MAC306

Using MXNet for Recommendation Modeling at Scale

Leo Dirac, Principal Engineer, AWS Deep Learning

November 30, 2016

amazon web services

What to Expect from the Session

Background on recommender systems and machine learning.

Learn how to implement them on MXNet using p2 instances and the AWS Deep Learning AMI.

Explore several types of recommender systems, including advanced deep learning ideas.

Learn tricks for handling sparse data in MXNet.

Background: Recommender Systems & Machine Learning

Netflix Prize: 2006-2009
$1,000,000

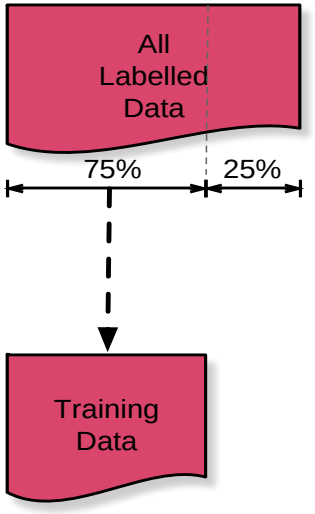
Recommending Movies

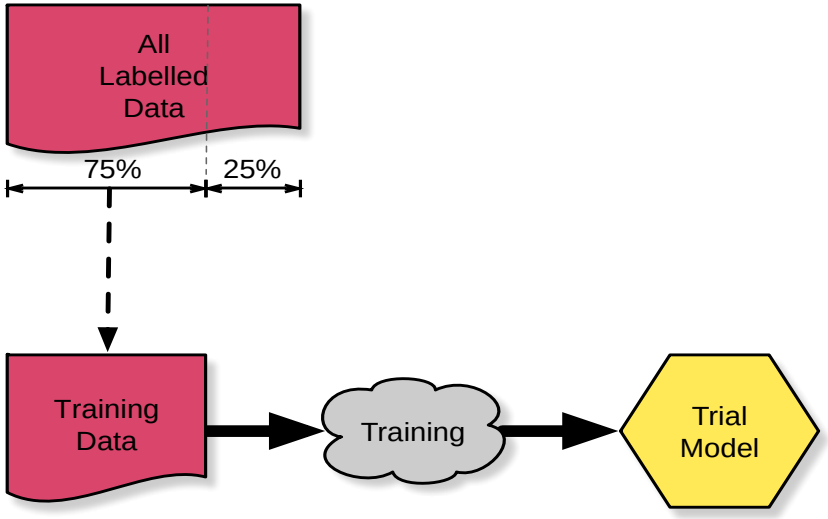
```
/* Predict what Star Rating will user u give
movie m */

float predictRating(User u, Movie m) {
  // How???
}
```

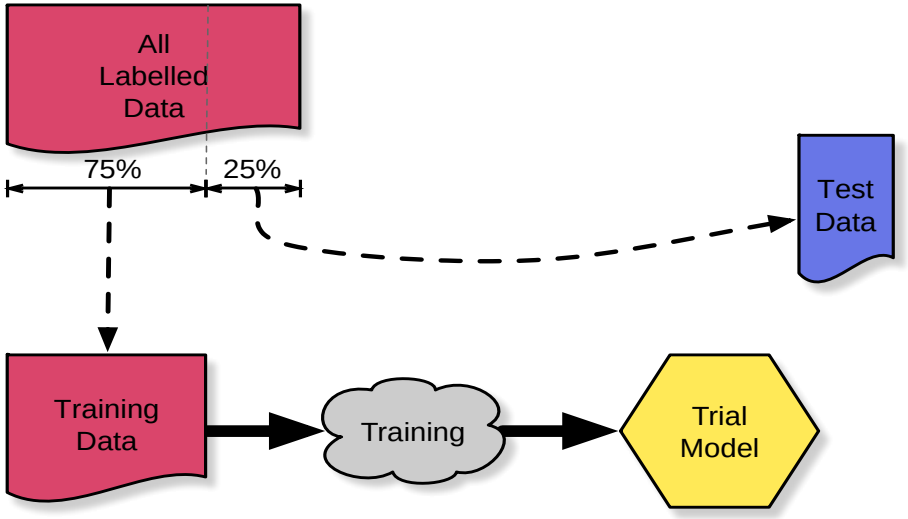
Q: How???
A: Machine Learning: Learn code from data

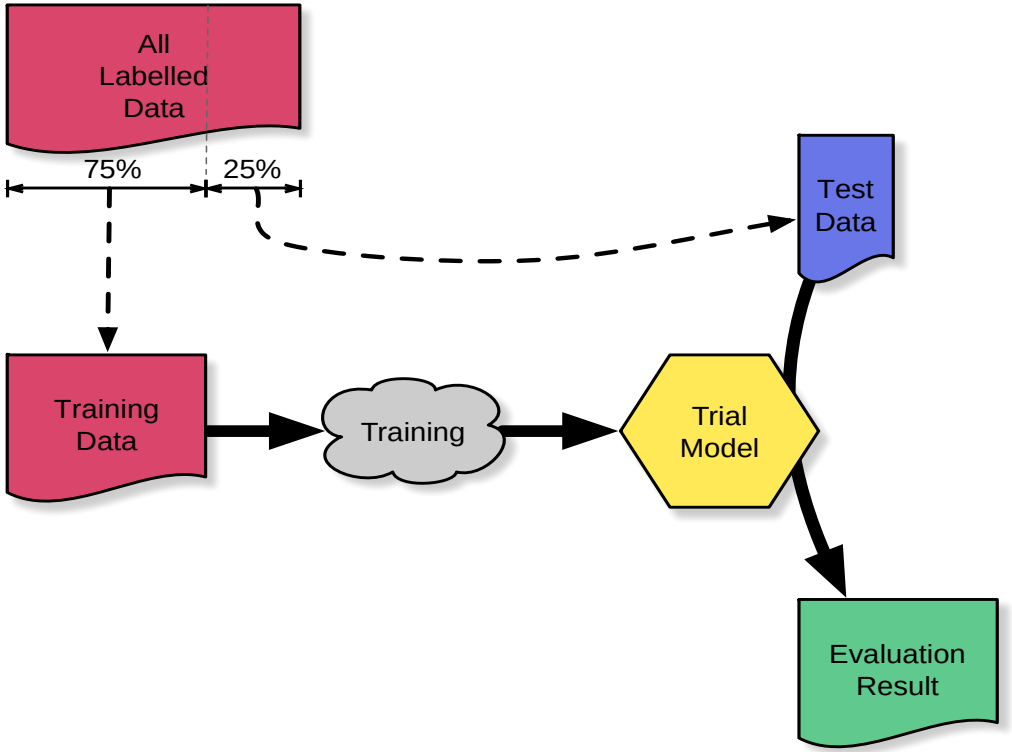
```
float predictRating(User u, Movie m) {
    return mlModel.run(u,m);
}
```

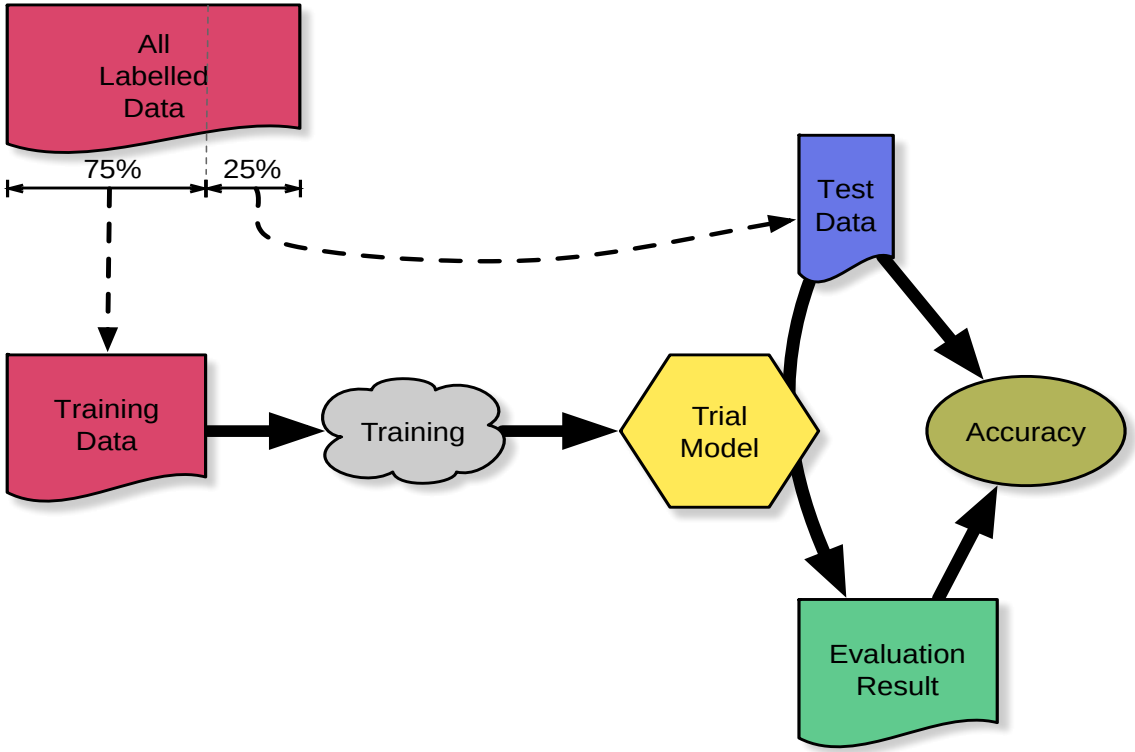

Training
Data
Training
Model


All
Labelled
Data
75%
25%
Training
Data


All
Labelled
Data
75%
25%
Training
Data
Training
Trial
Model

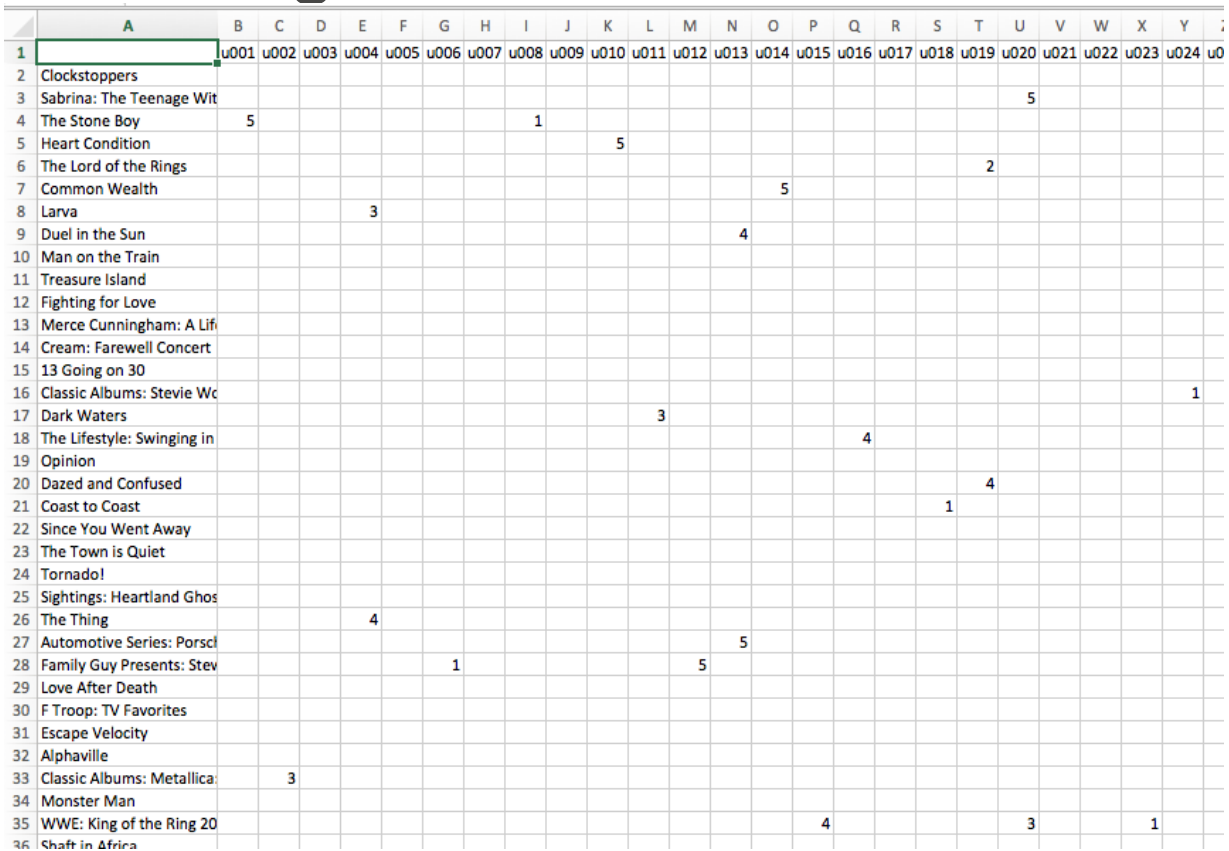

All
Labelled
Data
75%
25%
Test
Data
Training
Data
Training
Trial
Model


All
Labelled
Data
75%
25%
Test
Data
Training
Data
Training
Trial
Model
Evaluation
Result


All
Labelled
Data
75%
25%
Test
Data
Training
Data
Training
Trial
Model
Accuracy
Evaluation
Result

Sparse Data

User-Item Ratings Matrix



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y
1		u001	u002	u003	u004	u005	u006	u007	u008	u009	u010	u011	u012	u013	u014	u015	u016	u017	u018	u019	u020	u021	u022	u023	u024
2	Clockstoppers																								
3	Sabrina: The Teenage Wit																				5				
4	The Stone Boy	5							1																
5	Heart Condition										5														
6	The Lord of the Rings																			2					
7	Common Wealth														5										
8	Larva				3																				
9	Duel in the Sun													4											
10	Man on the Train																								
11	Treasure Island																								
12	Fighting for Love																								
13	Merce Cunningham: A Lif																								
14	Cream: Farewell Concert																								
15	13 Going on 30																								
16	Classic Albums: Stevie Wo																								1
17	Dark Waters											3													
18	The Lifestyle: Swinging in																4								
19	Opinion																								
20	Dazed and Confused																			4					
21	Coast to Coast																		1						
22	Since You Went Away																								
23	The Town is Quiet																								
24	Tornado!																								
25	Sightings: Heartland Ghos																								
26	The Thing				4																				
27	Automotive Series: Porsch													5											
28	Family Guy Presents: Stev						1						5												
29	Love After Death																								
30	F Troop: TV Favorites																								
31	Escape Velocity																								
32	Alphaville																								
33	Classic Albums: Metallica:		3																						
34	Monster Man																								
35	WWE: King of the Ring 20															4					3			1	
36	Shaft in Africa																								

Size of user-item ratings matrix

Sample dataset: MovieLens 20M

(27,000 movies) * (138,000 users)
= 3,700,000,000 possible ratings

But only 20,000,000 ratings available.

99.5% of ratings are unknown.

http://grouplens.org/datasets/movielens/20m/

Storing the matrix

Dense
3.7B entries

Each entry:
•Rating: 1 byte

Sparse
20M non-zero entries

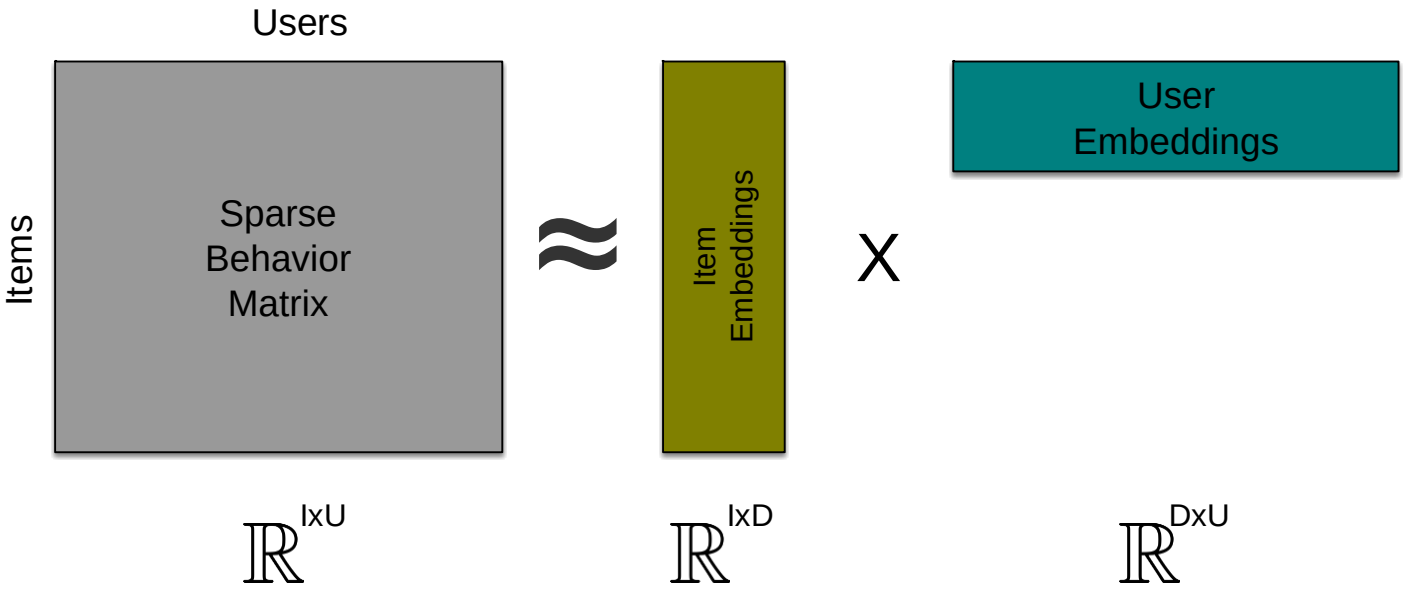
Each entry:
•Rating: 1 byte
•Movie_id: 32-bit integer
•User_id: 32-bit integer

3.7 GB **180 MB**

Sparse is 20x smaller

Matrix Factorization

MF as Math


Users
Items
Sparse
Behavior
Matrix
≈
Item
Embeddings
X
User
Embeddings
$\mathbb{R}^{IxU}$
$\mathbb{R}^{IxD}$
$\mathbb{R}^{DxU}$

Embeddings

Emb(“The Karate Kid”) =

[-3.168
-0.136
3.770
4.767
3.558
-4.168
0.464
2.034
3.411
…
0.866]

Amazon

Embeddings

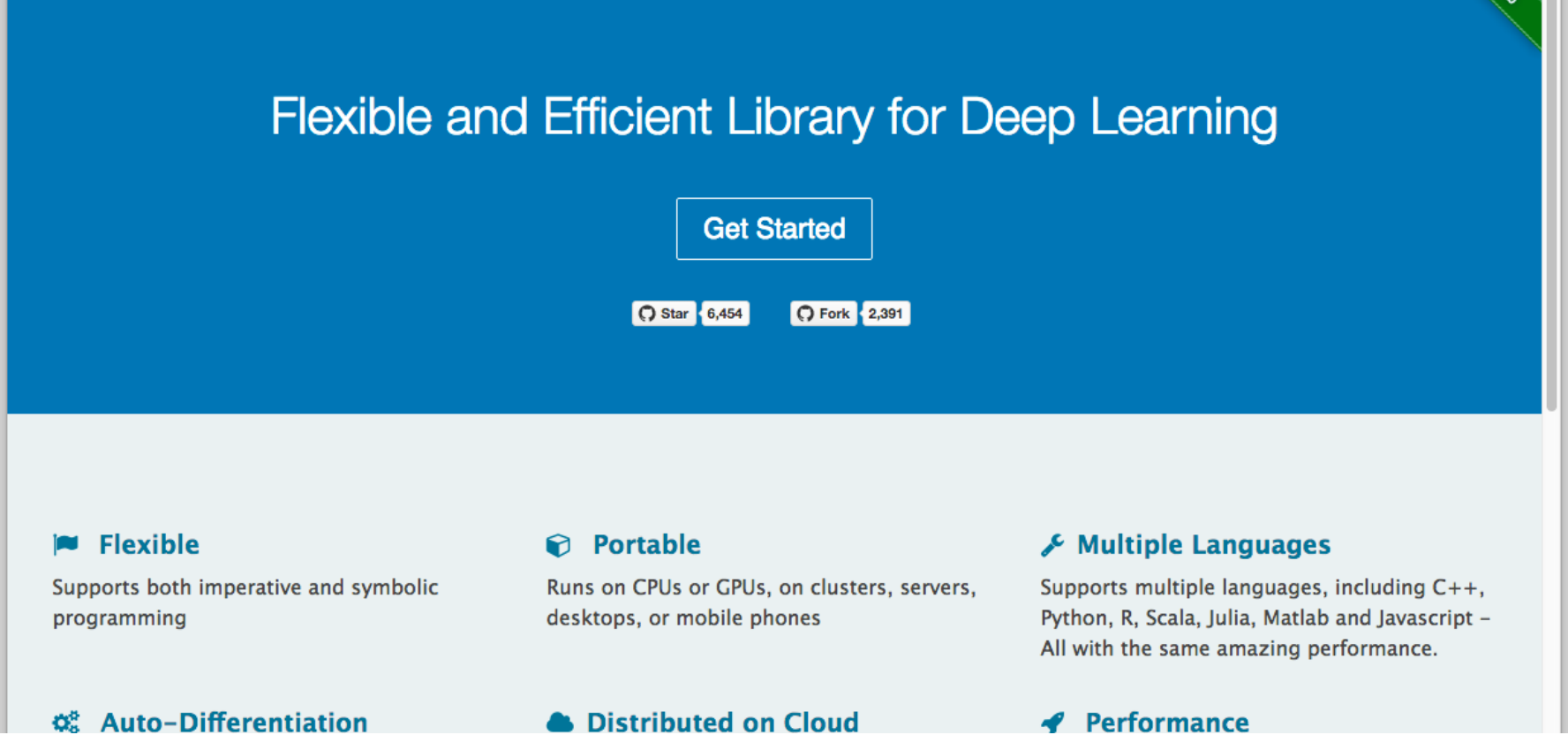
Emb(“The Karate Kid”) = [-3.168
-0.136
3.770
4.767
3.558
-4.168
0.464
2.034
3.411
…
0.866]

Emb(“Ferris Bueller”) = [-3.101
-0.057
3.800
4.862
3.632
-4.157
0.549
2.064
3.428
…
0.884]

D(Emb(“K.Kid”) – Emb(“Ferris”)) = 0.138

D(Emb(“K.Kid”) – Emb(“My Little Pony”)) = 1.572

MXNet

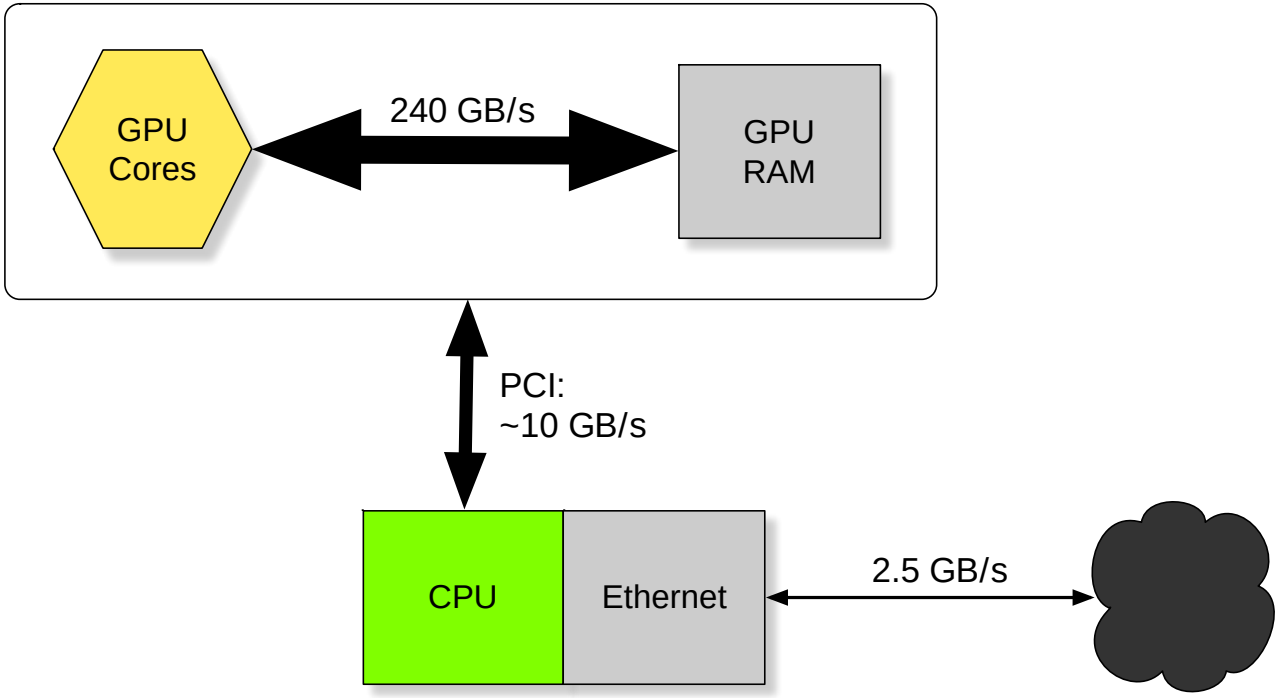

Flexible and Efficient Library for Deep Learning
Get Started
Star 6,454
Fork 2,391
Flexible
Supports both imperative and symbolic programming
Portable
Runs on CPUs or GPUs, on clusters, servers, desktops, or mobile phones
Multiple Languages
Supports multiple languages, including C++, Python, R, Scala, Julia, Matlab and Javascript – All with the same amazing performance.
Auto-Differentiation
Distributed on Cloud
Performance

p2.xlarge

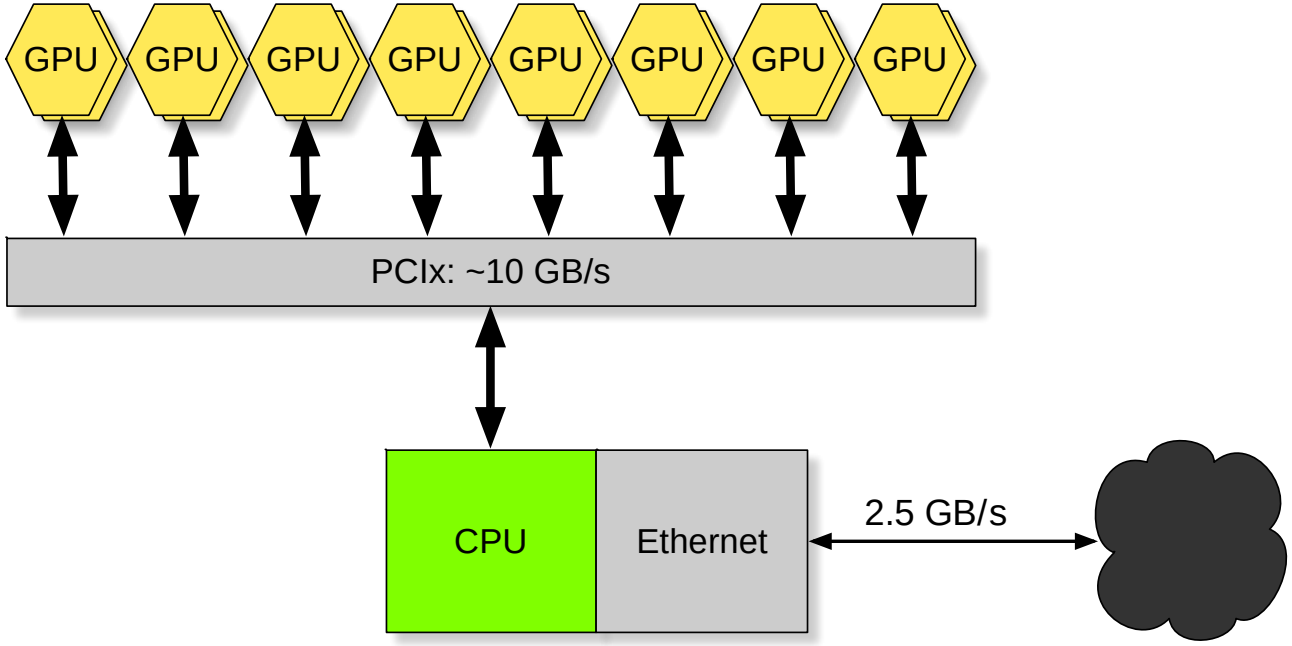
4,300,000,000,000

32-bit floating point operations/second

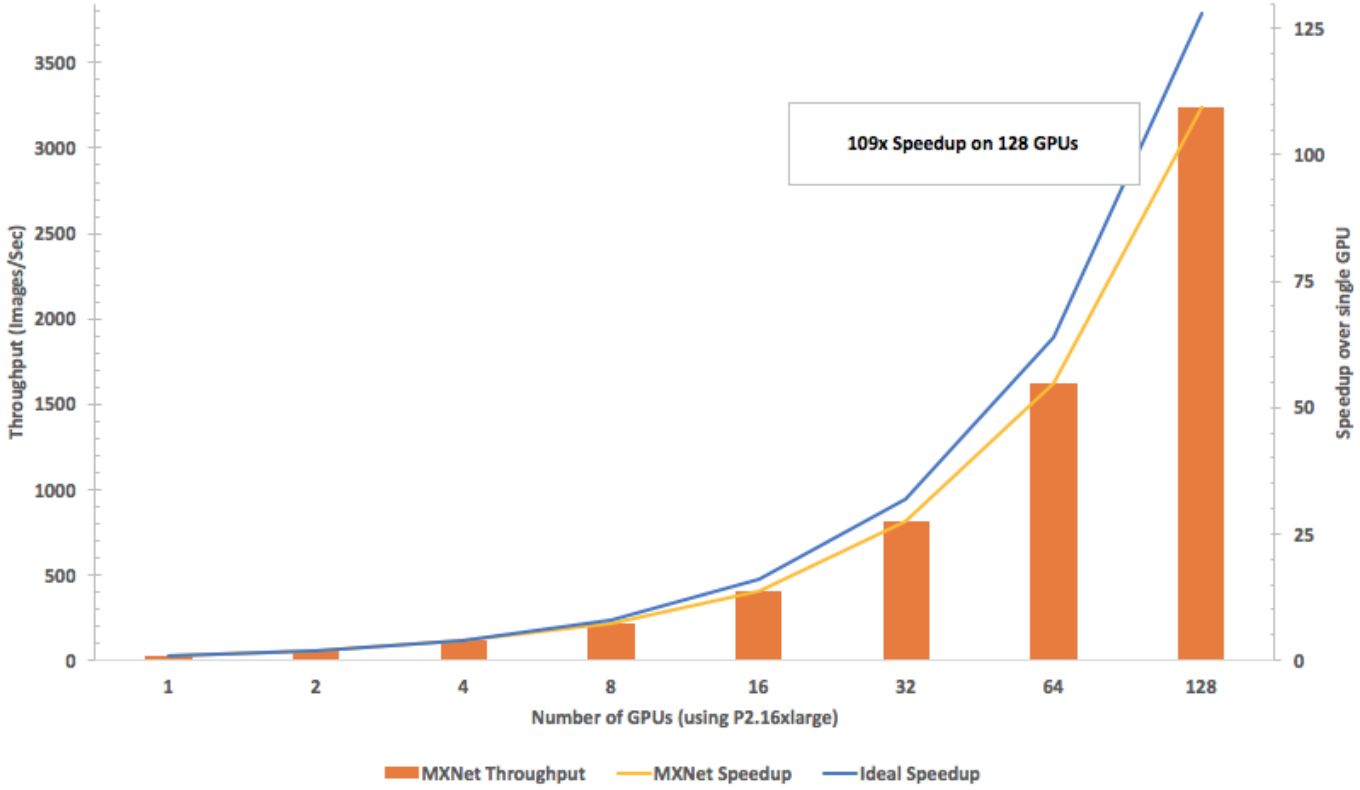
GPUs: Feeding the beast



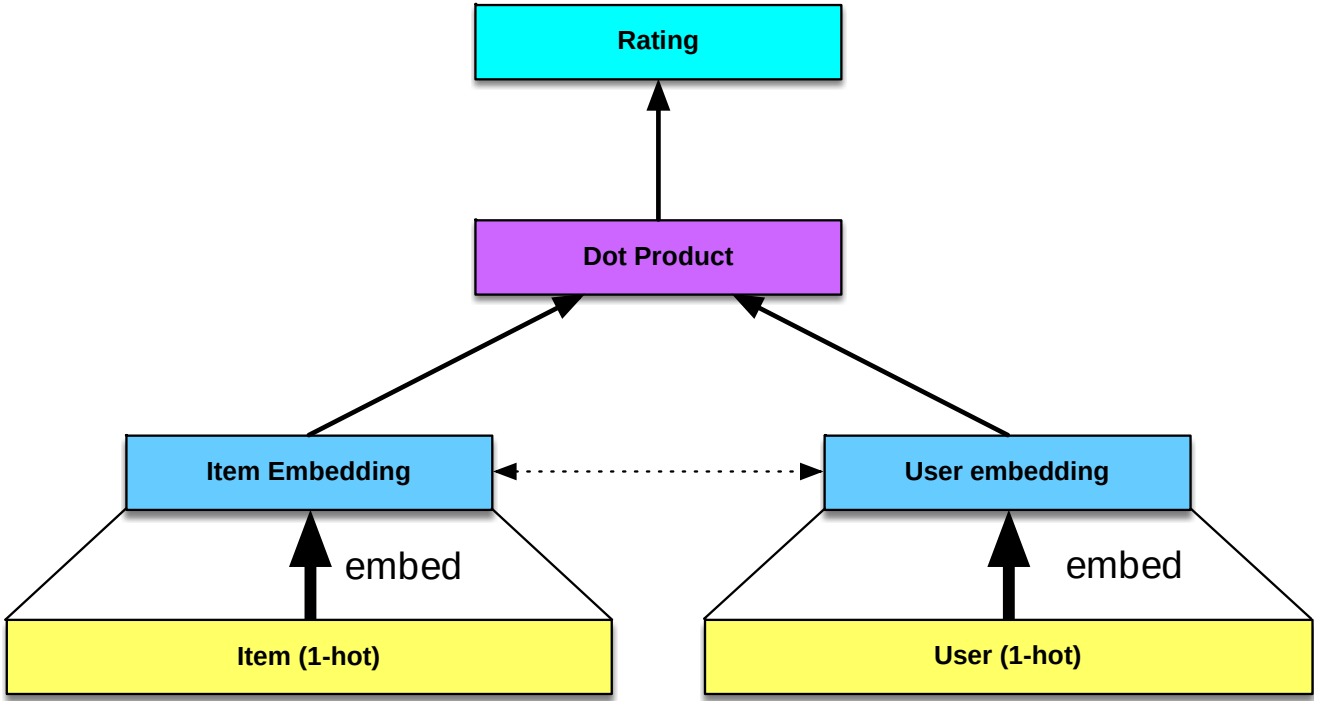
p2.16xlarge



MXNet scaling



MF as a neural network (NN)



Deep Learning AMI with p2

Pre-installed:

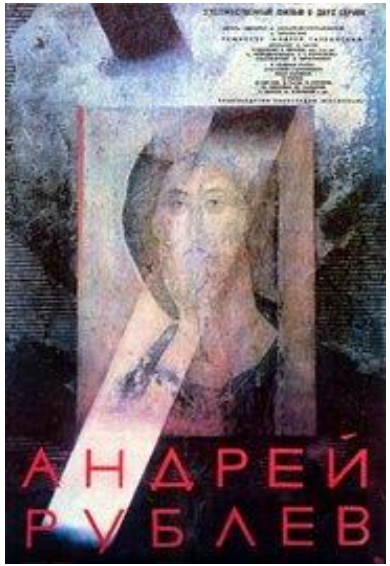
- MXNet & other popular deep learning frameworks
- GPU Drivers, CUDA, cuDNN
- Jupyter notebook & python libraries

MF Demo in MXNet

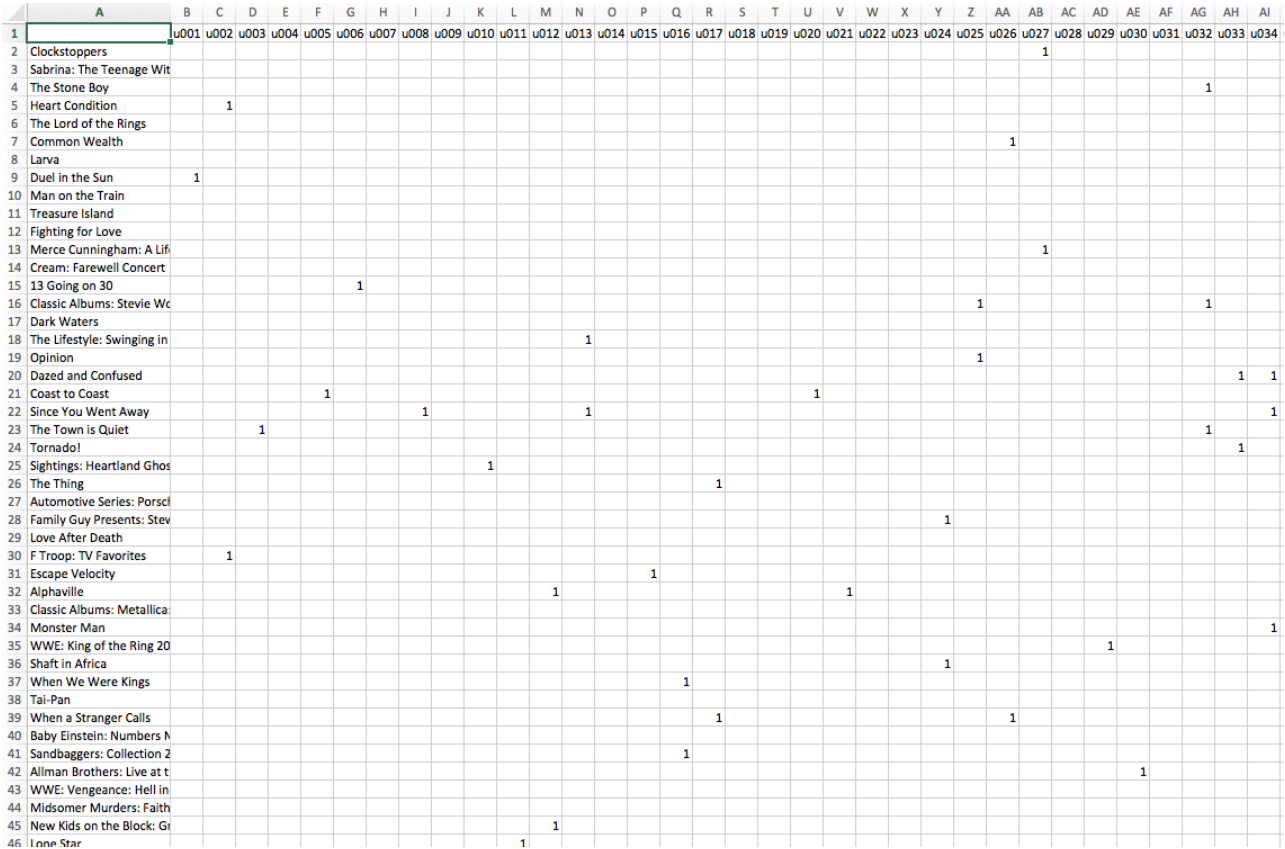
demo1-MF.ipynb

Binary Predictions

Why binary?



Binary user-item matrix



	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	AA	AB	AC	AD	AE	AF	AG	AH	AI
1		u001	u002	u003	u004	u005	u006	u007	u008	u009	u010	u011	u012	u013	u014	u015	u016	u017	u018	u019	u020	u021	u022	u023	u024	u025	u026	u027	u028	u029	u030	u031	u032	u033	u034
2	Clockstoppers																											1							
3	Sabrina: The Teenage Wit																																		
4	The Stone Boy																																1		
5	Heart Condition		1																																
6	The Lord of the Rings																																		
7	Common Wealth																										1								
8	Larva																																		
9	Duel in the Sun	1																																	
10	Man on the Train																																		
11	Treasure Island																																		
12	Fighting for Love																																		
13	Merce Cunningham: A Lif																											1							
14	Cream: Farewell Concert																																		
15	13 Going on 30						1																												
16	Classic Albums: Stevie Wo																									1							1		
17	Dark Waters																																		
18	The Lifestyle: Swinging in													1																					
19	Opinion																									1									
20	Dazed and Confused																																	1	1
21	Coast to Coast					1															1														
22	Since You Went Away								1					1																					1
23	The Town is Quiet			1																													1		
24	Tornado!																																	1	
25	Sightings: Heartland Ghos										1																								
26	The Thing																	1																	
27	Automotive Series: Porscl																																		
28	Family Guy Presents: Stev																								1										
29	Love After Death																																		
30	F Troop: TV Favorites		1																																
31	Escape Velocity															1																			
32	Alphaville												1									1													
33	Classic Albums: Metallica																																		
34	Monster Man																																		1
35	WWE: King of the Ring 20																														1				
36	Shaft in Africa																								1										
37	When We Were Kings																1																		
38	Tai-Pan																																		
39	When a Stranger Calls																	1									1								
40	Baby Einstein: Numbers N																																		
41	Sandbaggers: Collection 2																1																		
42	Allman Brothers: Live at t																														1				
43	WWE: Vengeance: Hell in																																		
44	Midsomer Murders: Faith																																		
45	New Kids on the Block: G												1																						
46	Lone Star											1																							

Original data

	A	B	C
1	**User**	**Movie**	**Label**
2	Leo	Xanadu	1
3	Leo	Caddy Shack	1
4	Petra	Dora the Explorer	1
5	Petra	My Little Pony	1

Predicting binary

```
float predictScore(User u, Movie m) {
    return 1.0;
}
```

Original data

	A	B	C
1	**User**	**Movie**	**Label**
2	Leo	Xanadu	1
3	Leo	Caddy Shack	1
4	Petra	Dora the Explorer	1
5	Petra	My Little Pony	1

Negative sampling

	A	B	C
1	**User**	**Movie**	**Label**
2	Leo	Xanadu	1
3	Leo	Caddy Shack	1
4	Petra	Dora the Explorer	1
5	Petra	My Little Pony	1
6	Leo	Dora the Explorer	0
7	Leo	My Little Pony	0
8	Petra	Xanadu	0
9	Petra	Caddy Shack	0

Negative sampling

```
from mxreco import NegativeSamplingDataIter
train_data = NegativeSamplingDataIter(
        train_data,
        sample_ratio=5)
```

More details: BlackOut: Speeding up RNNLM w/ Very Large Vocabularies
Shihao Ji, S. V. N. Vishwanathan, Nadathur Satish, Michael J. Anderson, Pradeep Dubey

Negative Sampling Demo

demo2-binary.ipynb

Content Features

What do we know?

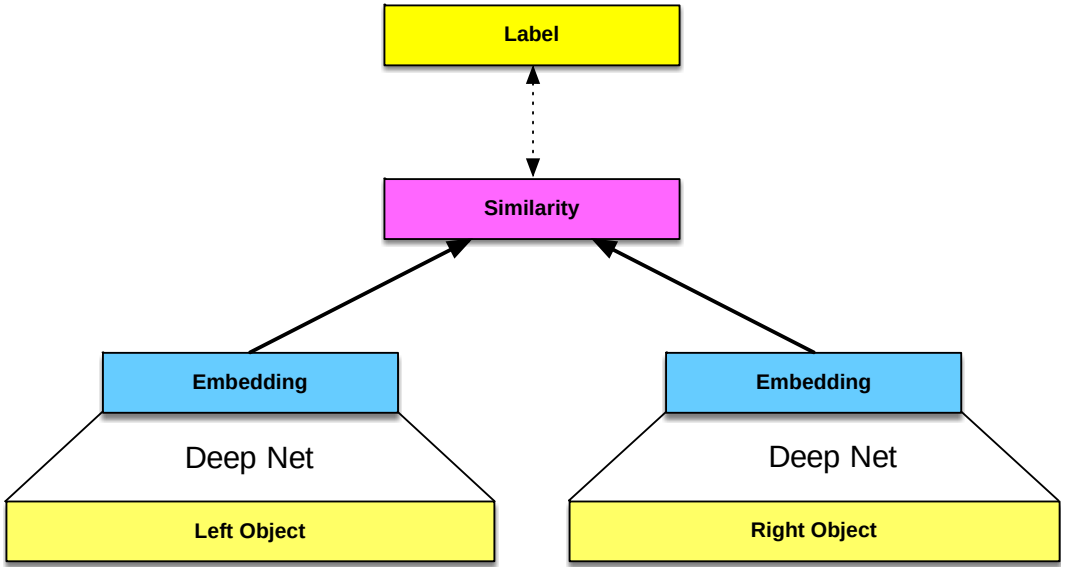
Behavioral interactions between users & items
Names of items
Pictures of items
What users searched for

How to represent these in NN?

Unique Identifier: Embedding
Images: ConvNet (a.k.a. CNN)
Text: LSTM
Text: Bag of Words

DSSM

Deep Structured Semantic Model

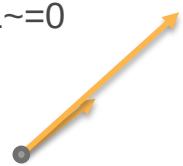


CosineLoss layer

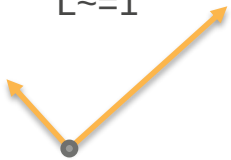
```
import mxreco
pred = mxreco.CosineLoss(a=user, b=item,
                         label=label)
```

$$\mathcal{L}(\mathbf{A}, \mathbf{B}) = 1 - \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

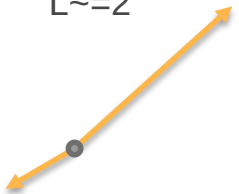
L~=0



L~=1



L~=2



Content Features DSSM Demo

demo3-dssm.ipynb

Inspirational References

Learning Deep Structured Semantic Models for Web Search using Clickthrough Data

- Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, Larry Heck, October, 2013

Deep Neural Networks for YouTube Recommendations

- Paul Covington and Jay Adams and Emre Sargin, 2016

Order-Embeddings of Images and Language

- Ivan Vendrov, Ryan Kiros, Sanja Fidler, Raquel Urtasun, March 2016

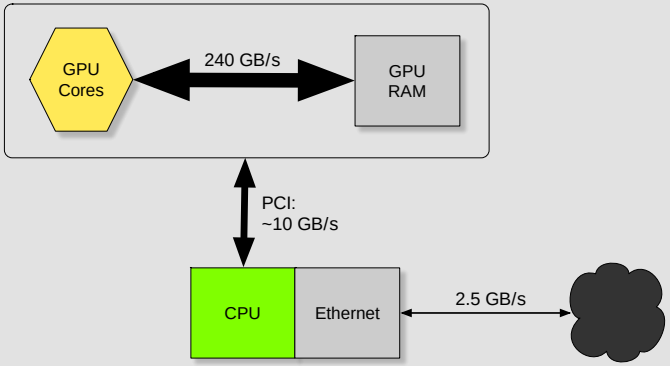
User-Level Models

Predicting with embeddings

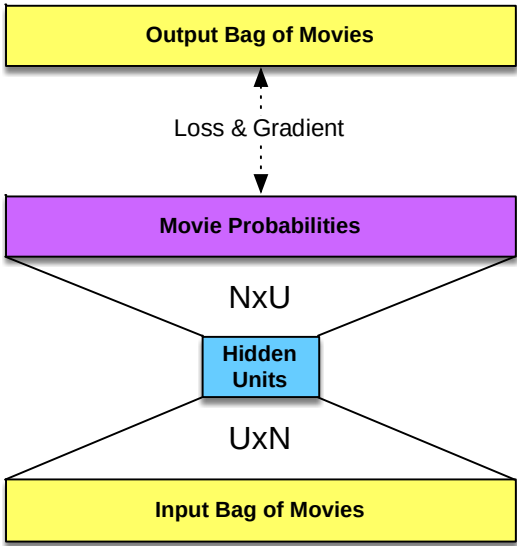
```
def movies_for_user(u):
  scores = {}
  for m in movies:
        score[m.id] = predictScore(u,m)
  top_movies = sorted(scores.items()…)
  return top_moves
```

All content at once

```
def movies_for_user(u):
  scores = userModel.predict(u)
  top_movies = sorted(scores.items()…)
  return top_movies
```



Multi-label neural network



Sparse output

Sparse input

Storing indexes

Conceptually:

- Predict: 1882, 2808, 24, 160, 1831, 2668
- Inputs: 2986, 329, 2012, 442, 512, 1544, 2615, 1037, 1876, 1917, 2532, 196, 1375, 1779, 2054, 2530, 2628, 1909, 2407, 316, 1356, 1603, 2046, 2428

Storing sparse data

Simpler if fixed width

Pad to end with “-1”

- Predict: 1882, 2808, 24, 160, 1831, 2668,-1,-1,-1,-1
- Inputs: 2986, 329, 2012, 442, 512, 1544, 2615, 1037, 1876, 1917, 2532, 196, 1375, 1779, 2054, 2530, 2628, 1909, 2407, 316, 1356, 1603, 2046, 2428,-1,-1,-1,-1,-1,-1,-1,-1

Trying It Yourself

Trying it yourself

Launch Deep Learning AMI

https://aws.amazon.com/marketplace/pp/B01M0AXXQB

Try examples in

https://github.com/dmlc/mxnet/example/recommender

Thank you!

Remember to complete your evaluations!