se usó keras con tensorflow 2.0 como *backend.* Para transformar los datos categóricos en datos numéricos se usó sklearn con *LabelBinazer*. Así mismo para normalizar los datos se utilizó sklearn con MinMaxScaler realizando el ajuste (*fit*) del modelo con el conjunto de entrenamiento y usando este ajuste para normalizar el conjunto de prueba.

Entrenamiento y pruebas

Para entrenar el *multi-layer perceptron* que predice los *multi-criteria ratings* se generaron varios modelos de *autoencoders* que tienen 10%, 25% y 50% de neuronas de la capa de entrada en la capa del *bottleneck.* Se realizaron variaciones en la cantidad de neuronas en los *hidden layers* del *encoder* y *decoder.* Para esto se usaron porcentajes de cantidad de neuronas en cada capa del *autoencoder*. Este porcentaje se obtiene por medio de la cantidad de neuronas que entrada que sean necesarios. A continuación, se muestran los distintos modelos de redes neuronales que se realizaron tanto para el *autoencoder* como para el *multi-layer perceptron.* Estos son los modelos (*autoencoder* y *multi-layer perceptron*) cuando la capa del *bottleneck* tiene un 10% de neuronas de las neuronas de la capa de entrada.

		Enc. Capa 1	Enc. Capa 2	Botteneck	Dec. Capa 1	Dec. Capa 2
Modelo 1	%	75	50	10	50	75
	funcion ac.	tanh	sigmoid	selu	sigmoid	tanh
Modelo 2	%	55	25	10	25	55
	funcion ac.	sigmoid	relu	relu	sigmoid	softmax
Modelo 3	%	50	25	10	25	50
	funcion ac.	sigmoid	relu	sigmoid	sifmoid	softmax
Modelo 4	%	25	15	10	15	25
	funcion ac.	sigmoid	selu	relu	selu	relu
Modelo 5	%	35	15	10	15	35
	funcion ac.	sigmoid	selu	relu	selu	sigmoid

Tabla 6.- Modelos de autoencoder para el bottleneck de 10%

		Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Optimizador	Loss Funct.
Modelo 1	Nro. Neuronas	128	64	32	16	8	4	Adam (lr=0.01)	mae
	funcion acti.	softmax	relu	sigmoid	relu	relu	sigmoid		
Modelo 2	Nro. Neuronas	164	82	41	21	11	4	Adam (lr=0.01)	mae
	funcion acti.	softmax	selu	softmax	relu	selu	sigmoid		
Modelo 3	Nro. Neuronas	154	77	36	18	9	4	Adam (lr=0.1)	mse
	funcion acti.	relu	softmax	relu	softmax	relu	sigmoid		
Modelo 4	Nro. Neuronas	190	95	47	23	12	9	Adam (lr=0.01)	mse
	funcion acti.	relu	relu	relu	relu	relu	relu		
Modelo 5	Nro. Neuronas	124	72	31	16	8	4	Adam (lr=0.01)	mse
	funcion acti.	sigmoid	relu	sigmoid	relu	sigmoid	sigmoid		

Tabla 7.- Modelos de multi-layer perceptron para el bottleneck de 10%

Estos son los modelos (*autoencoder* y *multi-layer perceptron*) cuando la capa del *bottleneck* tiene un 25% de neuronas de las neuronas de la capa de entrada.

		Enc. Capa 1	Enc. Capa 2	Bottleneck	Dec. Capa 1	Dec. Capa 2
Modelo 1	%	75	50	25	50	75
	función acti.	tanh	relu	tanh	tanh	relu
Modelo 2	%	75	50	25	50	75
	función acti.	selu	relu	sigmoid	tanh	selu
Modelo 3	%	75	50	25	50	75
	función acti.	relu	sigmoid	selu	tanh	relu
Modelo 4	%	75	50	25	50	75
	función acti.	selu	relu	tanh	tanh	tanh
Modelo 5	%	75	50	25	50	75
	función acti.	relu	sigmoid	tanh	sigmoid	relu

Tabla 8.- Modelos de autoencoder para el bottleneck de 25%

		Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8	Capa 9
Modelo 1	Nro. Neu.	124	94	74	56	32	16	4	-	-
	fun. Act.	sigmoid	relu	sigmoid	relu	sigmoid	relu	sigmoid	-	-
Modelo 2	Nro. Neu.	308	204	104	90	70	50	30	12	4
	fun. Act.	softmax	relu	softmax	relu	softmax	relu	softmax	relu	sigmoid
Modelo 3	Nro. Neu.	374	257	127	76	48	19	4	-	-
	fun. Act.	relu	relu	relu	relu	relu	relu	sigmoid	-	-
Modelo 4	Nro. Neu.	290	195	57	33	16	9	4	-	-
	fun. Act.	relu	softmax	relu	softmax	relu	softmax	sigmoid	-	-
Modelo 5	Nro. Neu.	124	72	31	16	8	4	-	-	-
	fun. Act.	selu	relu	sigmoid	relu	selu	sigmoid	-	-	-

Tabla 9.- Modelos de multilayer perceptron para el bottleneck de 25%

Estos son los modelos (*autoencoder* y *multi-layer perceptron*) cuando la capa del *bottleneck* tiene un 50% de neuronas de las neuronas de la capa de entrada.

		Enc. Capa 1	Enc. Capa 2	Bottleneck	Dec. Capa 1	Dec. Capa 2
Modelo 1	%	87	75	50	75	87
	función acti.	selu	relu	relu	tanh	relu
Modelo 2	%	87	75	50	75	87
	función acti.	relu	selu	sigmoid	tanh	relu
Modelo 3	%	87	75	50	75	87
	función acti.	relu	softmax	selu	tanh	softmax
Modelo 4	%	87	75	50	75	87
	función acti.	selu	relu	tanh	tanh	tanh
Modelo 5	%	87	75	50	75	87
	función acti.	softmax	sigmoid	selu	sigmoid	selu

Tabla 10.- Modelos de autoencoder para el bottleneck de 50%

		Capa 1	Capa 2	Capa 3	Capa 4	Capa 5	Capa 6	Capa 7	Capa 8	Capa 9
Modelo 1	Nro. Neu.	624	456	394	256	132	96	42	18	4
	fun. Act.	softmax	relu	relu	relu	relu	relu	relu	relu	sigmoid
Modelo 2	Nro. Neu.	578	434	354	292	176	92	56	26	4
	fun. Act.	softmax	selu	softmax	softmax	relu	selu	relu	selu	sigmoid
Modelo 3	Nro. Neu.	454	377	237	186	98	59	37	19	4
	fun. Act.	relu	sigmoid	relu	sigmoid	relu	sigmoid	relu	relu	sigmoid
Modelo 4	Nro. Neu.	590	495	347	223	112	72	52	19	4
	fun. Act.	softmax	relu	softmax	relu	softmax	relu	softmax	relu	sigmoid
Modelo 5	Nro. Neu.	124	72	31	16	8	4	-	-	-
	fun. Act.	sigmoid	relu	sigmoid	relu	sigmoid	sigmoid	-	-	-

Tabla 11.- Modelos de multilayer perceptron para el bottleneck de 50%

Las tablas 9 y 11 tienen los mismos optimizadores y *loss functions* de la tabla 7 en el mismo orden.

Discusión y resultados

Con los modelos de arquitectura presentados en la sección de entrenamiento y pruebas se probaron distintas combinaciones entre los modelos de *autoencoders* y los modelos de *multi-layer perceptron.* En la tabla 12 se muestra la combinación de modelos de *autoencoder y multi-layer perceptron* con los cuales se obtuvieron los mejores resultados de *accuracy* y *Area Under Curve (AUC).* En esta tabla también se muestran 3 tipos de arquitectura que varían de acuerdo con el porcentaje de neuronas que se tuvo en la capa del *bottleneck* en el *autoencoder*.

Los resultados que se muestran en la tabla 12 muestran que la arquitectura con mejores resultados fue la arquitectura 2. El *accuracy* de esta arquitectura es del 97% esto quiere decir que el modelo ha acertado en la predicción de casi todos los *multi-criteria rating* y según el *AUC* la arquitectura tiene una buena capacidad de generalización en nuevos datos. Estos resultados también ayudan a entender por que las arquitecturas 1 y 3 pueden tener resultados inferiores. La arquitectura 1 estaría eliminando características importantes en el proceso de reducción de dimensionalidad del vector *one hot encoding* concatenado. Por otro lado, en la arquitectura 3 se estaría tomando en características que generan ruido o no realizan ningún aporte en el proceso de reducción de dimensionalidad.

		Modelos			
	% Neuronas en Bottleneck	**Autoencoder**	**Multilayer Perceptron**	**Accuracy (%)**	**AUC (%)**
Arquitectura 1	10	4	4	44	94
Arquitectura 2	25	1	1	97	99
Arquitectura 3	50	4	4	21	91

Tabla 12.- Muestra de los mejores resultados con la arquitectura propuesta

CONCLUSIONES

La arquitectura propuesta ha demostrado ser factible para la predicción de *multi-criteria rating.* Sin embargo, la obtención de buenos resultados (*accuracy* y *AUC*) depende del porcentaje de número de neuronas que se tenga en la capa del *bottleneck.* En la tabla 12 se puede observar cómo los resultados pueden variar drásticamente de acuerdo con el porcentaje de neuronas del *bottleneck.* Haciendo que una reducción de dimensionalidad agresiva (10%) o mediana (50%) no sea factible para predecir los *multi-criteria rating* correctamente.

Esta arquitectura solo se probó con 1000 usuarios. Para futuros trabajos se tendría que comprobar como es el funcionamiento de la arquitectura con el crecimiento de usuarios de forma escalonada. También se recomendaría aumentar una red neuronal que permita predecir el *overall rating* en base a los *multi-criteria rating* que ya predice la arquitectura propuesta.

REFERENCIAS BIBLIOGRÁFICAS

Nassar, N., Jafar, A., & Rahhal, Y. (2020). A novel deep multi-criteria collaborative filtering model for recommendation system. *Knowledge-Based Systems*, *187*, 104811. https://doi.org/10.1016/j.knosys.2019.06.019

Adomavicius, G., & Kwon, Y. (2007). New Recommendation Techniques for Multicriteria Rating Systems. *IEEE Intelligent Systems*, *22*(3), 48–55. https://doi.org/10.1109/mis.2007.58

Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items. *Expert Systems with Applications*, *69*, 29–39. https://doi.org/10.1016/j.eswa.2016.09.040

Nassar, N., Jafar, A., & Rahhal, Y. (2020b). Multi-criteria collaborative filtering recommender by fusing deep neural network and matrix factorization. *Journal of Big Data*, *7*(1). https://doi.org/10.1186/s40537-020-00309-6

Li, P., & Tuzhilin, A. (2019). Latent multi-criteria ratings for recommendations. *Proceedings of the 13th ACM Conference on Recommender Systems*. Published. https://doi.org/10.1145/3298689.3347068

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Published. https://doi.org/10.3115/v1/d14-1179

He, X., Liao, L., Zhang, H., Nie, L., Hu, X., & Chua, T. S. (2017). Neural Collaborative Filtering. *Proceedings of the 26th International Conference on World Wide Web*. Published. https://doi.org/10.1145/3038912.3052569

Miao, J., & Niu, L. (2016). A Survey on Feature Selection. *Procedia Computer Science*, *91*, 919–926. https://doi.org/10.1016/j.procs.2016.07.111

Bobadilla, J., Alonso, S., & Hernando, A. (2020). Deep Learning Architecture for Collaborative Filtering Recommender Systems. *Applied Sciences*, *10*(7), 2441. https://doi.org/10.3390/app10072441

Sutskever, I., Vinyals, O., & Le, Ve. (2014). *Sequense to sequense learning with neural networks.*

Potdar, K., Pardawala, T., & Pai, C. (2017). A Comparative Study of Categorical Variable Encoding Techniques for Neural Network Classifiers. *International Journal of Computer Applications*, *175*(4), 7–9. https://doi.org/10.5120/ijca2017915495

Verleysen, M., Francois, D., Simon, G., Wertz, V. (2003). *On the effects of dimensionality on data with neural networks.* Artifitial Neural Nets Problem Solving Methods. Extraido de: https://link.springer.com/chapter/10.1007%2F3-540-44869-1_14

Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. *Applied Soft Computing*, *97*, 105524. https://doi.org/10.1016/j.asoc.2019.105524