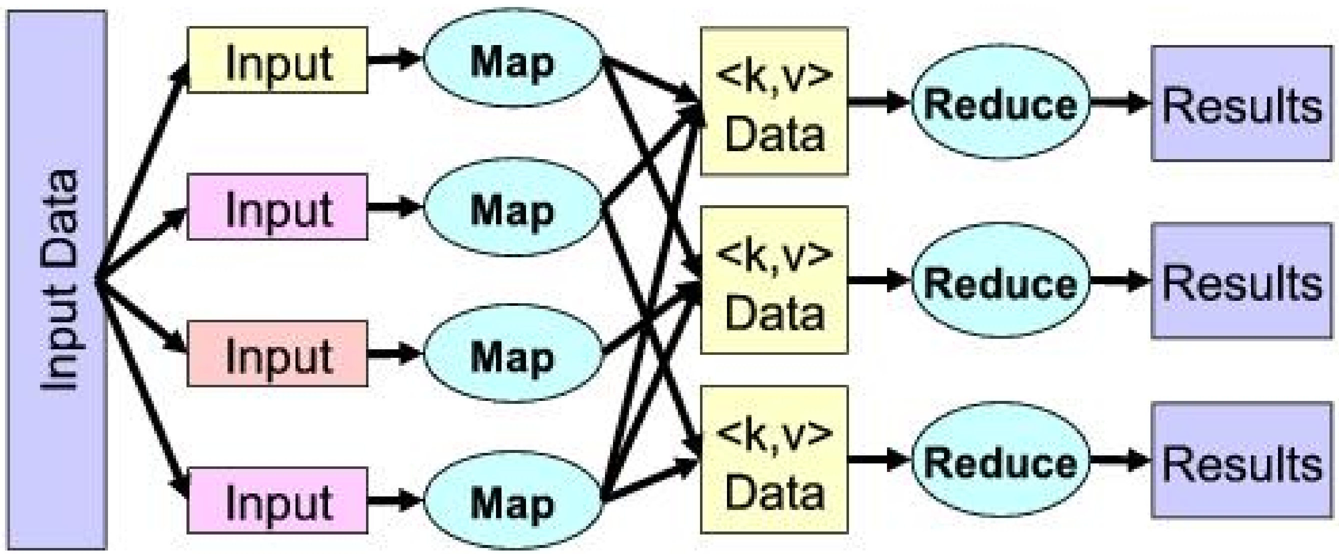
Bayesian ML with MapReduce

Zhuhua Cai
Google, Rice University
caizhua@gmail.com

Outline

- MapReduce.
- Bayesian LR with missing values.
- MapReduce Implementation.

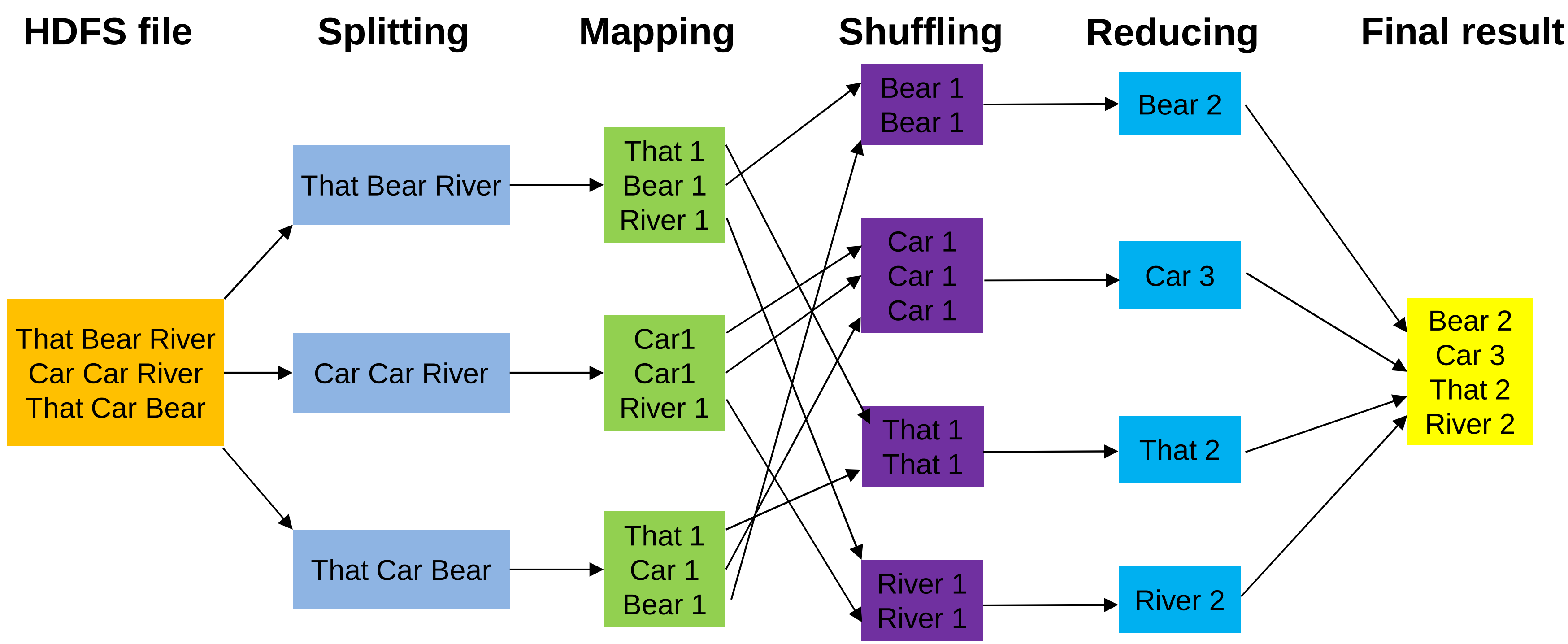
MapReduce



Input Data
Input
Input
Input
Input
Map
Map
Map
Map
<k,v> Data
<k,v> Data
<k,v> Data
Reduce
Reduce
Reduce
Results
Results
Results

Problem 5

- Given 20news-group dataset, for each word, compute its count in the whole dataset.



Code

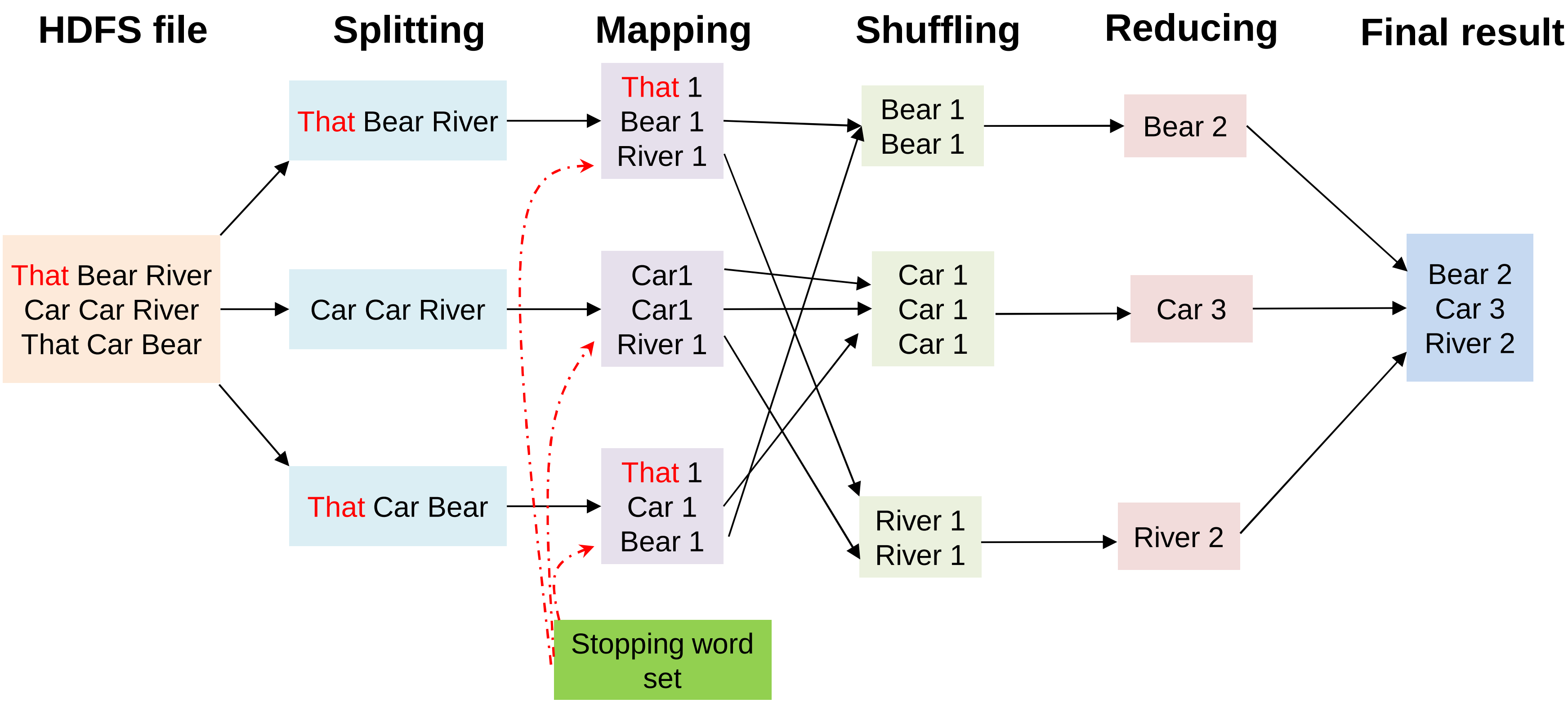
```
public static class Map extends Mapper<LongWritable, Text, Text,
LongWritable> {
  public void map(LongWritable key, Text value, Context context)
    throws IOException, InterruptedException {
   String line = value.toString();
   StringTokenizer tokenizer = new StringTokenizer(line, "\":;, (){}\t\r\n,|");
   while (tokenizer.hasMoreTokens()) {
    String str = tokenizer.nextToken();
    context.write(new Text(str), new LongWritable(1));
   }
  }
 }
```

Code

```
public static class Reduce extends
    Reducer<Text, LongWritable, Text, LongWritable> {
  public void reduce(Text key, Iterator<LongWritable> values, Context context)
      throws IOException, InterruptedException {
    long sum = 0;
    while (values.hasNext()) {
      sum += values.next().get();
    }
    context.write(key, new LongWritable(sum));
  }
}
```

Problem 6

- Given 20news-group dataset and a set of stopping words, find the top 10000 most frequent words.



Code

- WordCount.java

Outline

- MapReduce.
- Bayesian LR with missing values.
- MapReduce Implementation.

Problem 7

- Problem. Given the 20-newsgroup dataset with missing values, we want to create a binary classifier to classify religion and non-religion groups.
- Dataset: http://qwone.com/~jason/20Newsgroups/20news-19997.tar.gz

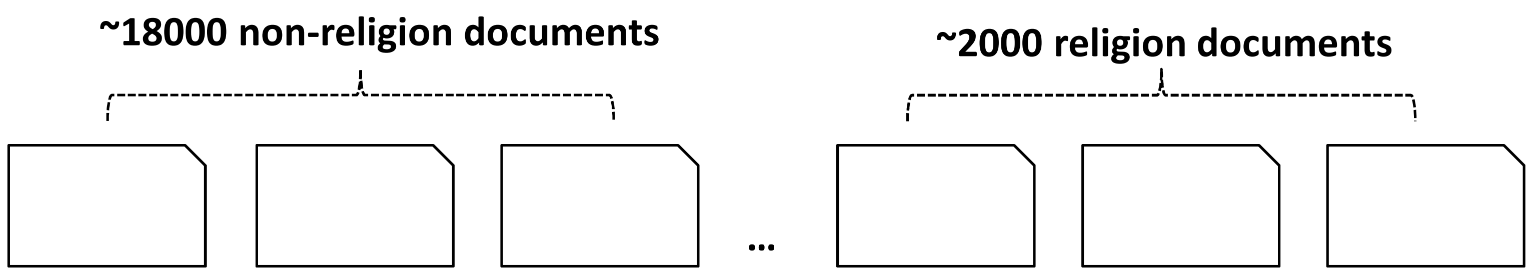


Figure. Data set.

Problem Modeling

training set

doc_0	$[word_0: count_{00}, word_1: ?, word_2: count_{02}, ..., word_m: ?]$	1
doc_1	$[word_0: ?, word_1: count_{11}, word_2: ?, ..., word_n: count_{1m}]$	0
	...	
doc_n	$[word_0: ?, word_1: count_{n1}, word_2: ?, ..., word_n: count_{nm}]$	1

test set

doc_{n+1}	$[word_0: count_{q0}, word_1: ?, word_2: count_{q2}, ..., word_n: ?]$	?
	...	
doc_r	$[word_0: ?, word_1: count_{r1}, word_2: ?, ..., word_n: count_{rm}]$	?

Problem 7

- Method: Bayesian LR + Imputation

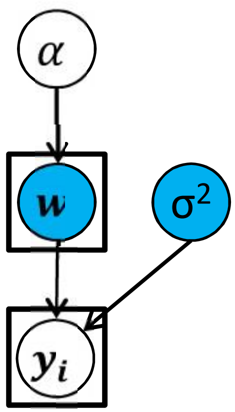


Figure. Graphical model for Bayesian linear regression.

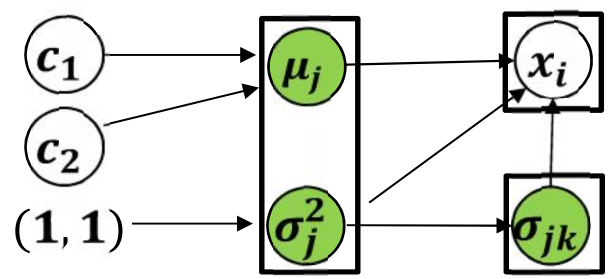


Figure. Graphical model for Markov random field.

Generative Model

- Method: Bayesian LR + Imputation

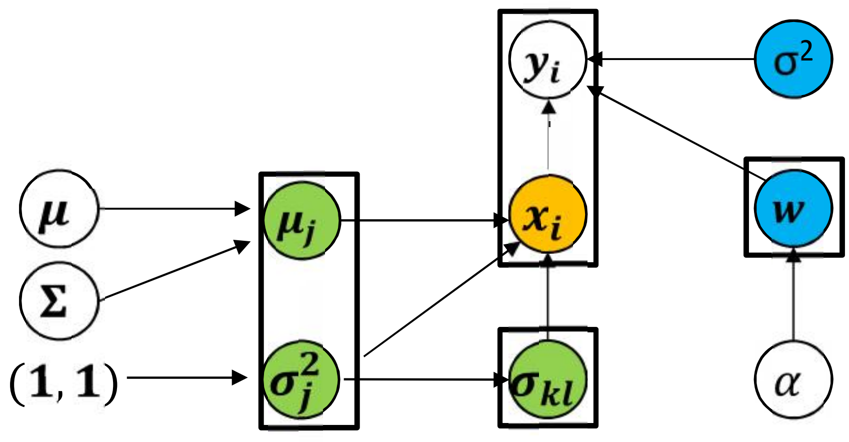


Figure. Graphical model for Markov random field.

Outline

- MapReduce.
- Bayesian LR with missing values.
- MapReduce Implementation.

Inference Via MCMC

- Let us start from Bayesian LR.

$$P(\sigma^2|.) \propto InvGamma(\sigma^2|1+\frac{n}{2}, 1+\sum_i \frac{(y_i - w^T x_i)^2}{2})$$

$$P(w_j|.) \propto N\left(w_j\left|\frac{\sum_i \left(y_i - \sum_{k!=j} w_k x_{ik}\right) x_{ij}}{\sum_i x_{ij}^2}, \frac{\sigma^2}{\sum_i x_{ij}^2}\right.\right) N\left(w_j|0, \alpha^{-1}\right)$$

$where$

$$\sum_i \left(y_i - \sum_{k!=j} w_k x_{ik}\right) x_{ij}$$
$$= \sum_i (y_i - \sum_{k \notin B} w_k x_{ik} - \sum_{k \in B, k!=j} w_k x_{ik}) x_{ij}$$
$$= \sum_i \left(y_i - \sum_{k \notin B} w_k x_{ik}\right) x_{ij} - \sum_{k \in B, k!=j} w_k \sum_i x_{ik} x_{ij}$$

Algorithm for Bayesian LR

1. Initialize w.
2. *For each block* B, compute the set of *inner cross aggregates* $\{\sum_i x_{ij}^2\}_{j \in B}$ and $\{\sum_i 2x_{ij}x_{ik}\}_{j \in B, k \in B, j \leq k}$ by one job.
3. Compute *sigma aggregate* $\sum_i (y_i - w^T x_i)^2$ by one job.
4. *Sample* σ^2.
5. *For each block* B,

   compute the *external aggregates* $\{\sum_i 2\left(y_i - \sum_{k \notin B} w_k x_{ik}\right) x_{ij}\}_{j \in B}$ by one job.

   Sample $\{w_j\}_{j \in B}$.
6. *Go to step 3.*

Code

1. Initialize w. Bayesian_LR_Main.*initializeCoefs(.)*
2. *For each block* B, compute $\{\sum_i x_{ij}^2\}_{j\in B}$ and $\{\sum_i 2x_{ij}x_{ik}\}_{j\in B,k\in B,j\leq k}$. LRInnerAggCollector.java
3. Compute $\sum_i (y_i - w^T x_i)^2$. LRSigmaAggCollector.java
4. *Sample* σ^2. Bayesian_LR_Main. *samplingSigma2(.)*
5. *For each block* B,

   compute $\{\sum_i 2\left(y_i - \sum_{k\notin B} w_k x_{ik}\right)x_{ij}\}_{j\in B}$. LRExternalAggCollector.java

   Sample $\{w_j\}_{j\in B}$. BlockCoefsSampler.java
6. *Go to step 3.*

LRInnerAggCollector.java

mapper 1	8.0	2.0	3.5	4.1	1.0	2.0	3.5	4.7	1.0	2.0	3.5	4.1	1.0	2.0	3.5	4.1
	9.0	2.0	3.5	4.1	1.0	2.0	3.5	4.7	1.0	2.0	3.5	4.1	1.0	7.0	3.5	4.1
mapper 2	8.0	5.0	5.5	4.1	4.1	2.0	3.5	4.7	1.0	2.0	5.5	4.1	1.0	2.0	2.5	4.1
	8.0	2.0	7.5	4.4	1.0	2.0	3.5	4.7	1.0	2.0	3.5	4.1	1.0	2.0	3.5	5.1
mapper 3	8.0	2.0	3.5	4.1	1.0	2.0	3.5	4.7	1.0	3.0	3.5	4.1	1.0	2.0	3.5	4.1
	8.0	2.0	3.5	4.2	1.0	2.0	3.5	4.7	1.0	2.0	4.5	4.7	1.0	5.0	5.5	4.1
mapper 4	8.0	2.0	3.5	4.1	1.0	2.0	3.5	4.7	5.0	2.0	3.7	4.1	1.0	2.0	3.5	4.1
	9.0	5.0	3.5	4.1	1.0	2.0	3.5	4.7	1.0	2.0	3.5	4.1	1.0	2.0	3.5	4.1

$$\{ \sum_i 2x_{i0}x_{i1}, \sum_i 2x_{i0}x_{i2}, \sum_i 2x_{i0}x_{i3}, \sum_i 2x_{i1}x_{i2}, \sum_i 2x_{i1}x_{i3}, \sum_i 2x_{i2}x_{i3} \}$$

$$\{ \sum_i x_{i0}^2, \sum_i x_{i1}^2, \sum_i x_{i2}^2, \sum_i x_{i3}^2 \}$$

LRInnerAggCollector.java

- Key Value Design
  - Mapper input $(long, string)$
  - Shuffle $((int, int), double)$
  - Reduce output $((int, int), double)$
- Procedure
  - Mapper reads a line with $\mathrm{m}+1$ values including y, and output all pair aggregates.
  - Combiner and reducer combine the aggregates.

LRInnerAggCollector.java

- Key Value Design
  - Mapper input $(long, string)$
  - Shuffle $((int, int), double)$
  - Reduce output $((int, int), double)$
- Change
  - Mapper reads a line with $\mathrm{m}+1$ values including y, and save the aggregates. The shuffle key and value will not be written out until the partition data is finished.
  - Change the shuffle key (j, k) into $j * m + k$ to reduce the communication and memory overhead.

Inference Via MCMC

- Let us turn to imputation

$$P(x_{ij}|.) \propto N\left(x_{ij}\left|\frac{y_i-\sum_{k!=j} w_k x_{ik}}{w_j}, \frac{\sigma^2}{w_j^2}\right.\right) \times \begin{cases} N(x_{ij}|\mu_j, \sigma_j^2) \\ \prod_{k|(j,k)\in\Psi} N(x_{ij}|\mu_j - \frac{A_{jk}}{A_{jj}}(x_{ik} - u_k), \frac{1}{A_{jj}}) \end{cases}$$

where

$$\begin{pmatrix} A_{jj} & A_{jk} \\ A_{jk} & A_{kk} \end{pmatrix} = \begin{pmatrix} \sigma_j^2 & \sigma_{jk} \\ \sigma_{jk} & \sigma_j^2 \end{pmatrix}^{-1}$$

Given the model, all the imputation can be done by one job.

Inference Via MCMC

- Sample Markov Random Field
  - Mathematically complicated.

$$P(\mu_j|.) \propto N(\mu_j|\mu,\Sigma) \times \prod_{k|(j,k)\in\Psi} N(\mu_j| \frac{\sum_i x_{ij}}{n} + \frac{A_{jk}}{A_{jj}}\left(\frac{\sum_i x_{ik}}{n} - \mu_k\right), \frac{1}{nA_{jj}})$$

$$P(\sigma_j^2|.) \propto InvG(\sigma_j^2|1,1) \times \prod_{k|(j,k)\in\Psi} f(\Delta_{jk}) \times \left(\frac{1}{|S|}\sum_{s\in S} \frac{1}{\left(\prod_{k|(j,k)\in\Psi} \sigma_{j|k}^2\right)^{0.5}} \sqrt{\frac{1}{B_j}} e^{-0.5\left(D_{js}-\frac{C_{js}^2}{4B_j}\right)}\right)^{-n}$$

$$P(\sigma_{jk}|.) \propto f(\Delta_{jk}) \times \left(\frac{1}{|S|}\sum_{s\in S} \frac{1}{\left(\prod_{k|(j,l)\in\Psi} \sigma_{j|l}^2\right)^{0.5}} \sqrt{\frac{1}{B_j}} e^{-0.5\left(D_{js}-\frac{C_{js}^2}{4B_j}\right)}\right)^{-n}$$

where

$$\begin{pmatrix} A_{jj} & A_{jk} \\ A_{jk} & A_{kk} \end{pmatrix} = \begin{pmatrix} \sigma_j^2 & \sigma_{jk} \\ \sigma_{jk} & \sigma_j^2 \end{pmatrix}^{-1}$$

Sample σ_j^2 and $\sigma_{j,k}^2$

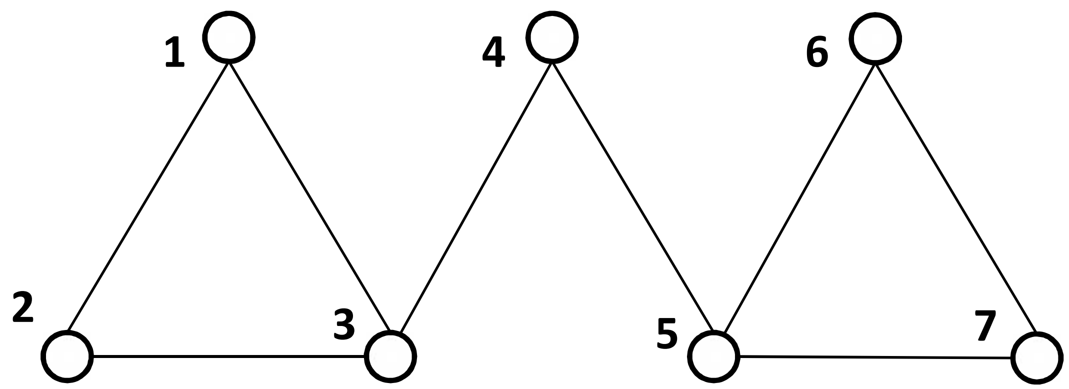


Figure. The correlation graph.

1. Find the maximum independent set (MIS).

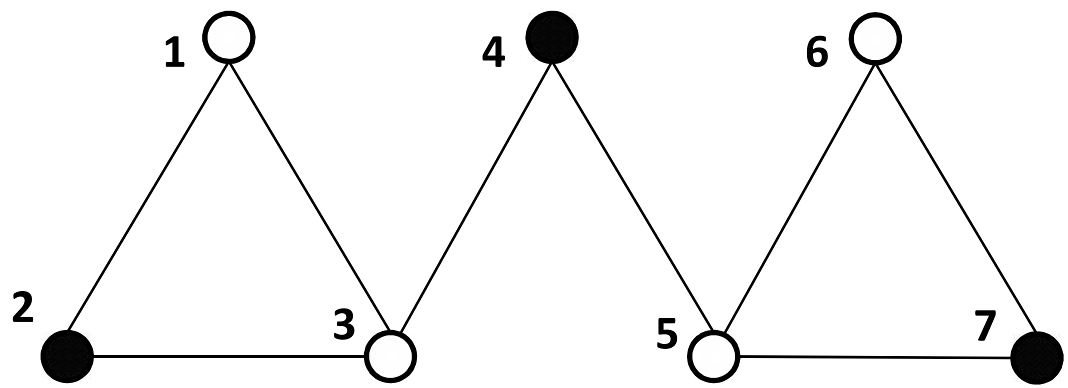


Figure. The correlation graph.

2. Sample σ_j^2 and $\sigma_{j,k}^2$ for selected vertices.

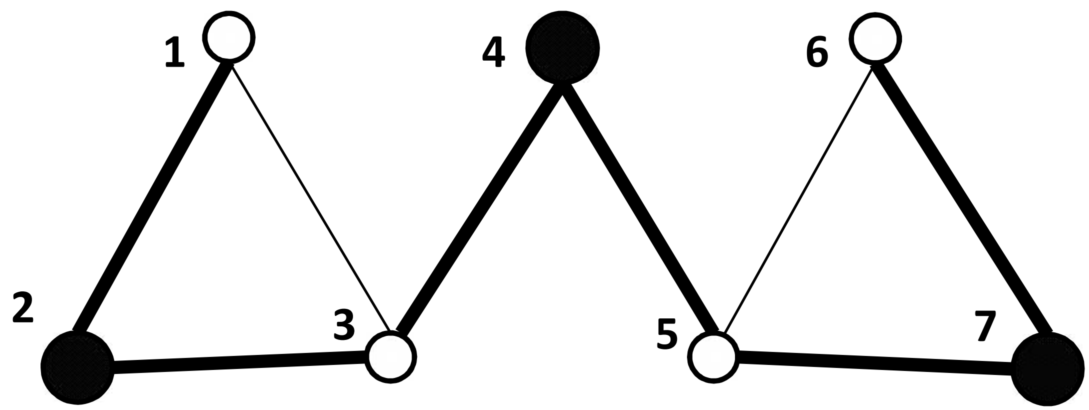


Figure. The correlation graph.

3. Find the MIS in the remaining graph.

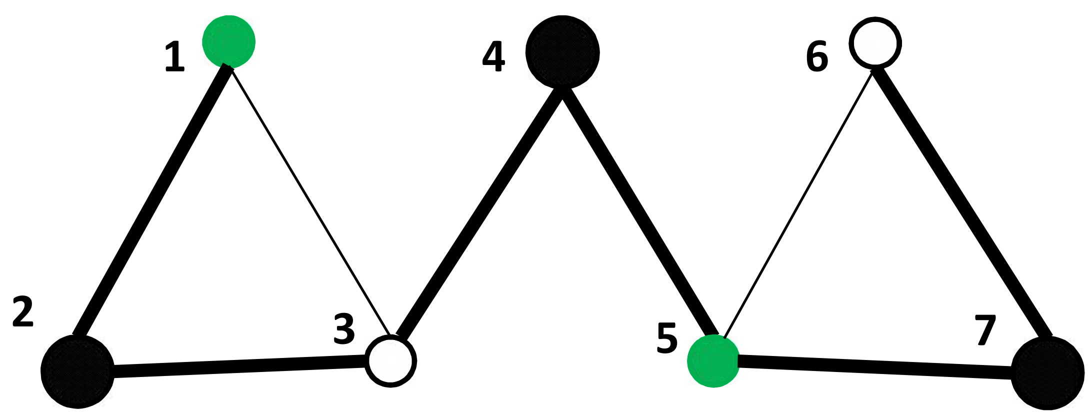


Figure. The correlation graph.

4. Sample σ_j^2 and $\sigma_{j,k}^2$ for selected vertices.

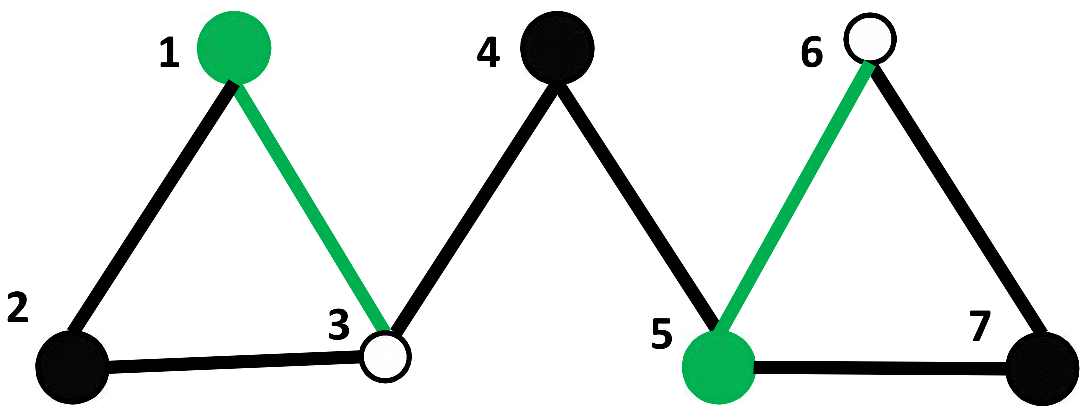


Figure. The correlation graph.

5. Sample σ_j^2 and $\sigma_{j,k}^2$ for remaining vertices.

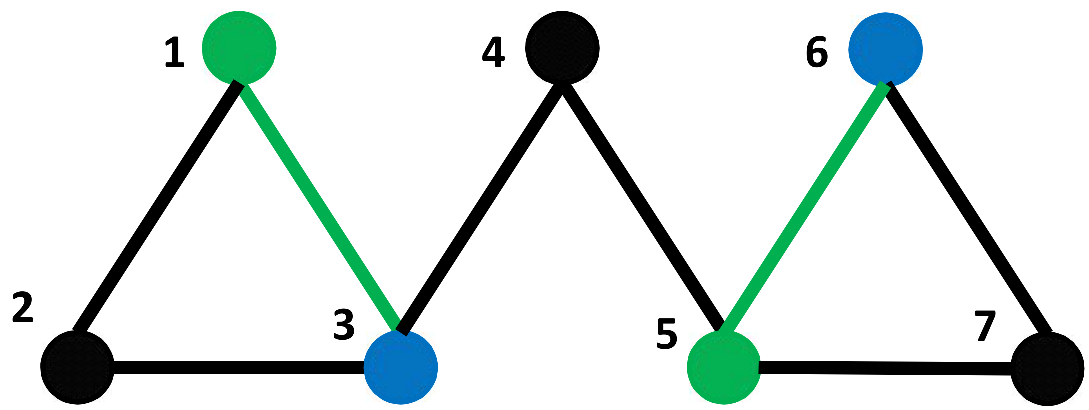


Figure. The correlation graph.

Algorithm for MRF

1. Collect *MRF aggregates* $\{\sum_i x_{ij}\}, \{\sum_i x_{ij}^2\}$ and $\{\sum_i x_{ij} x_{ik}\}_{(j,k)\in\Psi}$.
2. *Sample* $\{\mu_j\}$.
3. Find the *maximum independent set* of nodes in the MRF graph.
4. Sample the $\{\sigma_j^2\}$ and $\{\sigma_{jk}\}$ for nodes in the independent set.
5. Go to step 3.

Conclusion

- Parallel ML
  - Partition the problem into parallel jobs.
  - Figure out jobs to parallelize, not all the tasks are needed to be parallel.
  - Accelerate the convergence.