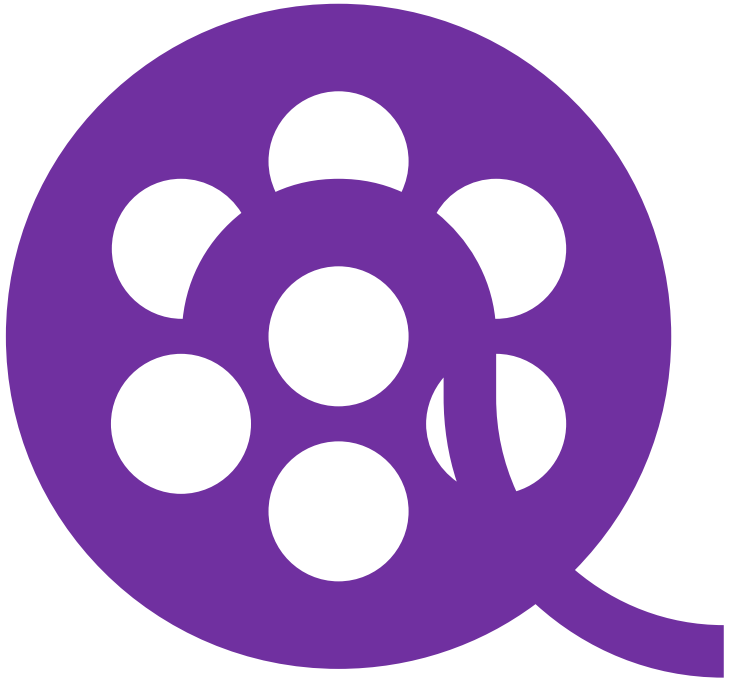
https://github.com/louisyang2015/movie_recommender



Movie Recommender

Louis Yang

2019 January

Technologies Used

Mostly Python (Numpy, SciPy)

C++ (ALS)

JavaScript (App)

MovieLens dataset

AWS (optional)

- Storage: S3, DynamoDB, EFS
- Compute: EC2, Lambda
- Web Hosting

Linear Model

(user profile)•(item profile) = score

Example:

	Science Fiction	Romance
Louis	1	0.1

	Science Fiction	Romance
Star Trek	1	0.05
Titanic	0.1	1

(Louis) • (Star Trek) = 1 * 1 + 0.1 * 0.05 = 1.005

(Louis) • (Titanic) = 1 * 0.1 + 0.1 * 1 = 0.2

ALS Model

Example:

	?	?
Louis	$u_{0,0}$	$u_{0,1}$

	?	?
Star Trek	$m_{0,0}$	$m_{0,1}$
Titanic	$m_{1,0}$	$m_{1,1}$

(Louis) • (Star Trek) = $u_{0,0}$ $m_{0,0}$ + $u_{0,1}$ $m_{0,1}$ = +2.5

(Louis) • (Titanic) = $u_{0,0}$ $m_{1,0}$ + $u_{0,1}$ $m_{1,1}$ = +0.5

ALS Solution

Randomize $\vec{m}$, solve for $\vec{u}$.

$u_{0,0}\ 0.3 + u_{0,1}\ 0.5 = +2.5$
$u_{0,0}\ 0.25 + u_{0,1}\ 0.6 = +0.5$

$$\vec{u} = \begin{bmatrix} 22.73 \\ -8.64 \end{bmatrix}$$

now solve for $\vec{m}$

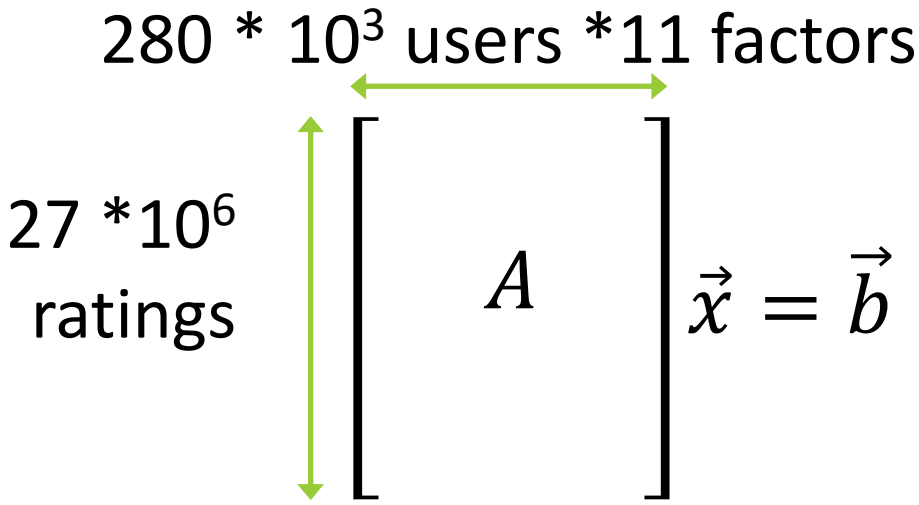
$22.73\ m_{0,0} + -8.64\ m_{0,1} = +2.5$
$22.73\ m_{1,0} + -8.64\ m_{1,1} = +0.5$

then solve for $\{\vec{m}, \vec{u}, \vec{m}, \ldots\}$

ALS Sparse Matrix

Each equation is a single (user, movie, rating)

As of 2018 September, the MovieLens dataset:



Solving ALS in Python

python\basics>python **lsmr.py**

python\basics>python **als.py**

- generate data using $\overrightarrow{user} \cdot \overrightarrow{movie}$
- split data into training and test sets
- train ALS models using training data
- test on test data

Rank Agreement Percentage

Rating is not a good metric

- Categorical data (scores are approximate)
- Users see ranking
- Some algorithms don’t use produce scores

Rank Agreement Percentage

- Range 0 ~ 100%
- Top heavy

Algorithms

python\100k_data>

- python count_tags.py
- python ls_tag.py
- python median_predictor.py
- python ratings_als.py

- python similar_movies.py
- python title_search.py > temp.txt

Movie Median vs Movie Average

Average

- Standard (least squares) uses average as the bias term

Median

- More responsive to the model

	User 1	User 2	User 3	Average	Median
Movie 1	3	3	4	3.33	3
Movie 2	2	3	4	3	3

Big Data Techniques

Pickle

Multiprocessing

- python\basics\multiprocessing>python add.py

Multithreading using C++ (ctypes)

- cpp\python>python cpp_ls_test.py

Cluster

- python\basics\distributed_work>python cluster_server.py, worker_server.py, add.py

AWS Setup (EC2, EFS)

Setup EC2 instance:

- Python3, numpy, scipy, boto3, requests
- C++

Upload all scripts in "python\full_data"

Mount EFS using "/etc/fstab"

- fs-155b98bd:/ /home/ec2-user/full_data/data efs defaults,_netdev 0 0

Build C++ ALS binary

Upload to AWS

- matrix.h, matrix.cpp, ls_linux_dll.cpp

Build C++ shared library file

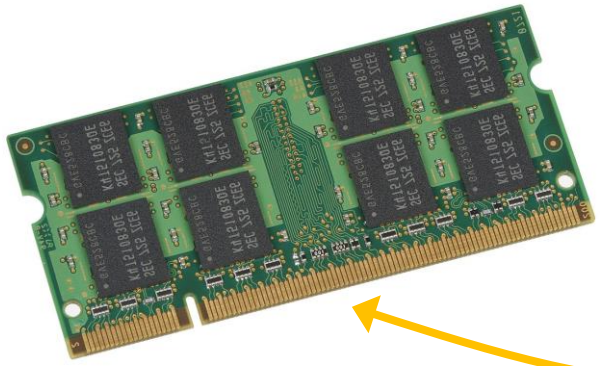
- g++ -O3 -fPIC -shared matrix.cpp matrix.h ls_linux_dll.cpp -o cpp_ls_lib.so -pthread -std=c++11
- chmod 644 cpp_ls_lib.so
- Testing (optional): python3 cpp_ls_test.py

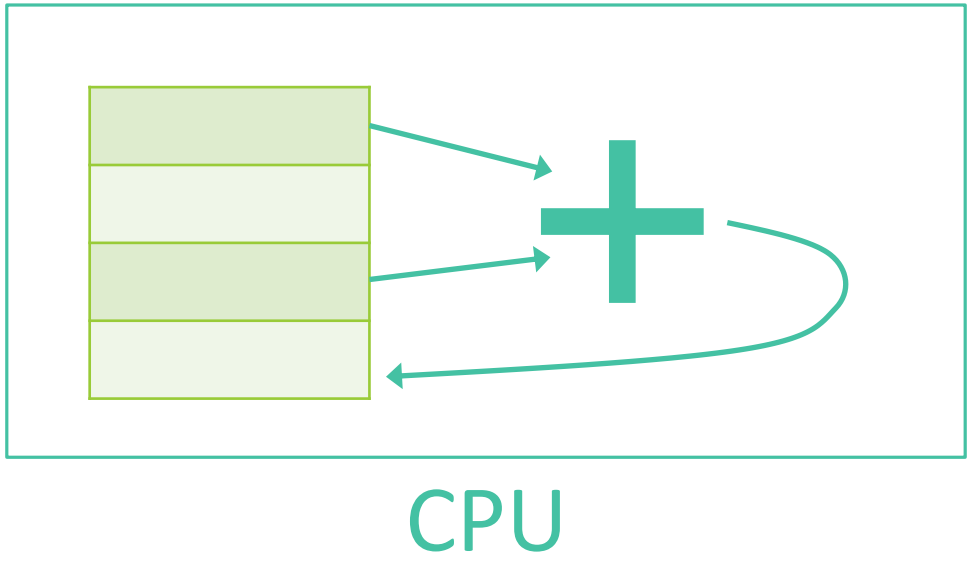
AWS EC2 Worker

build_similar_movies_db.py – 3GB per core

- If 2GB / core, then use half of total number of cores

ALS Benchmark





Model Evaluation (Non ALS)

Data Processing

- worker: sudo python3 randomize_training_set.py

Model Evaluation

- server: sudo python3 cluster_server.py
- worker: sudo python3 worker_server.py
- worker: sudo python3 median_predictor.py, tag_count_predictor.py, tag_ls_predictor.py

ALS Model Evaluation

Data Processing

- worker: sudo python3 train_als_models.py

Model Evaluation

- server: sudo python3 cluster_server.py
- worker: sudo python3 worker_server.py
- worker: sudo python3 als_predictor.py

ALS Full Dataset Models

alsX_item_factors.bin, alsX_movie_ids.bin

- sudo python3 train_als_models.py training_set_ratio=1
- download → "python\app\recommend"
- download → "python\app_local"

Similar Movies

Similar Movies (similar_movies.bin)

- server: sudo python3 cluster_server.py
- This step requires 3GB / Core
  If using 2GB / core, 16 cores:
  worker: sudo python3 worker_server.py cpu_count=8
- worker: sudo python3 build_similar_movies_db.py

Use "user data" to auto start "worker_server.py"

```
#!/bin/sh
cd /home/ec2-user
./startup
```

```
cd full_data
sudo python3 worker_server.py
```

Other Data Processing

Windows (title_search_index.bin, tmdb_data.bin)

- python build_title_search_index.py
- python download_tmdb_data.py

Application files: python\app_local

App Front End

Basic JavaScript only

Example:

-
- class ModelParams

API Example: recommend

App Local

E:\proj2018\movies_recommend\python\app_local>
python server.py

Open: python\app_local\web_page\index.html

App Back End

Lambda

DynamoDB

Build Process

- Windows (python\app\):
  - copy in the ALS models
  - python build.py
- Linux
  - cd recommend
    - sudo python3 -m pip install numpy --target .
    - sudo python3 -m pip install scipy --target .
  - python3 ec2_build.py