Distributed Machine Learning

Peacock

Yi Wang

Goal

- Scalability
  - engineering
  - mathematical

Gibbs Sampling

- Random initialization

words

	Bill	Gates							
	Bill	Gates	Microsoft						
			Microsoft	Windows					
text					Steve	Jobs			
					Steve	Jobs	Apple		
							Apple	iPhone	
							Apple		iPad

Gibbs Sampling

- Update iteratively
  - the color of a word tend to be similar to other words in the same document.
  - the color of a word tend to be similar to its major color in the whole corpus.

Gibbs Sampling

- After convergence

words

	Bill	Gates							
	Bill	Gates	Microsoft						
			Microsoft	Windows					
text					Steve	Jobs			
					Steve	Jobs	Apple		
							Apple	iPhone	
							Apple		iPad

Sufficient Statistics

- Random initialization

Bill	Gates							
Bill	Gates	Microsoft						
		Microsoft	Windows					
				Steve	Jobs			
				Steve	Jobs	Apple		
						Apple	iPhone	
						Apple		iPad

1	1
1	2
1	1
1	1
1	2
1	1
1	1

1	1	1		1	1	1		1
1	1	1	1	1	1	2	1	

Sufficient Statistics

- After convergence

Bill	Gates							
Bill	Gates	Microsoft						
		Microsoft	Windows					
				Steve	Jobs			
				Steve	Jobs	Apple		
						Apple	iPhone	
						Apple		iPad

2	
3	
2	
	2
	3
	2
	2

2	2	2	1					
				2	2	3	1	1

Many Documents

- Distribute by documents, duplicate model

Bill	Gates								2	
Bill	Gates	Microsoft							3	
		Microsoft	Windows						2	
				Steve	Jobs					2
				Steve	Jobs	Apple				3
						Apple	iPhone			2
						Apple		iPad		2

2	2	2	1					
				2	2	3	1	1

Many Documents

- Distribute by documents, duplicate model
    - n computers, each handle some documents and related statistics.
    - the model is duplicated on each computer.
    - computers sync up changes they made to local models.

Many Documents

- Sync-up local models
  - Synchronous can be done with MapReduce.
    - http://www.datalab.uci.edu/papers/distributed_topic_modeling.pdf
    - http://link.springer.com/chapter/10.1007/978-3-642-02158-9_26
  - Asynchronous sync-up needs self-made frameworks.
    - http://papers.nips.cc/paper/3524-asynchronous-distributed-learning-of-topic-models

Many Tokens

- Distribute by tokens.

Bill	Gates							
Bill	Gates	Microsoft						
		Microsoft	Windows					
				Steve	Jobs			
				Steve	Jobs	Apple		
						Apple	iPhone	
						Apple		iPad

2	
3	
2	
	2
	3
	2
	2

2	2	2	1					
				2	2	3	1	1

Many Tokens

- Distribute by tokens, duplicate topic distributions.
  - n computers, each handle some tokens, and maintains part of the model.
  - the topic distributions is duplicated on each computer.
  - computers sync up changes they made to topic distributions.

Many Documents and Tokens

- Distribute by documents and tokens.

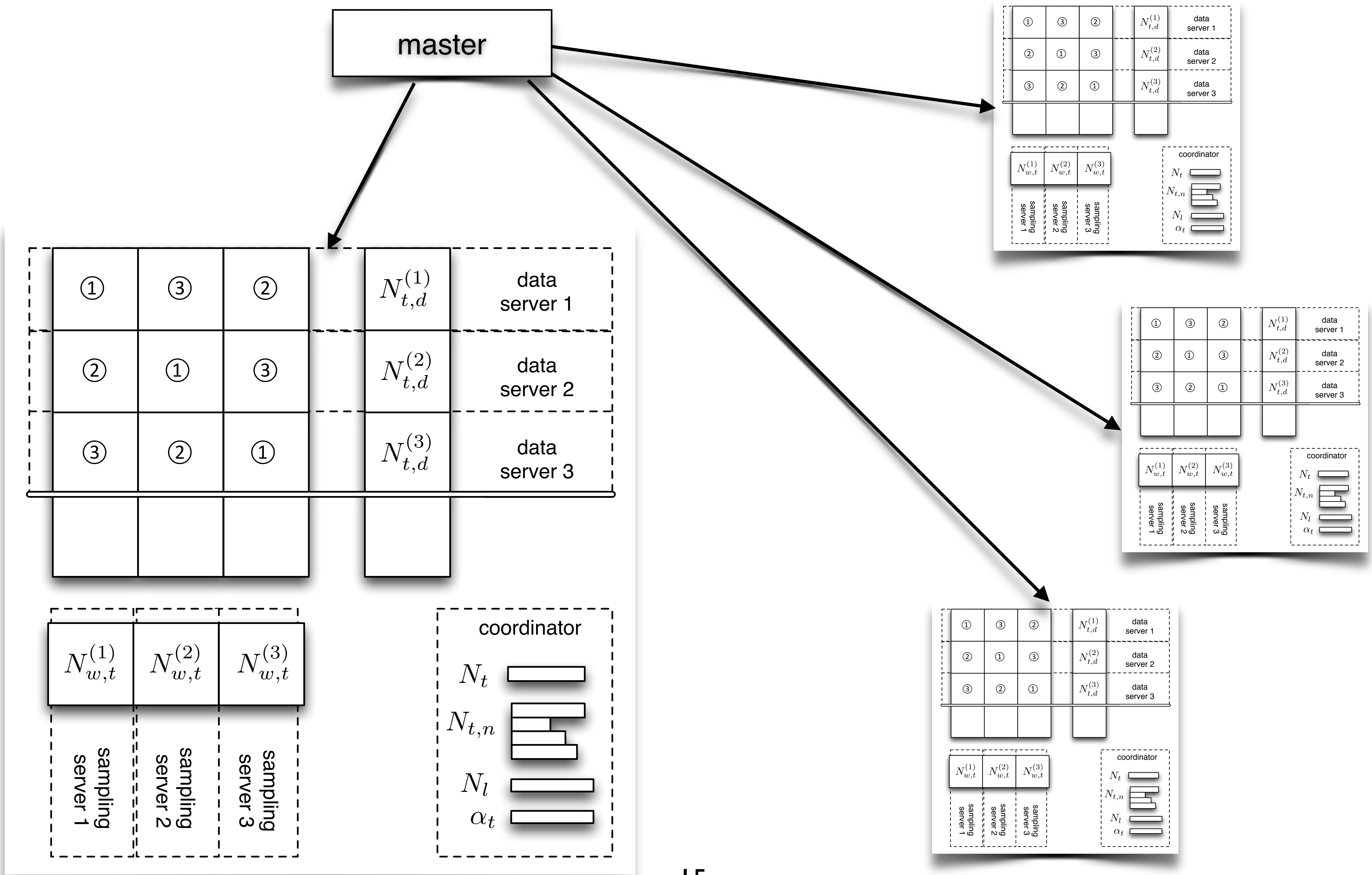
Bill	Gates								2	
Bill	Gates	Microsoft							3	
		Microsoft	Windows						2	
				Steve	Jobs					2
				Steve	Jobs	Apple				3
						Apple	iPhone			2
						Apple		iPad		2

2	2	2	1					
				2	2	3	1	1

Many Documents and Tokens

- Distribute by documents and by tokens
  - n *loaders*, each handle some documents and related statistics.
  - another n *samplers*, each has a partition of the model.
  - An iteration has n steps.
  - In each step, a loader works with a sampler on a *block*.
  - **no model or distribution sync-up required.**

Scalable and Recoverable



Processes

- A *master* process, which controls all groups.
- m groups, each contains
  - a *coordinator* process,
  - n *loaders*, and
  - n *samplers*
- A group of n *aggregators*.

Master

- Ask Kubernetes to start m coordinators,
  - Maintain an *active* queue of M segments,
  - assign each coordinator a segment, and move assigned segments to *pending* queue.
- Waiting for coordinators' calls
  - If a coordinator finishes a segment, move it to *done*.
- watch these coordinators
  - if anyone died, restart it, assign pending segments.

Coordinator

- Ask Kubernetes to start n loaders and n samplers.
- Report to master and accept a segment (task):
    - for step i = 0 … n-1
        - loader x works with sampler (x+i)%n on updating block located at x, (x+i)%4n.
    - each sampler report model updates to corresponding aggregator.
- If restarted, restart samplers and loaders, and samplers load model from aggregators.

Conclusion

- Done:
  - Modeling: asymmetric Dirichlet prior mimics Dirichlet process with huge K.
  - Engineering: asymmetric Dirichlet prior simplifies communication and sync-up so enables our architecture to learn huge K.
  - Automatically estimate K.
- Todo:
  - Extend Peacock to learn a deep hierarchical model.