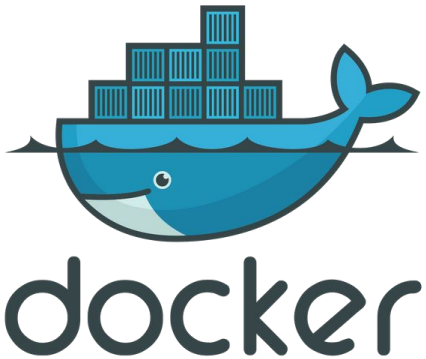
Uso de LIBMF (A LIBrary for large-scale sparse Matrix Factorization) para sistemas de recomendación de películas con una base de datos de usuarios de Netflix

Integrantes del Equipo:

- Óscar Aguilar
- Alfie González
- Dorely Morales
- Margarita Muñoz
- Javier Valencia
- Guillermo Zarazúa

Herramientas de trabajo



Objetivos del proyecto:

- Investigación sobre el uso y explotación de LIBMF.
- Exploración de la librería Python-LIBMF: clase MF, Método FIT y Predict.
- Generar un sistema de recomendación de películas para usuarios de Netflix a partir de las películas donde se estime que los usuarios darán las calificaciones más altas sobre una base muestra.
- Evaluación del Desempeño de la Implementación en LIBMF para obtener predicciones de calificaciones sobre la base completa del concurso de Netflix (*Netflix Prize*).

Sistemas de Recomendación y su aplicación en LIBMF

A partir del **enfoque colaborativo** de sistemas de recomendación en el contexto de películas, la librería LIBMF (*A LIBrary for large-scale sparse Matrix Factorization*), es utilizada para la predicción de calificaciones o *ratings r* que dará un usuario a las películas en una muestra de validación con base en la factorización de la matriz de entrenamiento *R* con las calificaciones observadas por *m* usuarios y *n* películas en términos de 2 matrices P y Q. Para ello, usaremos **datos explícitos** y se usó la base de datos de *Netflix (una implementación con una muestra de datos y otra con la base completa del famoso concurso Netflix Prize)* para sugerir aquellas películas con calificaciones más altas.

LIBMF encuentra una factorización para aproximar la matriz *R* de dimensiones $m \times n$ por dos matrices: *P* de $k \times m$ y *Q* de $k \times n$ de tal forma que:

$$R \approx \hat{R} = P^{\mathrm{T}}Q$$

Factorizaciones en LIBMF para diferentes tipos de recomendación

LIBMF es una librería enfocada a encontrar una factorización matricial (MF) para una matriz *R* en términos de *P* y *Q*. Se puede trabajar con los siguientes tipos de factorizaciones:

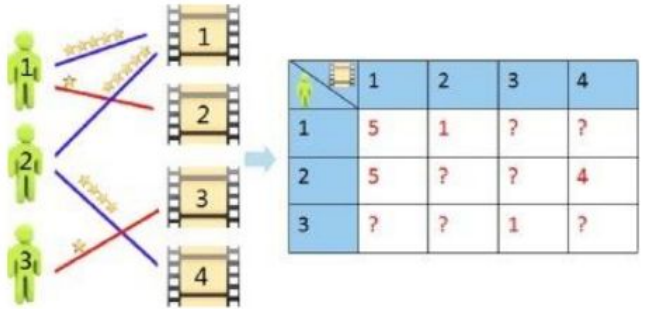
1. Real-Valued Matrix Factorization (RVMF)
2. Binary Matrix Factorization (BMF)
3. Non-Negative Matrix Factorization (NMF)
4. One-Class Matrix Factorization (OCMF)

Real-Valued Matrix Factorization (RVMF) y Binary Matrix Factorization (BMF)

El dominio de las entradas en R pueden ser valores reales o un conjunto binario {-1, 1} para cubrir los dos escenarios.

$$\begin{pmatrix} 1 & -1 & -1 & -1 \\ -1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 \\ -1 & -1 & 1 & 1 \end{pmatrix}$$

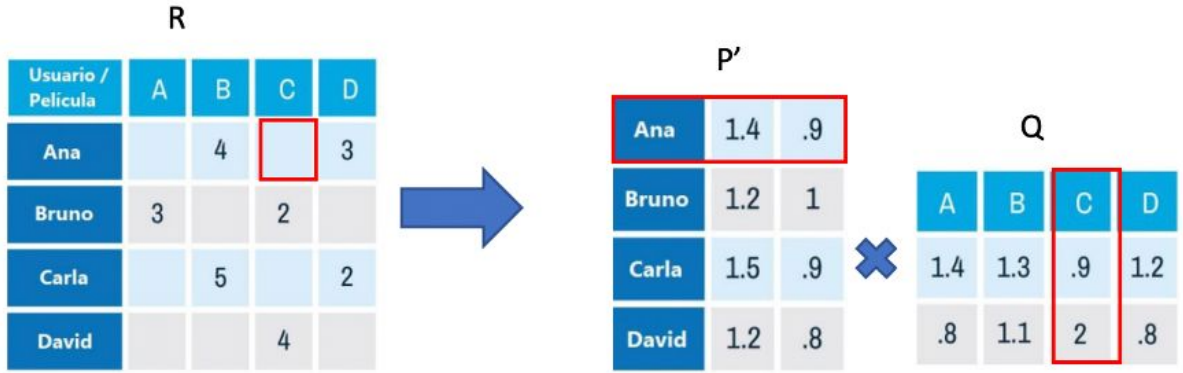
Non-Negative Matrix Factorization (NMF) y One-Class Matrix Factorization (OCMF)



	1	2	3	4
1	5	1	?	?
2	5	?	?	4
3	?	?	1	?

$$\begin{pmatrix} 1 & ? & ? & ? \\ ? & 1 & 1 & ? \\ 1 & 1 & 1 & ? \\ ? & ? & 1 & 1 \end{pmatrix}$$

Ejemplo práctico en sistemas de recomendación.



Acerca de LIBMF

Problema de optimización de nuestro caso práctico

Sobre nuestro caso particular de sistemas de recomendación de la base de Netflix, el problema de optimización no convexo de nuestro proyecto se expresa de la siguiente manera:

$$\min_{P,\ Q} \sum_{(u,\ v) \in R} \left((r_{u,\ v} - \mathbf{p}_u^T \mathbf{q}_v)^2 + \lambda_P \| \mathbf{p}_u \|^2 + \lambda_Q \| \mathbf{q}_v \|^2 \right)$$

Algoritmo para Paralelización: FPSG

Si bien el método de descenso en gradiente estocástico es secuencial, algunos bloques de la matriz de calificaciones son mutuamente independientes (ya que las predicciones de las calificaciones se hacen para cada usuario) y sus correspondientes variables pueden actualizarse en paralelo.

Algoritmo FPSG

1: Revolver la matriz R.
2: Dividir R en un conjunto B de al menos (s+1)x(s+1) bloques.
3: Ordenar cada bloque por usuario.
4: Programar los s threads.
5: Lanzar s threads para hacer el descenso en gradiente estocástico.
6: Esperar hasta que se realice el número de iteraciones determinadas por el usuario.

Sobre los parámetros disponibles...

La librería contiene muchas opciones a elegir para encontrar la factorización matricial en términos de P y Q. También cuenta con valores default para los mismos.

- Función de pérdida (f). (default error cuadrático medio)
- Parámetros de regularización para P y Q (l1,l2). (default 0 y 0.1, respectivamente)
- Dimensiones latentes (k). (default 8)
- Iteraciones (t). (default 20)
- Tasa de Aprendizaje (eta). (default 0.1)
- Threads(s). (default 12)
- Número de folds (v).
- Factorización no negativa (nmf).
- Error (e). (default raíz del error cuadrático medio)

Principales Diferencias:

LIBMF

- Tienen funciones mf-train y mf-predict.
- Genera un archivo con las columnas de las matrices P y Q (modelo).
- Genera un archivo con las predicciones y muestra el error de validación.
- La matriz de entrenamiento y validación deben estar en archivos de texto plano
- Muestra cada una de las iteraciones con su error.

python-LIBMF

- Tiene métodos fit y predict.
- Con los métodos Q y P se pueden obtener las matrices.
- Muestra el vector de las predicciones
- La matriz de entrenamiento y validación pueden generarse con funciones en python

LIBMF

```
!/home/miuser/repo_libmf/mf-train -l1 0.05 -l2 0.01 real_matrix.tr.txt model
```

python-LIBMF

```
mf_netflix=mf.MF(eta=0.11,nr_iters=21, k=5)
```

```
mf_netflix.fit(MC_entrena)
```

Resultados Sistema de Recomendación: Base Muestra de Netflix

Sobre la estructura de los datos de muestra

100,000 usuarios y 17,770 películas. Para esta base de datos se tiene un total de 20,968,941 de entradas ya que no todos los usuarios califican todas las películas.

Para la evaluación del desempeño de libmf en la predicción de las calificaciones, se separó el dataset en una muestra de entrenamiento y validación a partir de las columnas peli_id, y usuario_id.

Seleccionando aleatoriamente el 20% de los usuarios y películas en validación y el resto en entrenamiento.

	peli_id	usuario_id_orig	calif	fecha	usuario_id
0	1	2442	3	2004-04-14	1
1	1	1086807	3	2004-12-28	2
2	1	2165002	4	2004-04-06	3
3	1	1133214	4	2004-03-07	4
4	1	1537427	4	2004-03-29	5
5	1	525356	2	2004-07-11	6
6	1	1910569	4	2004-04-12	7
7	1	2421815	2	2004-02-26	8
8	1	2508819	3	2004-05-18	9
9	1	1342007	3	2004-07-16	10

Algunos insights sobre el conjunto de datos

El número de calificaciones que realizan los usuarios es muy variable, y son sólo pocos los que generan un gran número de calificaciones (entre los primeros 10 usuarios que califican más, los 2 primeros emitieron más de 14 mil calificaciones, pero el que está en el lugar no. 10 sólo generó 4,731 calificaciones).

Algunos insights sobre el conjunto de datos

Distribución población acorde numero de calificaciones que recibió cada película

num_calificaciones_group	percentil
(1, 25]	0.1
(25, 34]	0.2
(34, 48]	0.3
(48, 74]	0.4
(74, 118]	0.5
(118, 210]	0.6
(210, 404]	0.7
(404, 839]	0.8
(839, 2578]	0.9
(2578, 48633]	1.0

Distribución población acorde la cantidad de películas que cada usuario calificó

pelis_calificadas_group	percentil
(1, 19]	0.1
(19, 32]	0.2
(32, 46]	0.3
(46, 66]	0.4
(66, 96]	0.5
(96, 142]	0.6
(142, 211]	0.7
(211, 322]	0.8
(322, 542]	0.9
(542, 15813]	1.0

El 60 % de las películas calificadas recibió entre 1 y 210 calificaciones

El 60 % de los usuarios emitió entre 1 y 142 calificaciones, por lo cual se puede decir que en general los usuarios aportan pocas pocas calificaciones.

Desempeño Modelo Implementado LIBMF (Bash)

Se alcanzó después de 10 iteraciones obteniendo un RMSE de 0.8130 sobre la muestra de validación (la base muestra se partió en entrenamiento y validación para hacer una primer prueba con el modelo). En general, el desempeño del algoritmo es bueno, pues se consiguió un buen ajuste en un tiempo muy breve (menos de 1 minuto).

Parámetros utilizados:

```
---------------------------------
Real-valued matrix factorization
---------------------------------
iter      tr_rmse      va_rmse          obj
   0       0.9955       0.9236   2.5693e+07
   1       0.9140       0.8804   2.2704e+07
   2       0.8802       0.8588   2.1623e+07
   3       0.8632       0.8447   2.1150e+07
   4       0.8514       0.8353   2.0847e+07
   5       0.8429       0.8283   2.0636e+07
   6       0.8368       0.8231   2.0489e+07
   7       0.8320       0.8196   2.0378e+07
   8       0.8281       0.8163   2.0289e+07
   9       0.8245       0.8130   2.0209e+07
MAE = 0.6374
```

```
$train -f 0 -l2 0.05 -k 100 -t 10 -p muestra_valida.te.txt muestra_entrena.tr.txt rvmf_model.txt
# Do prediction and show MAE
$predict -e 1 muestra_valida.te.txt rvmf_model.txt rvmf_output.txt
```

Análisis de similitud de películas para dimensiones latentes

Se encontraron agrupaciones muy claras:

	peli_id	año	nombre	dimension_late
3793	3794	2000.000000	Dragon Ball Z: Bardock: The Father of Goku	-0.277668
5318	5319	2002.000000	Dragon Ball: Red Ribbon Army Saga	-0.279332
15399	15400	2000.000000	Dragon Ball Z: Androids	-0.284217
15422	15423	2003.000000	Dragon Ball Z: Garlic Jr.	-0.289724
11743	11744	2002.000000	Dragon Ball: Tien Shinhan Saga	-0.294233
4625	4626	1995.000000	Dragon Ball: Piccolo Jr. Saga: Part 2	-0.295411
11091	11092	2000.000000	Dragon Ball Z: Trunks Saga	-0.306649
14427	14428	2003.000000	Dragon Ball Z: Babidi	-0.306686
16491	16492	2002.000000	Dragon Ball Z: Cooler's Revenge	-0.308157
1729	1730	1995.000000	Dragon Ball: Piccolo Jr. Saga: Part 1	-0.309546
9668	9669	2002.000000	Dragon Ball Z: Kid Buu Saga	-0.309684
16941	16942	2003.000000	Dragon Ball Z: Perfect Cell	-0.310708
11562	11563	1989.000000	Dragon Ball Z: Imperfect Cell Saga	-0.313106

Y otras no tanto:

	peli_id	año	nombre	dimension_latente_19
1125	1126	2003.000000	Uncovered: The Whole Truth About the Iraq War	0.316326
10264	10265	2002.000000	Sex Pistols: Never Mind the Bollocks	0.301553
16733	16734	2004.000000	Uncovered: The War on Iraq	0.297189
7565	7566	1985.000000	Friday the 13th: Part 5: A New Beginning	0.295378
247	248	2001.000000	Michael Moore's The Awful Truth: Season 2	0.289096
1667	1668	1982.000000	Friday the 13th: Part 3	0.285690
11660	11661	1984.000000	Friday the 13th: Part 4: The Final Chapter	0.283938
11842	11843	1993.000000	NYPD Blue: Season 1	0.282106
3262	3263	2004.000000	Broadway: The Golden Age	0.279681
15663	15664	1987.000000	I've Heard the Mermaids Singing	0.278438
11021	11022	2004.000000	Fahrenheit 9/11	0.276949
4921	4922	2002.000000	Sorcerers and Wizards: Real Magic	0.276621
4180	4181	2003.000000	Walking with Cavemen	0.275520

Análisis de recomendaciones para un usuario fijo

Pelis vistas:

nombre
Dinosaur Planet
Contact
Harry Potter and the Chamber of Secrets
A League of Their Own
Aladdin and the King of Thieves
The Great Escape
Hotel Rwanda
Cadence
The Man with the Golden Gun
Ferngully: The Last Rainforest
The Sopranos: Season 5
Swordfish
Indiana Jones and the Temple of Doom
Airplane!
The Delta Force

Pelis Recomendadas:

nombre
Revenge of the Ninja
Chinese Box
X: The Unheard Music
South Park: Season 1
Dickie Roberts: Former Child Star
Strange Days
The French Chef with Julia Child
New York Firefighters: The Brotherhood of 9/11
K-19: The Widowmaker
The Twilight Zone: Vol. 27
Sophie's Choice
The Dukes of Hazzard: Season 3
Superstars of '70s Soul Live
The Big Empty
Matchbox Twenty: Show

Se observa un gusto por películas animadas o de fantasía y populares como Harry Potter y Aladdin. En contraste, las pelis más recomendadas incluyen títulos similares como Fantasía (de Disney), The Dukes of Hazzard, aunque hay otras tantas de la que es difícil encontrar una relación de manera directa (Revenge of the Ninja)

Resultados Modelo sobre Base Completa de

Limpieza de la base de datos de Netflix

1. Extracción

Extraemos datos con API de Kaggle

1.99 GB

- README
- combined_data_1.txt
- combined_data_2.txt
- combined_data_3.txt
- combined_data_4.txt
- movie_titles.csv
- probe.txt
- qualifying.txt

2. Consolidación

Concatenamos los 4 txts en un **GRAN DATAFRAME**

100 Millones

```
Dataset 1 shape:
(24058263, 2)
Dataset 2 shape:
(26982302, 2)
Dataset 3 shape:
(22605786, 2)
Dataset 4 shape:
(26851926, 2)
```

```
Full dataset shape: (100498277, 2)
-Dataset examples-
           Cust_Id  Rating
0               1:     NaN
5000000    2560324     4.0
10000000   2271935     2.0
15000000   1921803     2.0
20000000   1933327     3.0
25000000   1465002     3.0
30000000    961023     4.0
35000000   1372532     5.0
40000000    854274     5.0
45000000    116334     3.0
50000000    768483     3.0
55000000   1331144     5.0
60000000   1609324     2.0
65000000   1699240     3.0
70000000   1776418     4.0
75000000   1643826     5.0
80000000    932047     4.0
85000000   2292868     4.0
90000000    932191     4.0
95000000   1815101     3.0
100000000   872339     4.0
```

3. Limpieza

Eliminamos filas con Movie_ID y agregamos una columna con estos

Guardamos datos en un CSV

Cust_Id	Rating	Movie_Id
1488844	3.0	1
501954	2.0	996
404654	5.0	1962
886608	2.0	2876
1193835	2.0	3825
1899206	3.0	4661
154804	4.0	5496
2078749	5.0	6274
450763	5.0	7057
102092	3.0	7991
220298	5.0	9023
550530	5.0	10042
222570	3.0	11038
1273080	5.0	11875
2026970	5.0	12676

Filtrado de base, separación de train y test

Filtrado (Pruning)

Filtramos películas y usuarios con pocas evaluaciones para reducir el tamaño del dataset y mejorar las predicciones.

Método de percentiles (percentil 60):

- Mínimo de calificaciones por **película**: **1,006**
- Mínimo de calificaciones por **usuario: 142**

Separación en entrenamiento y prueba

Aplicamos un random shuffle y dividimos:

- **80%** conjunto de **entrenamiento**
- **20%** conjunto de **prueba**

Guardamos datos en dos archivos txt con el formato adecuado para LIBMF.

Cuando se generó la base limpia de Netflix se guardó en el formato equivocado en (CSV), por lo que se tuvo que transformar a un txt sin comas. esto se hizo usando **AWK**

```
Comando AWK  para transformar CSV a txt
awk -F "\"*,\"*" '{if (NR≠1) {print $2,$4,$3}}'
netflix_cleaned.csv >netflix_cleaned.txt
```

Desempeño del Modelo Implementado en LIBMF (Bash)

Código

```
ubuntu@ip-172-31-12-73:~/datasets/libmf/demo$ nano netflixdemo.sh

  GNU nano 2.9.3                          netflixdemo.sh

#!/bin/sh
train=../mf-train
predict=../mf-predict

################################################################################
# Build package if no binary found and this script is exectuted via the
# following command.
#  libmf/demo > sh demo.sh
################################################################################
if [ ! -s $train ] || [ ! -s $predict ]
then
    (cd .. && make)
fi

################################################################################
# Real-valued matrix factorization (RVMF)
################################################################################
echo "--------------------------------"
echo "Real-valued matrix factorization"
echo "--------------------------------"
# In-memory training with holdout valudation
$train -f 0 -l2 0.05 -k 100 -t 10 -p netflix_test.te.txt netflix_train.tr.txt rvmf_model.txt
# Do prediction and show MAE
$predict -e 1 netflix_test.te.txt rvmf_model.txt rvmf_output.txt
```

Parámetros utilizados

Resultados

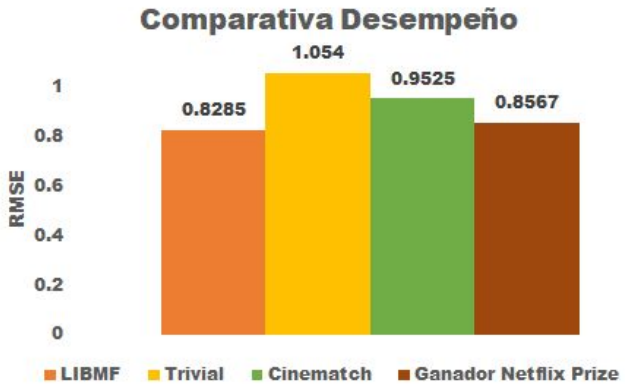
```
--------------------------------
Real-valued matrix factorization
--------------------------------
iter      tr_rmse      va_rmse          obj
   0       0.9486       0.9185   7.7184e+07
   1       0.8920       0.8717   7.1176e+07
   2       0.8620       0.8570   6.8188e+07
   3       0.8480       0.8471   6.7049e+07
   4       0.8379       0.8404   6.6245e+07
   5       0.8314       0.8365   6.5754e+07
   6       0.8270       0.8338   6.5441e+07
   7       0.8238       0.8317   6.5216e+07
   8       0.8211       0.8299   6.5045e+07
   9       0.8187       0.8285   6.4888e+07
MAE = 0.6493
-----------------------------
```

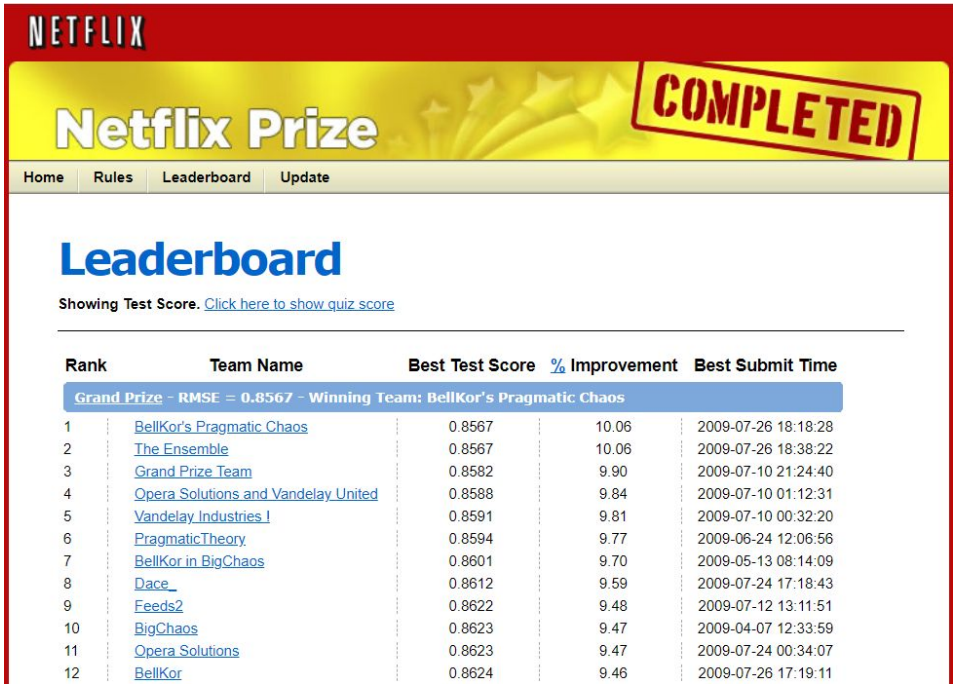
Se alcanzó convergencia en sólo 10 iteraciones
RMSE de **0.8187** en entrenamiento
RMSE de **0.8285** en validación.
Tiempo de cómputo **7 minutos**

Evaluación de Nuestros Resultados:

(RMSE de 0.8285)

Benchmarks de Referencia sobre Test:





Conclusiones:

- LIBMF es una herramienta poderosa para encontrar una descomposición de R por un producto de matrices ralas en cuestión de minutos.
- A pesar de que los resultados de la predicción de las calificaciones tuvieron un desempeño muy bueno en términos de la raíz del error cuadrático medio, este sistema de recomendación debería combinarse con otros enfoques como los basados en contenido para mejorar los resultados. Adicionalmente, se debería integrar un proceso adicional para atacar el problema de la heterogeneidad en uso de escala.
- Cabe mencionar que estrictamente hablando no podemos comparar nuestros resultados con los benchmarks de referencia, pues nosotros realizamos procesos de filtrado a la base total, y el test del concurso utilizaba una muestra de prueba separada de películas vistas después del periodo de entrenamiento.
- La documentación sobre el uso de la librería con ejercicios prácticos es casi nula y por ello, todos los resultados se dejarán como un manual que sirva como guía ayudar a la comunidad a incrementar su uso práctico.

¡GRACIAS!