Peer-to-Peer Distributed Data Mining for Multi-Agent Applications

Hillol Kargupta
University of Maryland, Baltimore County and AGNIK
www.cs.umbc.edu/~hillol

Multi-Agent Systems (MAS)

- **Distributed collaborative problem solving**
- **Examples:**
  - Distributed control
  - Electronic commerce
  - Supply-chain management
  - Information gathering
  - Data mining

Roadmap

- Introduction
- Data Mining for Peer-to-Peer Networks
- Local Algorithms
  - Exact Local Algorithms
  - Approximate Local Algorithms
- Privacy Issues
- Resources

MAS for Data Mining Applications

- MAS + Existing Centralized Data Mining Algorithms
- Problems: Does not match with MAS architecture and philosophy
- Alternate Possibility: MAS + Inherently distributed algorithms for data mining

Data Mining and Distributed Data Mining

- Data Mining: Scalable analysis of data by paying careful attention to the resources:
  - computing,
  - communication,
  - storage, and
  - human-computer interaction.
- Distributed data mining (DDM): Mining data using distributed resources.

What is a Peer-to-peer (P2P) Network?

- Relies primarily on the computing resources of the participants in the network rather than a relatively low number of servers.
- P2P networks are typically used for connecting nodes via largely ad hoc connections.
- No central administrator/coordinator
- Peers simultaneously function as both "clients" and "servers"
- Privacy is an important issue in most P2P applications

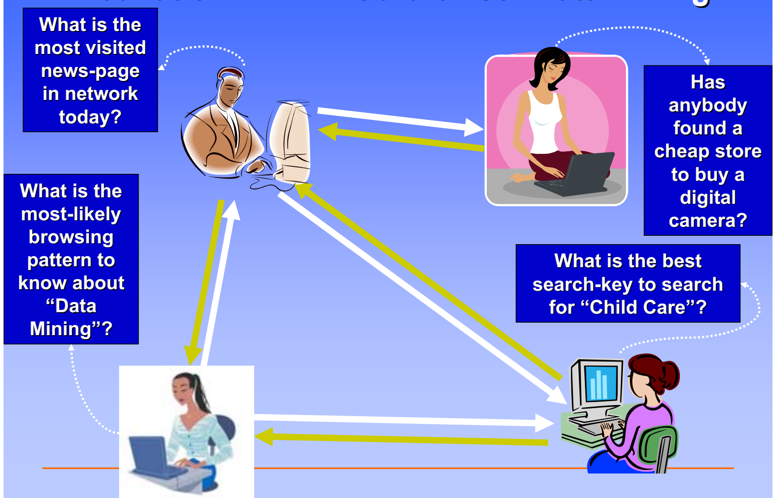
Data Mining for Distributed and Ubiquitous Environments: Applications

- **Mining Large Databases from distributed sites**
  - Grid data mining in Earth Science, Astronomy, Counter-terrorism, Bioinformatics
- **Monitoring Multiple time critical data streams**
  - Monitoring vehicle data streams in real-time
  - Monitoring physiological data streams
- **Analyzing data in Lightweight Sensor Networks and Mobile devices**
  - Limited network bandwidth
  - Limited power supply
- **Preserving privacy**
  - Security/Safety related applications
- **Peer-to-peer data mining**
  - Large decentralized asynchronous environments

Where do we find P2P Networks?

- Applications:
  - **File-sharing networks: KaZAa, Napster, Gnutella**
  - **P2P network storage, web caching,**
  - **P2P bio-informatics,**
  - **P2P astronomy,**
  - **P2P Information retrieval**
- P2P Sensor Networks?
- P2P Mobile Ad-hoc NETwork (MANET)?
- Next Generation:
  - **P2P Search Engines, Social Networking, Digital libraries, P2P "YouTube"?**

Motivation : P2P Web-browser Data Mining



P2P NASA Astronomy Data Mining

- Virtual Observatories
  - Client-server architecture
  - Consider Sloan Digital Sky Survey:
    - 2M hits per month
    - traffic is doubling every 15 months
  - Need better scalability
- MyDB: Download and locally manage your data
- Network of such databases
- Searching, clustering, and outlier detection in P2P virtual observatory data network.
- NASA AIST Project at UMBC

Useful Browser Data

1. Web-browser history
2. Browser cache
3. Click-stream data stored at browser (browsing pattern)
4. Search queries typed in the search engine
5. User profile
6. Bookmarks

Analyzing & Monitoring Data in P2P Environments

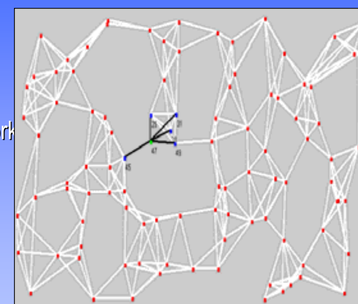
- Very large decentralized networks
- Dynamic topology
- Heterogeneous computing, communication, and storage resources
- Privacy is an important issue in most P2P applications

Problems of Existing DDM Algorithms

- Synchronous
- Require some central control
- Maintain information about the entire network by communicating with every node

Locality Sensitive Distributed Algorithms

- Global algorithms: Know everything about the entire network
  - Every node needs to maintain information about the entire network
  - Maintaining this information is resource intensive for large networks
- Local algorithms: Communicate only with the local neighborhood.



- Does locality imply efficiency?

Related Work on Peer-to-Peer Data Mining

- Distributed Majority Voting Algorithm (Gifford, 1979; Thomas, 1979; Wolff, Schuster, 2003)
- P2P Association Rule Learning (Wolff, Schuster, 2003)
- Gossiping (Kempe, Dobra, Gehrke 2003)
- Bandhopadhaya et al.'05
- P2P L2 Norm Monitoring (Wolff, Bhaduri, Kargupta, 2005)
- P2P Clustering (Dutta, Giannella, Kargupta, 2005)

Bounded Communication Local Algorithms

- Every node communicates with its local neighborhood
- In addition, the total amount of communication with its neighbors is also bounded

Some Useful Results

- Locally Checkable Labeling (LCL) problems---node labels can be checked by local computation
  - Undecidable: In general, whether a given LCL problem has a local algorithm
  - Decidable: Whether a given LCL has an algorithm that operates in a given time t
  - Randomization cannot make an LCL local
    - May help if we are willing to live with approximate solutions

Approaches

- Functions computation through decomposable representations
  - Exact decompositions
    - Deterministic techniques
  - Approximations
    - Randomized techniques
    - Sampling-based approximations
    - Variational approximations

Defining the Problem

- Let *G=(V, E)* be a graph
- Let Ω_k be the set of all neighboring nodes of the *k*-th node $v_k \in V$
- Need a decomposable representation where $f(V)$ can be computed from locally computed functions $\Phi_k(\Omega_k)$
- Example: $$f(V) = \sum_k w_k \Phi_k(\Omega_k)$$

Peer-to-Peer Majority Vote Computation

- Distributed Majority Voting Algorithm (Gifford, 1979; Thomas, 1979; Wolff, Schuster, 2003)
  - Node u send the following message to node v: ($count^{uv}$, sum^{uv})
  - $count^{uv}$: Number of bits the message reports
  - sum^{uv}: Number of those bits that are equal to 1.
  - For every neighbor v the node u records the last message it received from and sent to v.
  - S^u: The local bit
  - E^u: The set of edges colliding with u

Updating and Propagating Information

- Node u calculates the following:

$$\Delta^u = s^u + \sum_{(v,u)\in E^u} sum^{vu} - \lambda\left(c^u + \sum_{(v,u)\in E^u} count^{vu}\right)$$

$$\Delta^{uv} = sum^{uv} + sum^{vu} - \lambda\left(count^{uv} + count^{vu}\right)$$

- Update Δ^u when:
  - S^u changes, a message is received, E^u changes
- Update Δ^{uv} when:
  - A message is sent to or received from v

Continued

On changes in s^u, E^u or receiving a message:

For each $(v,u) \in E^u$

If $count^{uv} + count^{vu} = 0$ and $\Delta^u \geq 0$ or $count^{uv} + count^{vu} > 0$ and

either $\Delta^{uv} < 0$ and $\Delta^u > \Delta^{uv}$ or $\Delta^{uv} \geq 0$ and $\Delta^u < \Delta^{uv}$

Set $sum^{uv} = s^u + \sum_{(w,u)\neq(v,u)\in E^u} sum^{wu}$ and $count^{uv} = c^u + \sum_{(w,u)\neq(v,u)\in E^u} count^{wu}$

Send $\{sum^{uv}, count^{uv}\}$ over vu to v

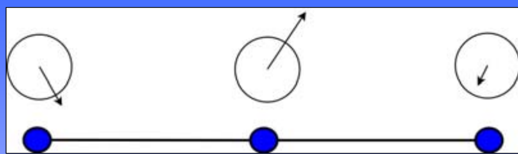
Continued

- Input the Edge set, local bit su and the majority ratio.
- At any given time the algorithm outputs 1 if $\Delta^u \geq 0$
- Each node performs the protocol independently.

Note

- As long as $\Delta^u \geq \Delta^{uv} \geq 0$ and $\Delta^v \geq \Delta^{vu} \geq 0$ there is no need for u and v to exchange data
- If $\Delta^{uv} > \Delta^u$ then v might mistakenly calculate $\Delta^v \geq 0$ then u needs to send v a message.
- Other similar conditions

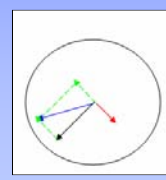
Local L2 Norm Monitoring Algorithm



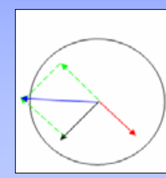
- Initial setup: each peer has
  - A data vector
  - Some global pattern vector
- Monitoring Problem:
  - is the L2 norm of the distance between the average data vector and the pattern vector greater than a given constant ε
- Applications:
  - Centroid monitoring
  - Eigenvector monitoring

Possibilities

1. All 3 vectors inside circle
2. All 3 vectors outside circle
3. Some are inside, some are outside

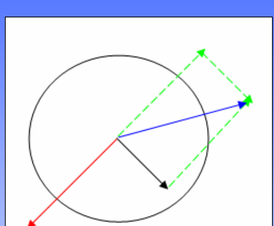


Case 1



Case 3

Local Vectors



- For peer P_i
  - Own estimate of global average (X)
  - Agreement with neighbor P_j (Y)
  - Withheld knowledge w.r.t neighbor P_j (Z=X-Y)

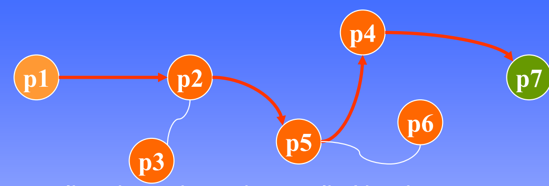
Theorem

- If for **every** peer and **each** of its neighbours **both** the agreement and the withheld knowledge are in a convex shape (here a circle) - **then so is the global average**
- **Wolff, Bhaduri, Kargupta, 2005**

Extension: Computing Cluster Centroid

- Beyond Monitoring
- Exact *local* algorithm not available
- How about Approximation?

Sampling in Distributed Environments



- i.i.d sampling through random walk: Needs more attention for data mining application
- Metropollis-Hasting algorithm for random walk with O(lg n) steps for selecting i.i.d. samples
- Issues:
  - Nodes may have different degrees
  - Nodes may contain different number of data tuples

Approximation

- Estimate $\Phi_k(\Omega_k)$
  - Cardinal sampling
  - Ordinal relaxation
    - Interested in constructing an ordering
    - Find the ones that rank high

Random Sampling

Protocol 5.2.1 Metropolis-Hastings Random Walk

FOR each node i, $1 \le i \le n$
  IF receives a query q
    Replies with d_i
  IF receives a random walk message
    IF TTL == 0
      Terminates the walk
    ELSE
      TTL = TTL - 1
      Sends out a query q to its neighbors $\Psi(i)$
      IF receives all the replies from its $\Psi(i)$
        Modifies transition probability p_{ij} as follows:

$$p_{ij} = \begin{cases} 1/\max(d_i, d_j) & \text{if } i \ne j \text{ and } j \in \Psi(i) \\ 1 - \sum_{k \in \Psi(i)} p_{ik} & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}$$

        Walk to next node with probability p_{ij}.

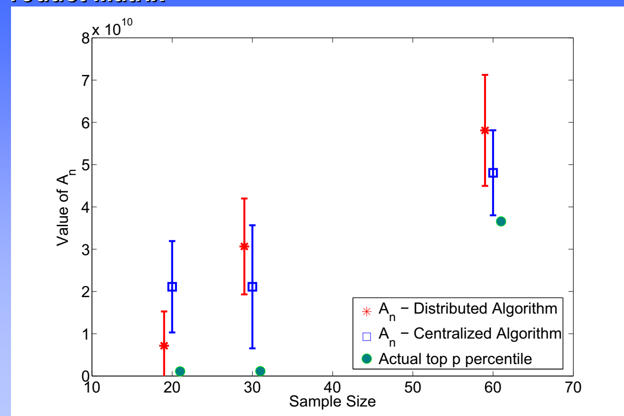
Ordinal Relaxation

- Let X be a continuous random variable
- Let ξ_p be the population percentile of order p, i.e. $\Pr\{x \le \xi_p\} = p$
- Let $x_1 < x_2 < \ldots < x_N$ be N independent samples from X
- We have

$$\Pr\{x_N > \xi_p\} > q \Rightarrow N \ge \left\lceil \frac{\log(1-q)}{\log p} \right\rceil$$

- Example:
  - q=95% and p =80% ➔ N=14
  - If we took 14 independent samples from any distribution, we can be 95% confident that 80% of the population would below x_{14}.

Ordinal Identification of Significant Entries from the Inner Product Matrix



Bhaduri, Das, Kargupta. (2006). An Ordinal Approach for Detecting Feature-Interaction in a Peer-to-Peer Network

Ordinal Inner Product Computation

- Each node has a vector X_i
- Compute the Inner Product Matrix
  - Every node needs X_i from every node.
- How about finding just the top-k entries of the inner product matrix?

Variational Approximation: Another Possibility

- Formulate as an optimization problem
- Introduce approximations
- Example: Finite Element Technique

Solve $-u''(x) = f(x), \quad x \in (a,b), u(a) = u(b) = 0$

Equivalent to minimizing $J(u) = \int_a^b (u^{*'}(x) - u'(x))^2 \, dx$

Continued

- Introduce approximation using locally decomposable representation
- Example: $u(x) = \sum_i \alpha_i \gamma(x)$
- Plug-in the approximation in the objective function

Continued

$$Z_c = \sum_{x_h} \exp(-\phi_c(x_h;\theta))$$

- Very expensive to compute

Inferencing in a P2P Network

- Compute $P(x_h \mid x_v;\theta) = \frac{P(x_h, x_v \mid \theta)}{P(x_v \mid \theta)}$
- Hidden variables x_h; observed variables x_v
- Boltzmann machine: $P(x_h \mid x_v;\theta) = \frac{1}{Z_c}\exp(-\phi_c(x_h;\theta))$
- Where $\phi_c(x_h;\theta) = -(\sum_{i<j} \theta_{i,j} x_i x_j + \sum_i \theta_{i0}^c x_i)$
- indices i, j vary over the hidden variables and

$$\theta_{i0}^c = \theta_{i0} + \sum_{x_j \in x_v} \theta_{ij} x_j$$

Variational Approximation

- Approximate $P(x_h \mid x_v;\theta)$ by a distribution $Q(x_h)$

$$Q(x_h) = \prod_i Q_i(x_i)$$

Optimization Formulation

- Objective function

$$J(Q) = -\sum Q(x_h)\ln \frac{Q(x_h)}{\exp(-\phi_c(x_h;\theta))}$$

- Q_i be a binary distribution $Q_i(x_i) = \mu_i^{x_i}(1-\mu_i)^{1-x_i}$
- Solution that maximize J(Q) is a set of mean field equations (one for each μ_i)

$$\mu_i = \sigma(\sum_{j>i}\theta_{ij}\mu_j + \theta_{i0}^c)$$

$$\sigma(z) = 1/(1+\exp(-z))$$

The Algorithm: VIDE

Output: The mean μ_i of the marginal distribution under the variational approximation, for every hidden variable X_i.

Let p be the number of nodes in the network.

for each hidden variable X_i **do**
  $\psi_i \leftarrow Distributed\_Average(\{\theta_{ij}x_j : x_j \in \mathbf{x}_v\})$
  $\theta_{i0}^c \leftarrow \theta_{i0} + p \cdot \psi_i$
  Also, initialize $\mu_i \leftarrow \frac{1}{2}$
end for
while μ has changed significantly in the last iteration **do**
  for each hidden variable X_i **do**
    $\mu_i \leftarrow \sigma(\sum_{j>i}\mu_j + \theta_{i0}^c)$
  end for
end while

Distributed Inferencing

- Heterogeneous Data
  - Each site has a subset of observed attributes
- Each node knows about the parameters of the distribution
- Compute $P(x_h \mid x_v;\theta)$ using the variational approach

Experimental Results

- Objectives:
- To measure the **accuracy** of the variational approximation as a function of the communication requirement.
- To measure the **scalability** of the variational approach: For a given accuracy level, how does communication requirement change as the number of sites increases?

How to Quantify the Accuracy

- The inference problem is reduced to the problem of computing the solution vector to a system of mean-field equations, in a decentralized manner.
- Each *j* site has its own estimate of the solution. Corresponding to this site, there is a relative error *err_j* with respect to the centralized variational approximation.
- Our error function takes the maximum of *err_j over all sites j.* We denote it by *max_err.*
- We measure the mean and variance of *max_err,* in our experiments.

Description of Data Sets

- Synthetic Data Sets (contd.):
- *Note: Since it will take exponential time to generate the test vectors according to Boltzmann Distribution, we choose the test vectors in a uniformly random way.*
- We choose *(H=10, V=50), (H=20, V=100) and (H=40, V=200) in our experiments.*

Description of Data Sets

- Synthetic Data Sets:
- Let *H = number of hidden variables, and V = number of visible variables.*
- Parameters of the Boltzmann Machine are selected in a uniformly random way from [0,1).
- Generate a set of *V-dimensional vectors of visible variables.* For each vector, compute the maximum relative error, as defined in the previous slide.

Description of Data Sets

- Real-life Data Sets:
- Parameters were chosen just as with synthetic data.
- For testing, we used time series data sets ecg200 , Adiac, and 50words
- UC Riverside Time Series Data Repository
- H and V are chosen as follows:
- Ecg200: H= 18 , V=50
- Adiac: H=36 , V=100
- 50words: H=72 , V=200

Accuracy – Uniform Newscast, Synthetic Data

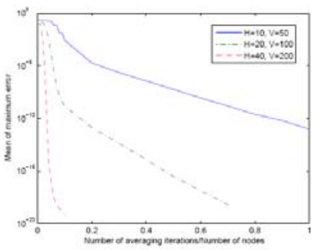


Figure 1. Mean of maximum relative error vs. communication factor β_N using Uniform Newscast and synthetic data. Recall that β_N = (Number of iterations in the Newscast averaging process)/(Number of nodes in the network).

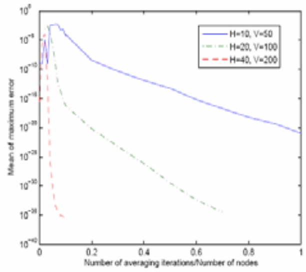


Figure 2. Variance of maximum relative error vs. communication factor β_N using Uniform Newscast and synthetic data.

Accuracy: Uniform Newscast, Real Data

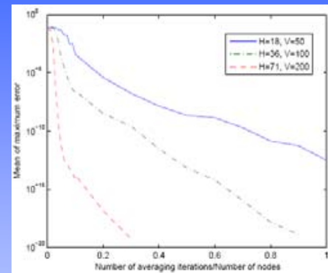


Figure 5. Mean of maximum relative error vs. communication factor β_N using Uniform Newscast and real-life data.

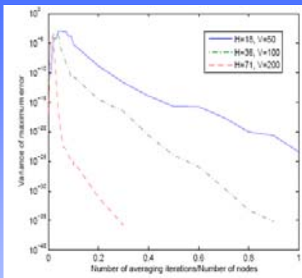


Figure 6. Variance of maximum relative error vs. communication factor β_N using Uniform Newscast and real-life data.

Accuracy: Metropolis Hastings Sampling, Synthetic Data

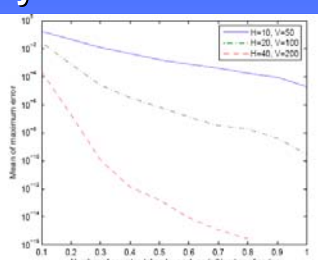


Figure 3. Mean of maximum relative error vs. communication factor β_{MH} using Metropolis-Hastings sampling and synthetic data. Recall that β_{MH} = (Number of samples taken by each node)/(Number of nodes).

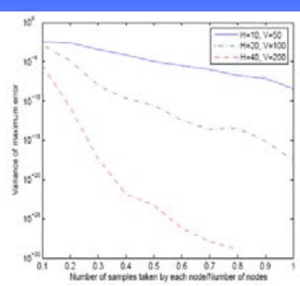


Figure 4. Variance of maximum relative error vs. communication factor β_{MH} using Metropolis-Hastings sampling and synthetic data.

Accuracy: Metropolis-Hastings Sampling, Real Data

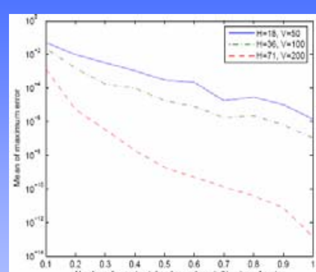


Figure 7. Mean of maximum relative error vs. communication factor β_{MH} using Metropolis-Hastings sampling and real-life data.

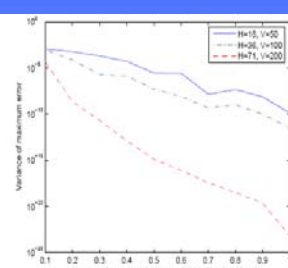


Figure 8. Variance of maximum relative error vs. communication factor β_{MH} using Metropolis-Hastings sampling and real-life data.

Scalability: Uniform Sampling

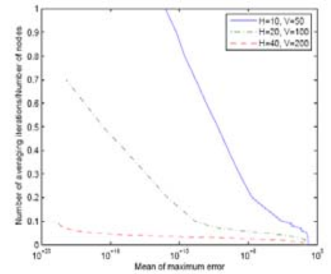


Figure 9. Dependence of the communication factor β_N on the allowed mean-value for maximum relative error, for Uniform Newscast with synthetic data.

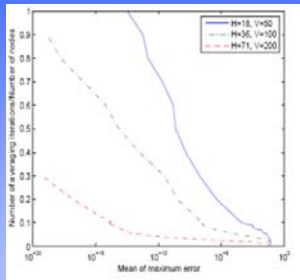
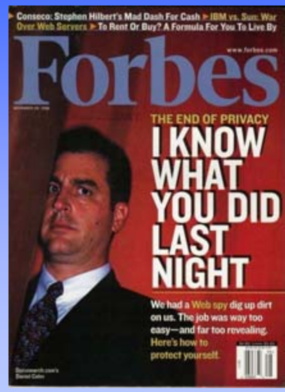


Figure 11. Dependence of the communication factor β_N on the allowed mean-value for maximum relative error, for Uniform Newscast with real-life data.

Privacy and Data Mining



- "the best (and perhaps only) way to overcome the 'limitations' of data mining techniques is to do more research in data mining, including areas like data security and privacy-preserving data mining, which are actually active and growing research areas."
  - SIGKDD Executive Committee, "'Data Mining' Is NOT Against Civil Liberties," 2003.
- Privacy-preserving data mining is "the study of how to produce valid mining models and patterns without disclosing private information."
  - F. Giannotti and F. Bonchi, "Privacy Preserving Data Mining," KDUbiq Summer School, 2006.

Scalability: Metropolis-Hastings Sampling

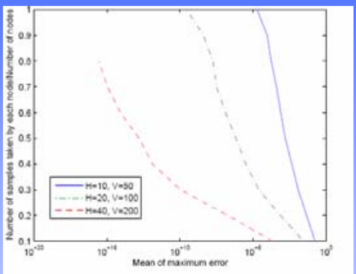


Figure 10. Dependence of the communication factor β_{MH} on the allowed mean-value for the maximum relative error, for Metropolis-Hastings sampling with synthetic data.

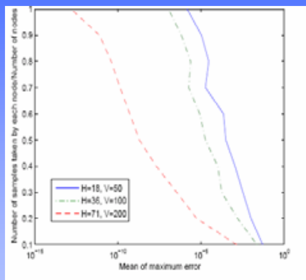
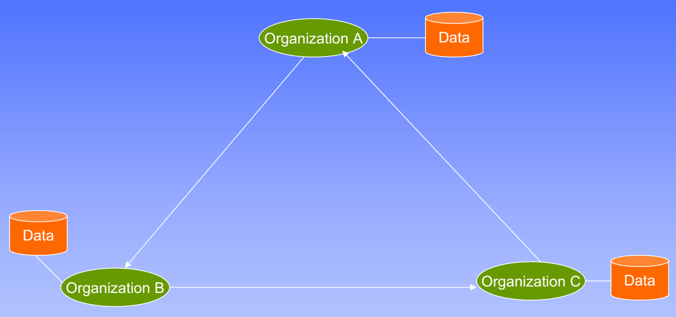


Figure 12. Dependence of the communication factor β_{MH} on the allowed mean-value for the maximum relative error, for Metropolis-Hastings sampling with real-life data.

Privacy-Preserving Data Mining (PPDM)



Compare, match, and analyze data from different organizations without disclosing the private data to any other party

Blending Privacy-Preserving Techniques

- Some of the Existing Frameworks:
  - Data sanitization
  - Random additive perturbation (Agrawal and Srikant, 2001)
  - Random multiplicative noise (Liu, Kargupta, 2004)
  - Secured Multi-Party Computation (Goldreich, 1998, Cliffton et al., 2003)
  - K-Anonymity (Sweeney, 2002)
- Problems: Makes many assumptions about the user behavior.
  - Performs computations and communications as expected
  - Semi-honest

Resources

- DDMWiki
- http://www.umbc.edu/ddm/wiki/
- Visit and list your papers, projects, books,…..

Conclusions

- P2P Data Mining: An emerging area of distributed data mining with many potential applications
- Communication-bounded local algorithms
- Exact local algorithms may not be able to solve all interesting data mining problems
- Approximate local algorithms:
  - Probabilistic approximation
  - Deterministic approximation

References: Local Computation

- Distributed Computing: A Locality-Sensitive Approach, David Peleg, SIAM, 2000.
- M. Naor and L. Stockmeyer. (1995). What can be computed locally? SIAM Journal on Computing, Volume 24 , Issue 6, Pages: 1259 - 1277

References: P2P and Distributed Data Mining

- S. Datta, K. Bhaduri, C. Giannella, R. Wolff, H. Kargupta. (2006). Distributed Data Mining in Peer-to-Peer Networks. IEEE Internet Computing special issue on Distributed Data Mining. Volume 10, Number 4, page 18-26.
- R. Wolff, K. Bhaduri, H. Kargupta. (2006). Local L2 Thresholding Based Data Mining in Peer-to-Peer Systems. Proceedings of the 2006 SIAM International Data Mining Conference. http://www.cs.umbc.edu/~hillol/Kargupta/pubs.html
- M. Mehyar, D. Spanos, J. Pongsajapan, S. Low, R. Murray. (2005). Distributed Averaging on Peer-to-Peer Networks. http://www.cds.caltech.edu/~demetri/docs/IPSN05_MSPLM.pdf
- S. Bandyopadhyay, C. Gianella, U. Maulik, H. Kargupta, K. Liu, and S. Datta. (2005). Clustering Distributed Data Streams in Peer-to-Peer Environments.

References: P2P and Distributed Data Mining

- K. Das, K. Bhaduri, K. Liu, H. Kargupta. (2006). Identifying Significant Inner Product Elements in a Peer-to-Peer Network. *IEEE Transactions on Knowledge and Data Engineering*. (Accepted, in press)
- K. Liu, K Bhaduri, K. Das, P. Nguyen, H. Kargupta (2006). Client-side Web Mining for Community Formation in Peer-to-Peer Environments. *ACM SIGKDD Explorations*. Volume 8, Issue 2, Pages 11 - 20.
- H. Kargupta and K. Sivakumar, (2004) Existential Pleasures of Distributed Data Mining. Data Mining: Next Generation Challenges and Future Directions. Editors: H. Kargupta, A. Joshi, K. Sivakumar, and Y. Yesha. AAAI/MIT Press.