UNIVERSITY OF OREGON

Tensor Decomposition for Topic Modeling in AI

Jee Choi
Dept. of Computer and Information Science, University of Oregon
Artificial Intelligence Research (AIR) Conference, UCLA
February 7th, 2020

My research intersects HPC and data analytics

- High Performance Computing (HPC)
  - How to write fast program/code
- Why?
  - Saves time and energy
- How
  - Parallelism
  - Data locality
  - Specialization

My research intersects HPC and data analytics

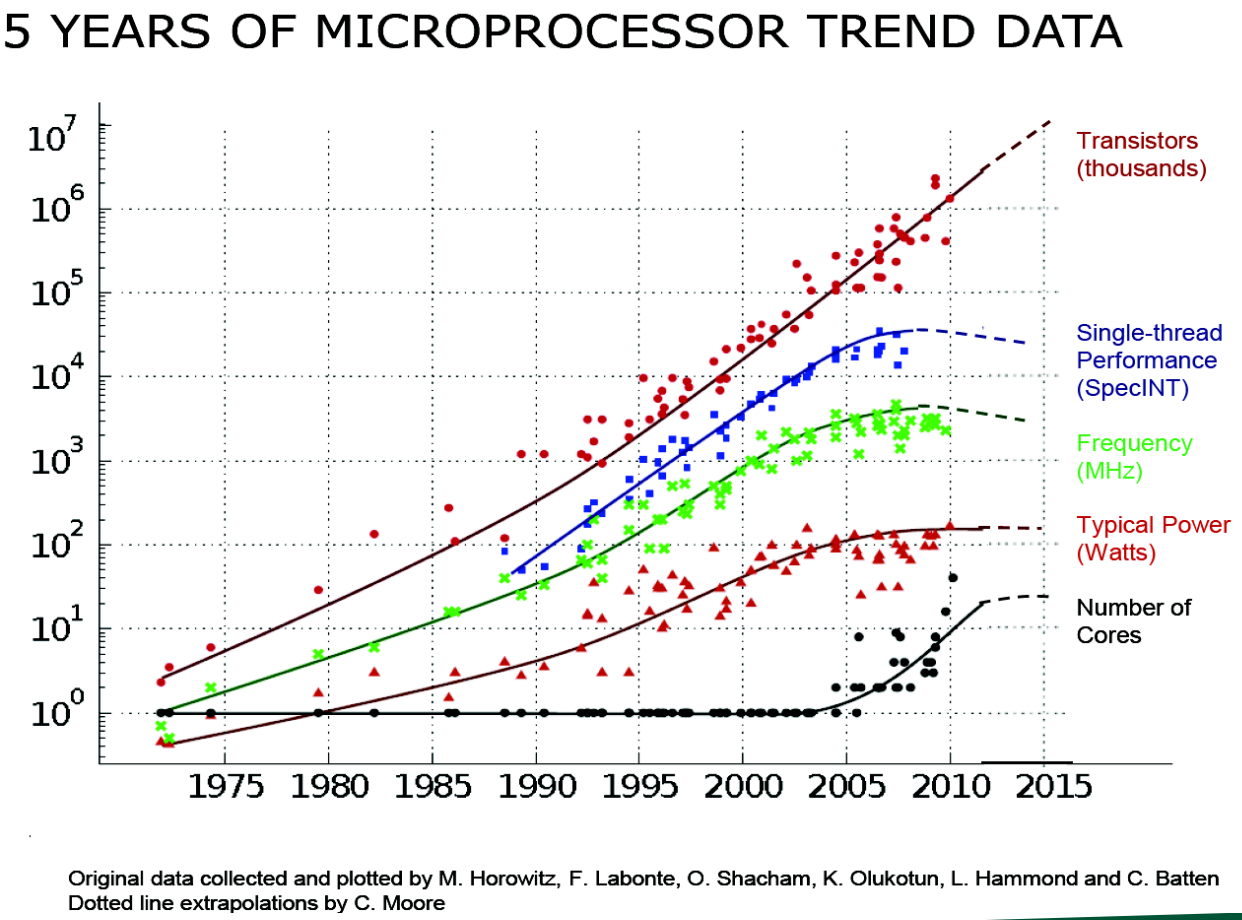
- Data analytics through tensor decomposition
  - Tensors are higher-order (dimensional) generalization of matrices
- Why
  - Many real-world data have n-way relationship
  - E.g., Electronic health record (EHR), product recommendation, network analysis
- How
  - Canonical polyadic
  - Tucker

We generate 2.5 quintillion bytes of data each day

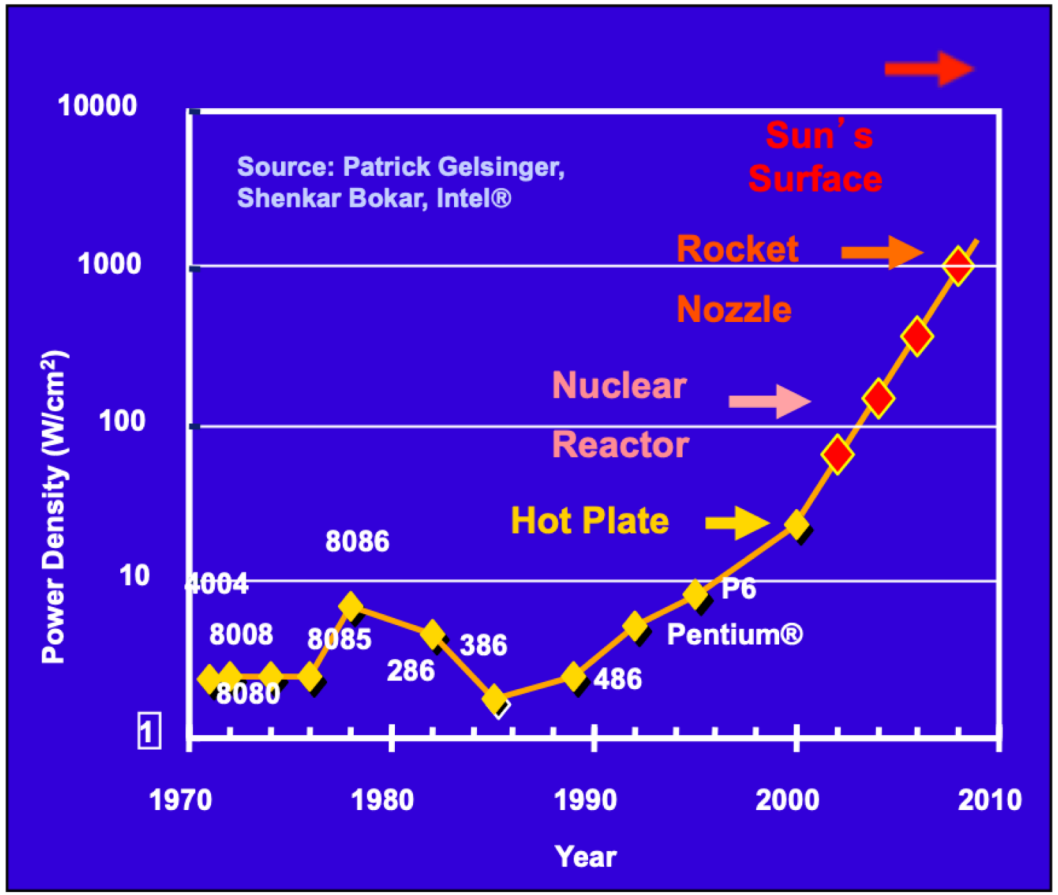
- 2.5×10^{18} bytes
- Every minute
  - 350,000 tweets
  - 4.2 million posts are liked
  - 300 hours of video are uploaded
- 90% of world's data has been created in the past 2 years

https://www.ibm.com/blogs/watson/2016/06/welcome-to-the-world-of-a-i/

Transistors continue to scale, but serial performance stopped in the early 2000s



How power would have scaled with transistor and frequency



Dennard Scaling

- Robert N. Dennard (IBM) laid down the basic “recipe” for for technology scaling in the early 1970s.
- Transistor scale down by 30% (0.7x) every two years ->
  - area scales down by 0.7x0.7 = 0.49x (i.e., half) ->
  - transistor density doubles (Moore’s Law)
- Delay also reduces by 30% (0.7x) ->
  - operating frequency increases by 1.4x (1/0.7x)
- To keep the electric field constant to maintain reliability, supply voltage is reduced by 30% (E = V/d) ->
  - Capacitance reduces by 30% (WL/t_{ox} = 0.7x) ->
  - power reduced by 50% ($C'V'^2f'$ = 0.7C * $(0.7V)^2$ * (1/0.7)f = 0.5x)
  - energy reduced by 65% (E’ = P’ * t’ = 0.5P * 0.7t = 0.35 E)
- Power density (area/power = 0.5/0.5 = 1) remains constant

In summary, as transistors get smaller, both voltage and current scales down to maintain a **constant power density** (i.e., power per area).

Parallelism is ubiquitous

- ***Summit Supercomputer (Department of Energy)***
- **4608x** POWER9 nodes
  - **2x** POWER9 CPUs @ **4 GHz**
    - **20x** Cores per CPU
  - **6x** Nvidia Tesla V100 GPUs @ **1.53 GHz**
    - **5120x** CUDA cores per GPU
- **13 MWatts**

Parallelism is ubiquitous

- ***Intel Core i7-9700k***
- **8x** Cores @ **4.90 GHz**
- **1x** Intel UHD Graphics 630 @ **1.20 GHz**
  - **24x** execution units
- **95 Watts**

Parallelism is ubiquitous

- ***Qualcomm Snapdragon 855 Processor***
- **4x** Kyro 485 high-efficiency cores @ **1.8 GHz**
- **3x** Kyro 485 high-performance cores @ **2.42 GHz**
- **1x** Kyro 485 high-performance cores @ **2.84 GHz**
- **1x** Adreno 640 GPU @ **600 MHz**
  - **384x** ALU
- **~10 Watts**

Specialization

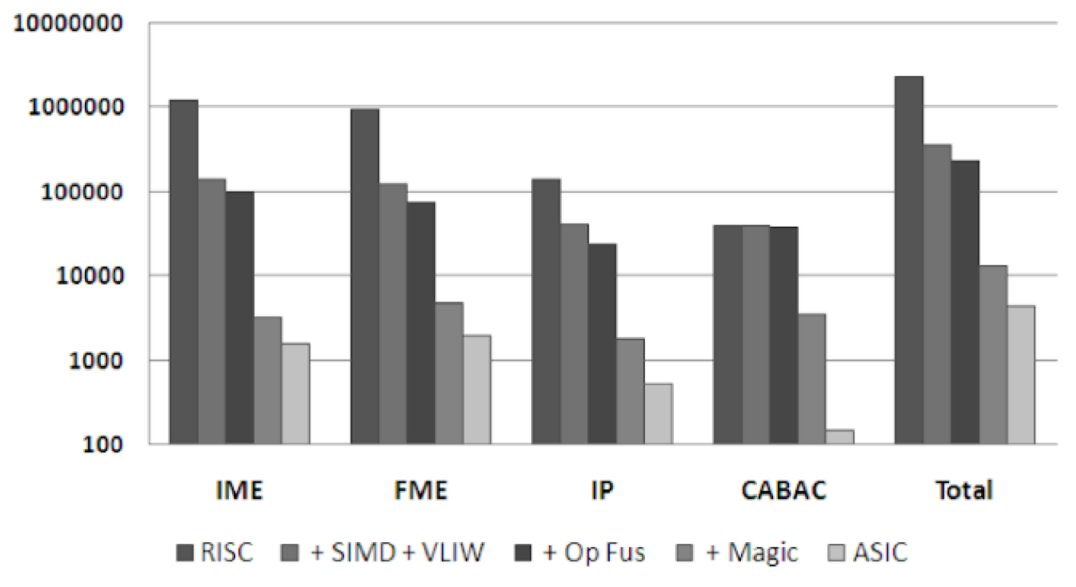


Figure 2. Each set of bar graphs represents energy consumption (µJ) at each stage of optimization for IME, FME, IP and CABAC respectively. Each optimization builds on the ones in the previous stage with the first bar in each set representing RISC energy dissipation followed by generic optimizations such as SIMD and VLIW, operation fusion and ending with "magic" instructions

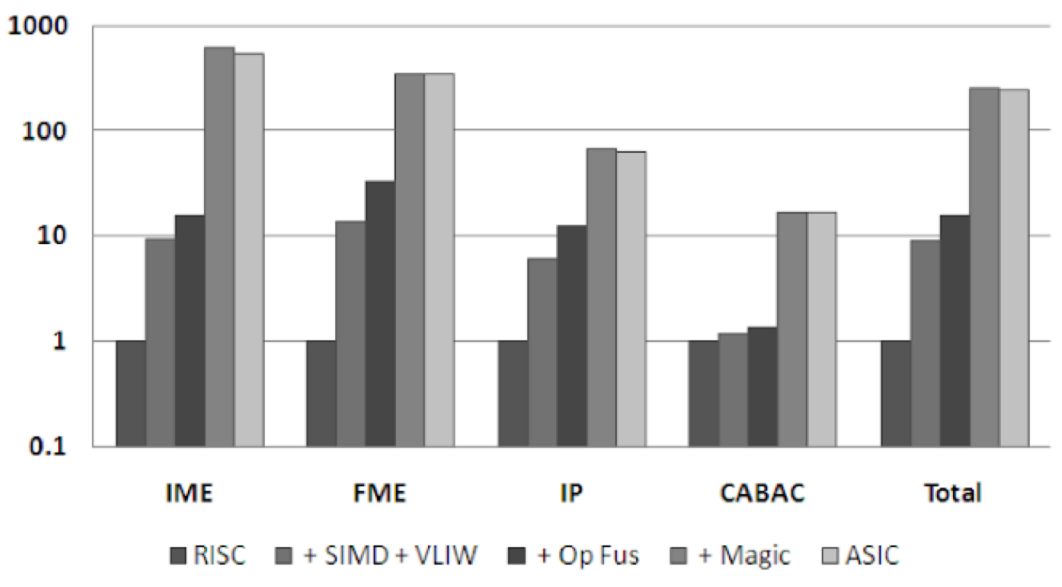


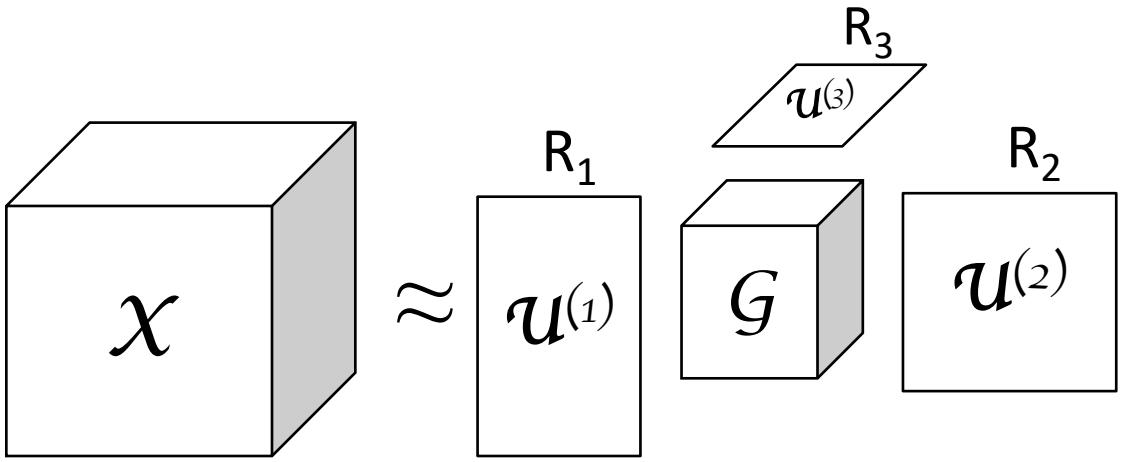
Figure 3. Each set of bar graphs represents speedup at each stage of optimization. Each optimization builds on those of the previous stage with the first bar in each set representing RISC speedup, followed by generic optimizations such as SIMD and VLIW, then operation fusion and finally "magic" instructions

occurring complex instruction subgraphs. Operation fusion is

Conclusion?

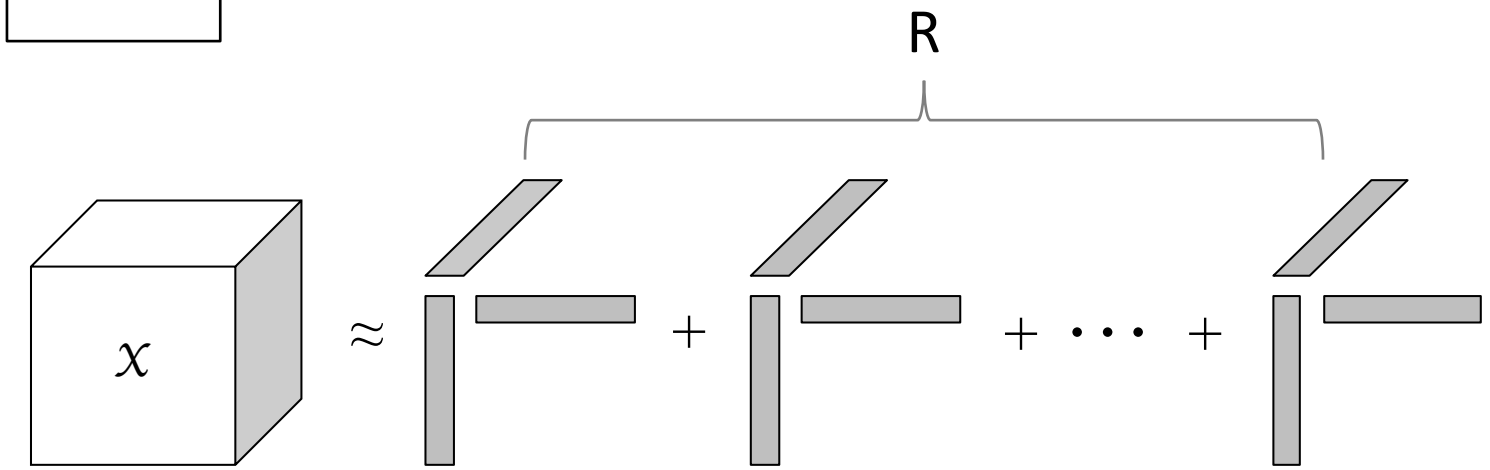
- ~~Winter~~ Parallelism is coming (or is already here)
- We need parallelism to process the massive amount of data being generated each day
- We must consider performance and energy/power when we implement software

Two popular tensor decomposition algorithms



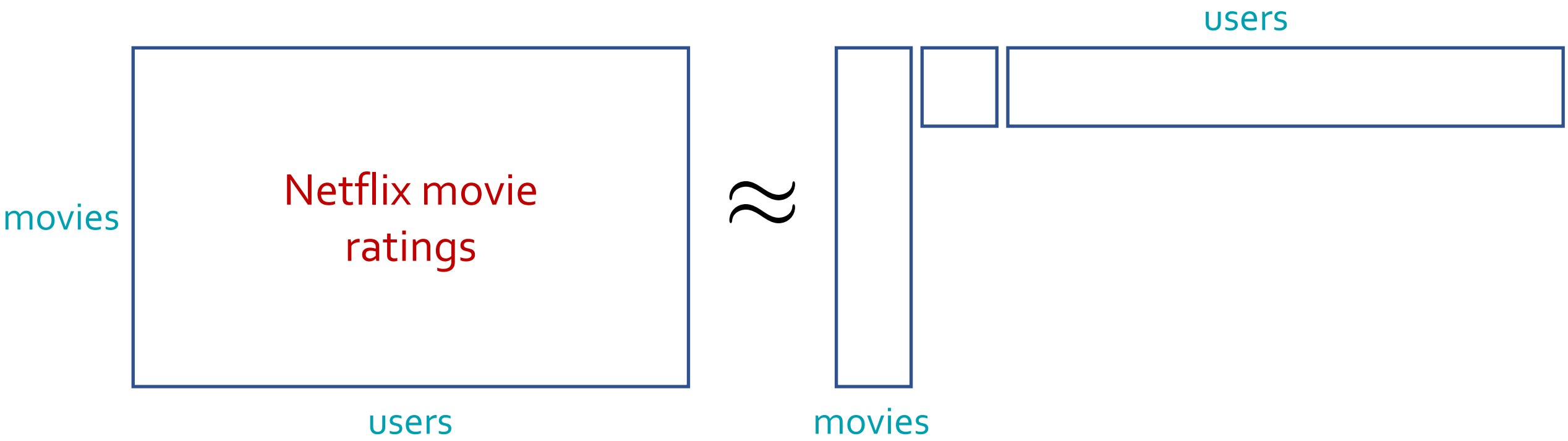
Tucker

Canonical Polyadic (CP)

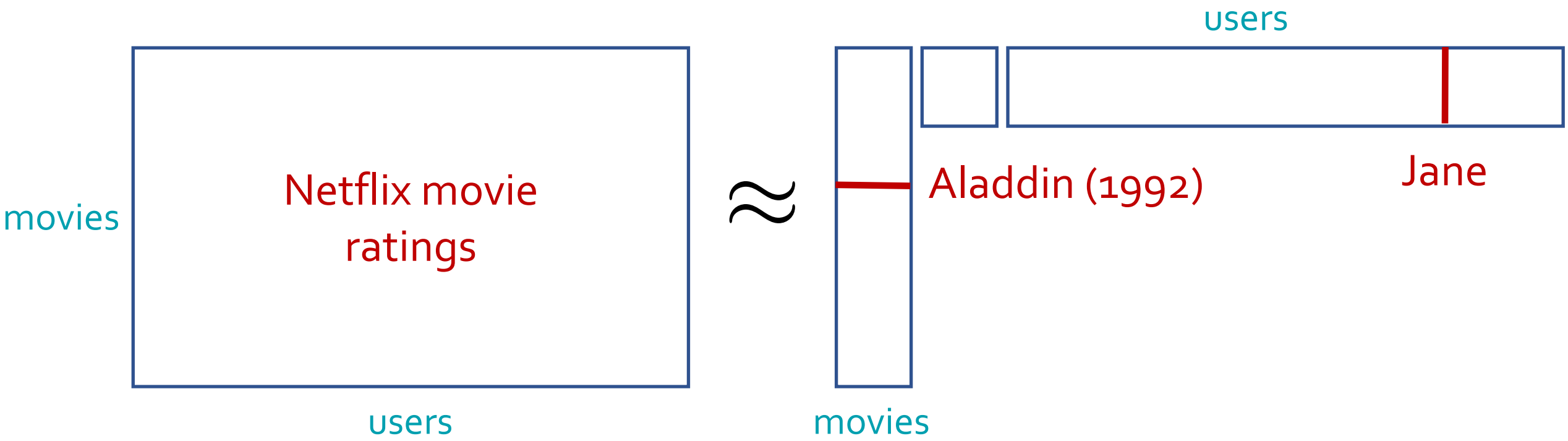


Tensor decomposition is analogous to SVD

Tensor decomposition is analogous to SVD

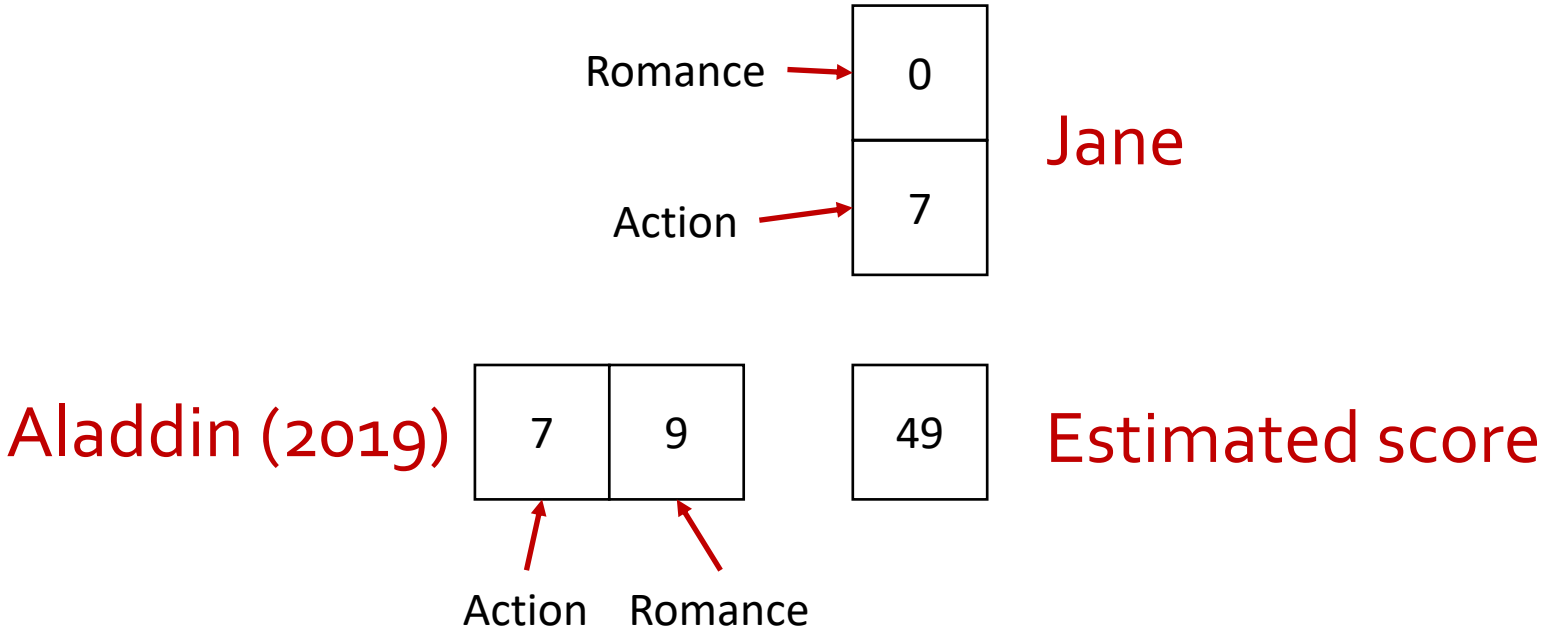


Tensor decomposition is analogous to SVD



Estimating the "score" is as simple as taking the dot product

- Let's say movies only have two "latent" properties – action and romance



Applications of tensor decomposition

- Signal processing
  - Signal separation, code division
- Data analysis
  - Phenotyping (electronic health record), network analysis, data compression
- Machine learning
  - **Latent variable model** (natural language processing, topic modeling, recommender systems, etc.)
  - Neural network compression

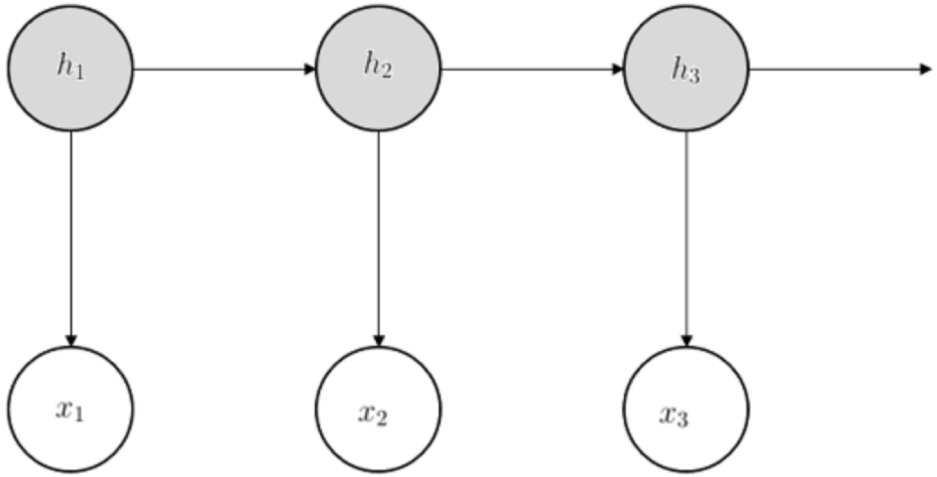
Why tensor decomposition?

- Pros
  - Matrix factorization is **not** unique whereas tensor decomposition is unique (given some conditions)
  - Retains the multi-way relationship that is typically lost when formulated as a matrix problem
- Cons
  - Determining the exact rank is NP-hard
  - Thinking in higher dimensions is difficult

Tensor decomposition for topic modeling

- Topic modeling
  - Model for discovering abstract topics in a collection of documents
  - Hidden Markov Model
    - Each word (in the document) has a hidden topic *h* (e.g., specifies whether the current word is talking about “sports” or “politics.”)
    - Topic for the next word **only** depends on the topic of the current word
    - Each topic specifies a distribution over words – instead of the topic itself, we observe a random word *x*, drawn from this topic distribution (e.g., if the topic is “sports” we will more likely see the word “score.”)

Tensor decomposition for topic modeling



- To generate a sentence in HMM, we start with some initial topic h_1, and this topic will evolve as a Markov Chain to generate the topics for future words (h_2, h_3, etc.)
- We observe words x_1, x_2, etc. from these topics

Tensor decomposition for topic modeling

- Given many sentences that follow this model, we can construct a tensor using correlations
- For every triplet of words (i, j, k), we count the number of times that these three words are the first three words of a sentence
- Enumerating over (i, j, k) gives us a three dimensional tensor T
- Entry (i, j, k) in T gives us the probability that the first three words are (i, j, k)

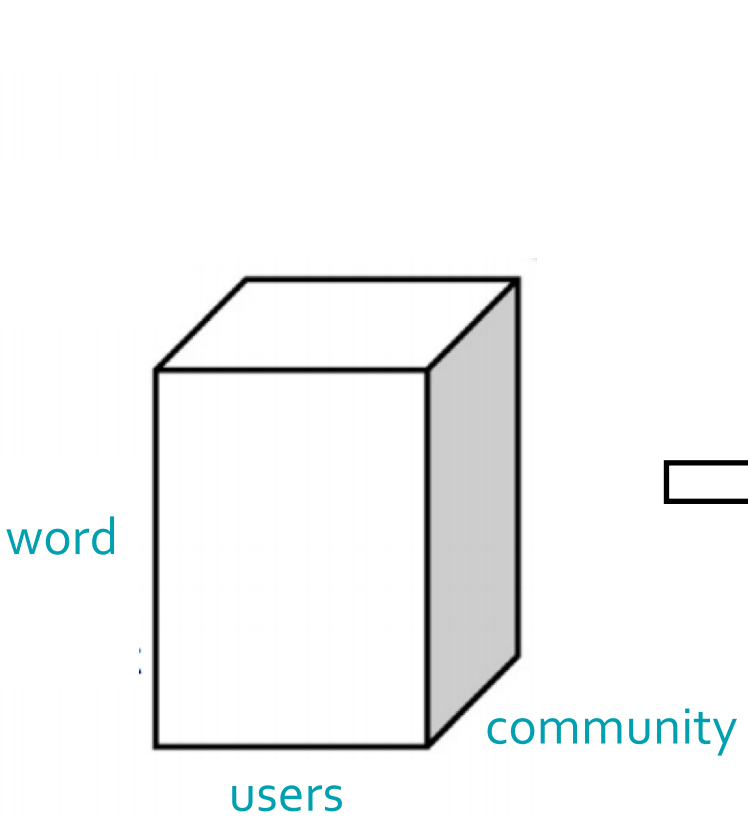
Tensor decomposition for topic modeling

- If we fix the topic of the second word (h_2), the tensor is cut into three parts – h_1/x_1, x_2, and h_3/x_3
- x_1, x_2, x_3 are independent conditioned on the topic h_2
- If we decompose this tensor, then we have vector x_L, whose i^{th} entry is the probability the first word is i, given the topic of the second word is L
- We can compute a rank n decomposition where we estimate the n most prominent topics of the document

Tensor decomposition for topic modeling II

- We can also apply bag-of-words to determine topics (LSA – latent semantic analysis)
  - Consider each document simply as a collection of words
- Build a co-occurrence matrix (or tensor), decompose it, and apply a clustering algorithm
  - I x J x K matrix where I is the collection of words, and J is the collection of documents, K is the location of the word in the document
  - Each ”rank” represents a prominent topic within the documents

Plot the rows in R-dimensional space



R factors (columns)

word factor

user factor

community factor

Each **row**

point in R-dimensional Euclidean space

K-nearest neighbor

Words by political association

- **Democrat**
- **Republican**

Words by political association

- **Democrat**
  - parti
  - presid
  - elect
  - bill
  - congress
  - senat
  - mccain
  - campaign
  - candid
  - vote
  - Hillary
  - conserv
  - constitut
  - ron
  - voter
  - palin
  - right
  - liber
  - he
- **Republican**
  - rightard
  - gop
  - traitor
  - honorless
  - bush
  - lie
  - wing
  - scumbag
  - f*cktard
  - moron
  - pretend
  - parti
  - stupid
  - dishonest
  - rightw
  - a*shol
  - fox
  - right
  - democrat

Words by political association

- **Democrat**
  - parti
  - presid
  - elect
  - bill
  - congress
  - senat
  - mccain
  - campaign
  - candid
  - vote
  - Hillary
  - conserv
  - constitut
  - ron
  - voter
  - palin
  - right
  - liber
  - he
- **Republican**
  - **rightard**
  - gop
  - traitor
  - honorless
  - bush
  - lie
  - wing
  - scumbag
  - **f*cktard**
  - **moron**
  - pretend
  - parti
  - **stupid**
  - dishonest
  - rightw
  - **a*shol**
  - **fox**
  - right
  - democrat

Topic Modeling for Detecting Malware

- 3-D tensor of
  - Features (e.g., permission, hardware components, sensitive API calls) x File x Location -> how many times it occurs
  - With rank-N decomposition, the N features are used as input to SVM
    - Top 32 features offers excellent variance and a median accuracy of
      - **88.75% With Rank 3**
      - 88.5% With Rank 5
      - 89.25% With Rank 10
      - 89.5% With Rank 50
      - 89.5% With Rank 100
      - 89.5% With Rank 200
      - 89.5% With Rank 500

Topic Modeling for Detecting Malware

Feature Set	Benign	Malware	% Benign	% Malware
1	1	7	13%	88%
2	1591	189	89%	11%
3	0	4	0%	100%
4	0	8	0%	100%
5	14	21	39%	58%
6	3	11	21%	79%
7	19	64	23%	77%
8	0	13	0%	100%
9	0	4	0%	100%
10	39	64	38%	62%
11	30	20	60%	40%
12	2	6	25%	75%
13	0	14	0%	100%

Performance is better using tensor decomposition

	Execution Time (s)		
	Tensor Decomposition	Training	Total
Baseline (SVM)	0	6.91	6.91
Rank 3 CPD	0.1	0.02	0.12
Rank 5 CPD	0.21	0.02	0.23
Rank 10 CPD	0.33	0.01	0.34
Rank 50 CPD	0.82	0.01	0.83
Rank 100 CPD	2.1	0.03	2.13
Rank 200 CPD	2.24	0.04	2.28
Rank 500 CPD	5.47	0.09	5.56

Conclusion

- HPC is becoming increasingly more important in HPC
  - Particularly with IoT – certain applications run with limited processing power and strict performance constraints (e.g., self-driving cars)
- Understanding how AI works is important
  - Deep Neural Network does not provide such information
  - Tensor decomposition provides more information (but does not perform as well as DNN)
- Collaboration between wider range of areas will become necessary to push the AI frontier