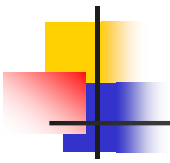
Distributed Data Mining for Sensor Networks

Hillol Kargupta

Department of Computer Science and Electrical Engineering
University of Maryland Baltimore County
Baltimore, MD 21250, USA
http://www.cs.umbc.edu/~hillol
hillol@cs.umbc.edu
&
AGNIK, LLC
Columbia, MD 21045
http://www.agnik.com
hillol@agnik.com

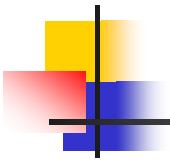
Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources



Data Mining

- Scalable analysis of data by paying careful attention to issues in
  - computing,
  - communication,
  - storage, and
  - human-computer interaction.

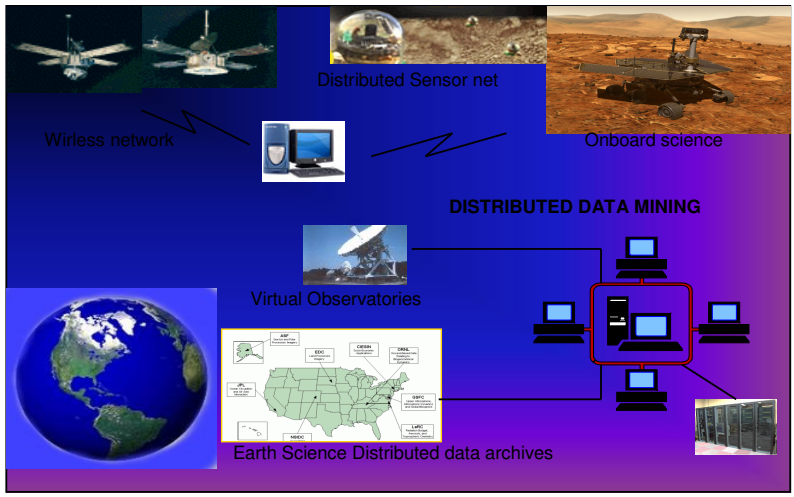


Distributed Data Mining

- Mining data using distributed resources.
- Pays careful attention to the distributed resources of data, computing, communication, and human factors in order to use them in a near optimal fashion.
- Typical application environments:
  - Distributed computing nodes
  - Single or multiple sources of data, possibly privacy-sensitive
  - Distributed/mobile users
  - Wired or wireless networks

Data Mining in Distributed and Mobile Environments

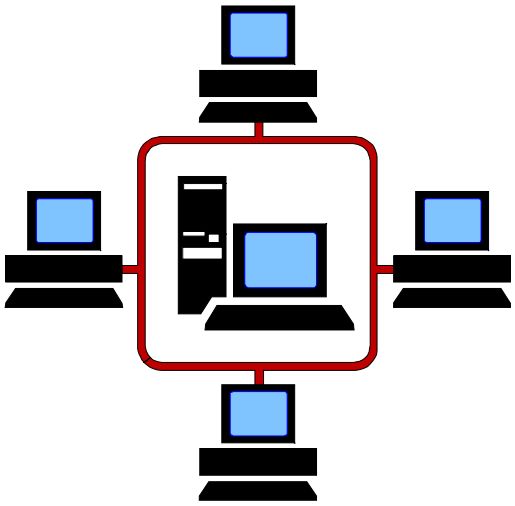
- **Mining Databases from distributed sites**
  - Earth Science, Astronomy, Counter-terrorism, Bioinformatics
- **Monitoring Multiple time-critical data streams**
  - Monitoring vehicle data streams in real-time
  - Onboard science
- **Mining sensor networks**
  - Limited bandwidth
  - Limited power supply
- **Preserving privacy**
  - Security/Safety related applications



DDM Applications: Typical Characteristics

- Distributed computing environment
- Heterogeneous communication links with bandwidth constraints
- Wireless networks
- Distributed data
- Continuous data streams
- Multi-party data, sometimes privacy sensitive (difficult to centralize)

Network of Desktop Computers



Network of desktop workstations

Ubiquitous Computing Devices

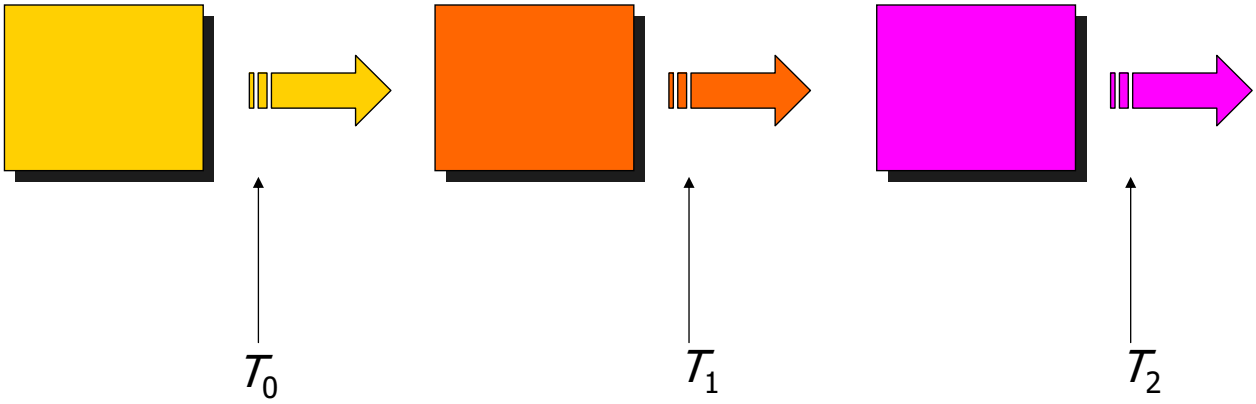
Laptops, palmtops, PDAs, cell phones, smart cards, and wearable computers

Courtsey: http://www.handspring.com and http://www.casio.com, http://www.scmegastore.com/ http://www.wearcam.org/computing.html/, http://java.sun.com/products/javacard/,

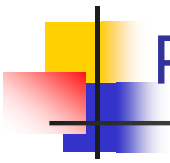
Data Communication

- Case I: Participating nodes are connected by high speed networks and efficient redistribution of data is possible.
- Case II: Nodes are connected by low speed networks and data redistribution is difficult to support.

Stream Data Sources

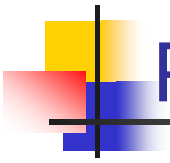


- Continuous stream of incoming data
- Storing all the historical data is difficult
- Each record is examined a very small number of times, often just once
- Limited memory for storing summary/pattern from the data
- Data observation in burst
- Real-time performance



Privacy Sensitive Multi-Party Data

- Privacy-sensitive data
- Multi-party owned data stored at different locations
- So collecting even part of the raw data sets from different locations is not acceptable unless privacy is protected



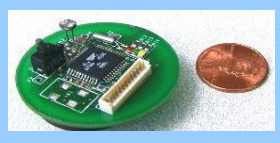
Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources

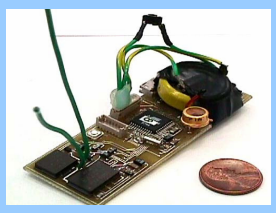
Sensor Networks

- Network of sensor and other supporting nodes (Akyildiz et al., 2002)
- Nodes may carry out local computations and transmit partially processed or raw data
- Nodes can also receive information from other nodes and combine that with its local information

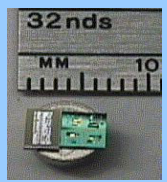
Sensor Nodes: Some Examples



UC Berkeley: COTS Dust



UC Berkeley: COTS Dust



UC Berkeley: Smart Dust

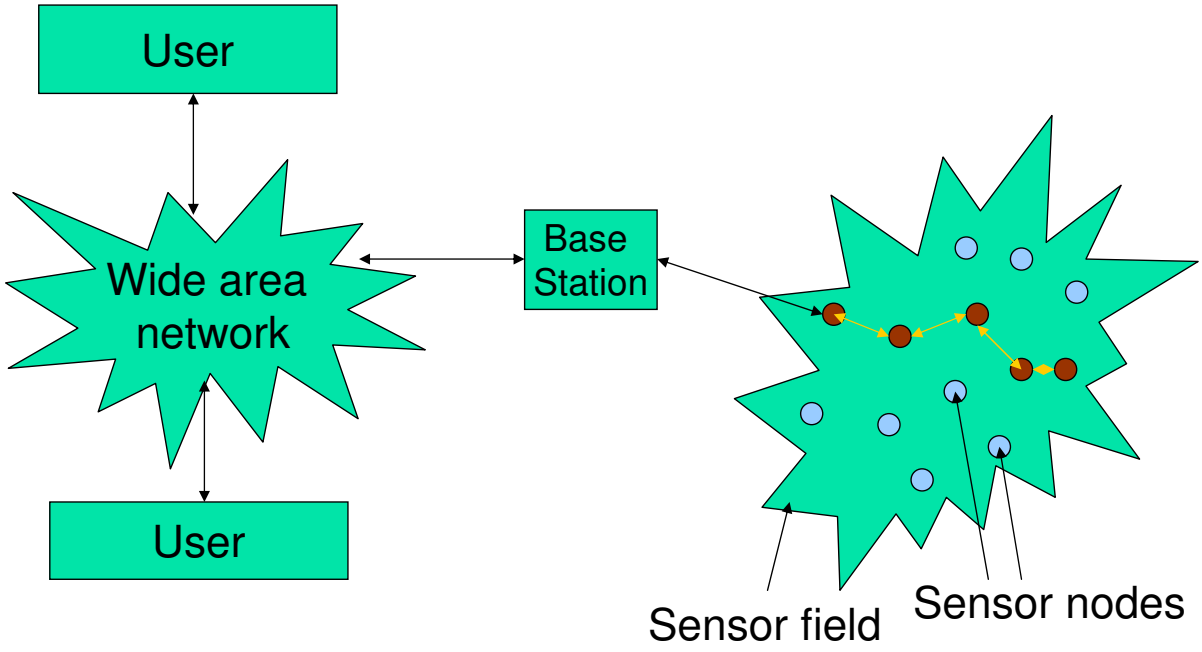
UCLA: WINS

Rockwell: WINS

JPL: Sensor Webs

Courtesy of *A Programmable Routing Framework for Autonomic Sensor Networks* by Yu He, Cauligi S. Raghavendra, Steven Berson, Robert Braden
Information Sciences Institute, University of Southern California.

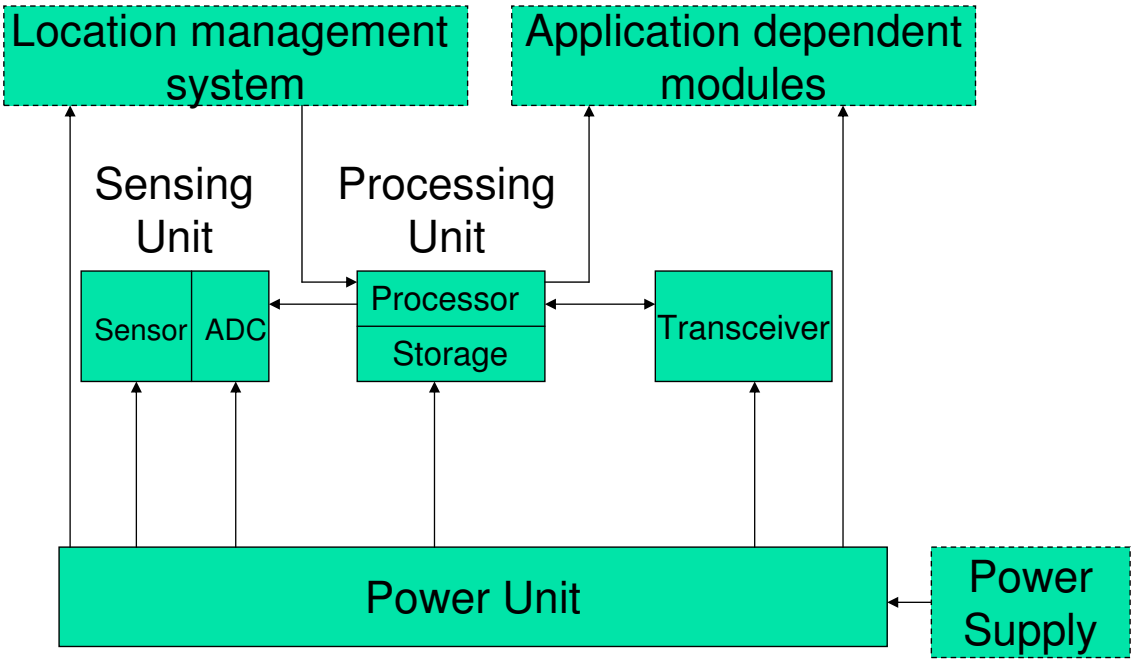
Communication Architecture



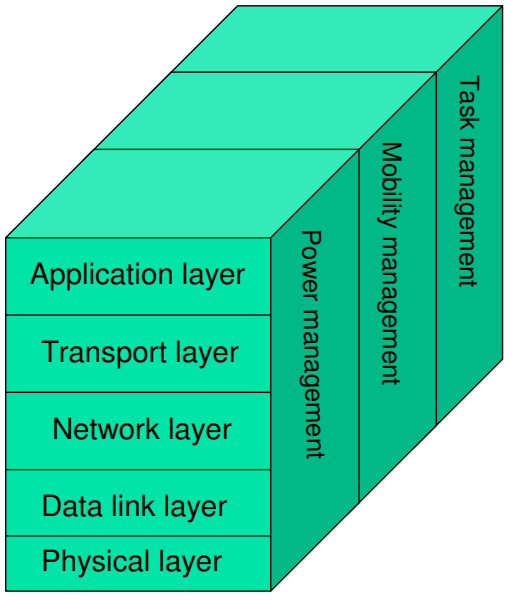
Some Design Issues

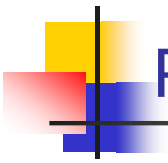
- Fault Tolerance: Sensor nodes are prone to failures because of no battery power, physical damage, or other interferences
- Scalability: Most envisioned applications of sensor networks are likely to use large number of nodes
- Production Costs: Deployment of large number of sensor nodes demand low cost
- Operating Environment: Remote unattended deployment
- Network Topology: How the nodes are deployed
- Node Identification: Lack of global IP for the nodes
- Hardware Constraints: Limited computing capabilities
- Transmission Media: Radio connection
- Power Consumption: Limited battery life

Typical Components of a Sensor Node



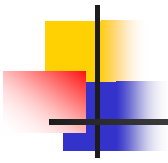
Sensor Networks Protocol Stack





Routing techniques

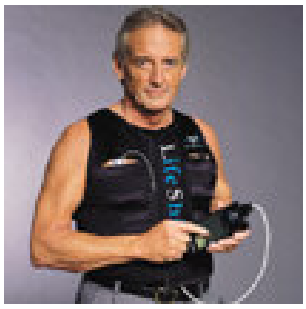
- Flooding: Each node receiving a data packet repeats it by broadcasting.
- Gossiping: Send the incoming packets to a randomly selected neighbor.



Sensor Networks: Applications

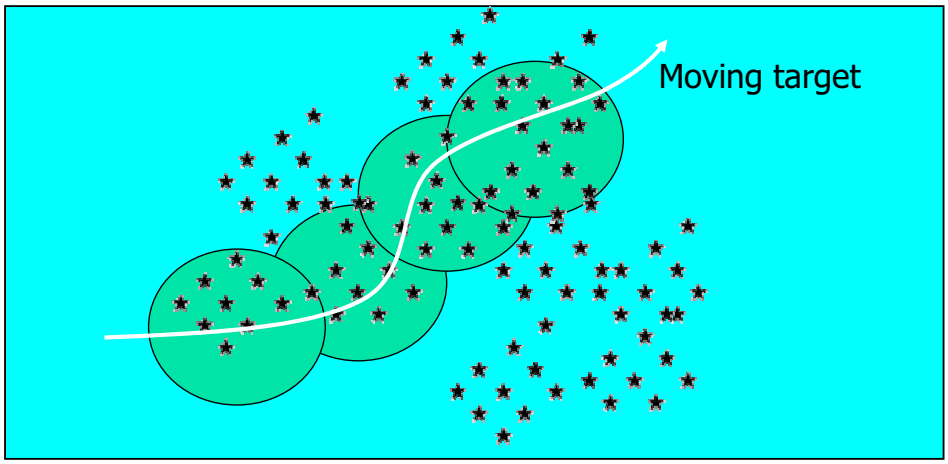
- Surveillance for defense and security applications
- Habitat Monitoring
- Chemical waste monitoring
- Soil condition monitoring and irrigation
- Pollution monitoring
- Disaster management: Floods, forest fires, earthquakes, tornadoes

Data Analysis Applications: Real-time Health Monitoring



- Smart shirts that collect many attributes in real-time
- Health Monitoring for fire fighters for safety evaluation

Data Analysis Applications: Target Tracking



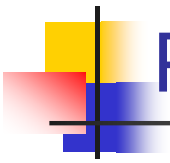
Identifying and tracking an object

Sensor Networks: Resource Constraints

- Limited bandwidth connections.
- Limited computing power.
- Limited battery power.
- Limited storage.
- Security and power management.
- Privacy issues in some applications

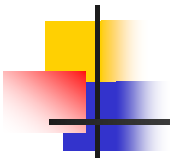
Need for In-Network Processing of Data

- Central collection and processing of data may not be scalable:
  - Data transmission consumes lot of power
  - Bandwidth is limited
  - Monitoring applications often require fast response time
  - Data may be privacy-sensitive



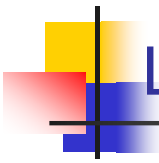
Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources



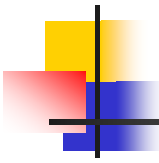
Data Processing in a Sensor Network

- Local computing at a given node:
  - Resource constrained data stream mining and monitoring
- Distributed collaborative sharing and processing of information:
  - Resource constrained distributed data mining



Local Data Stream Mining Algorithms

- Query processing
- Computing statistical aggregates
- Representation construction techniques
  - Principal component analysis
  - Randomized projection techniques
- Mining algorithms
  - Regression
  - Decision trees
  - Bayesian algorithms
  - Support vector machine
  - Clustering
  - Ensemble-based approaches
    - Bagging, Arching, Boosting
    - Advanced meta-analysis: Decision tree aggregation using Fourier analysis

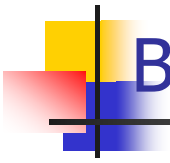


Why Stream Mining?

- Fixed distribution
  - Take a decent size sample, analyze the data, and that's it.
- Most stream applications for sensor networks are not like that
  - Distribution changes.
  - Detecting changes, modifying/replacing models, identifying outliers are important.

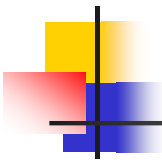
Most Stream Mining Algorithms: The Philosophy

- Incrementally update representations if needed.
- Incrementally update the models and summary statistics.
- Look for outliers, if interested.



Basic Stream Computing Primitives

- Query processing primitives
  - Gibbons, 2001
  - Alon, Gibbons, Matias, Szegedy, 1999
- Quantile computation
  - Manku, Rajagopalan, Lindsay, 1998
- Sketch-based computation
  - Alon, Gibbons, Matias, Szegedy, 1999.
- Counting problem
  - Datar, Gionis, Indyk, Motwani, 2001

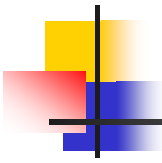


An Example: Counting Problem

- Problem: Count the number of 1-s from a moving window in a binary stream.

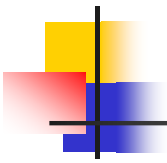
. . . . .1 0 0 0 1 1 1 0 1 0 1 0 0 0 1

- Need to account for the expiring bits. Naïve solution takes o(n) space.
- Can we come up with an approximate solution in o(log n) space?



An Approximate Solution

- Store the counting information among a set of buckets of known counts.
- Time-stamp of a bucket = time stamp of the most recent entry in the bucket.
- Track the buckets.
- When the time-stamp of a bucket expires, through away the bucket.
- Error in oldest bucket only.



Continued

- Exponential histograms: Buckets of exponentially increasing size.
- Bucket sizes: 1, 2, 2^2, 2^3, ... 2^h .
- Need only O(log N) buckets.
- A bucket size can take at most log N bits.

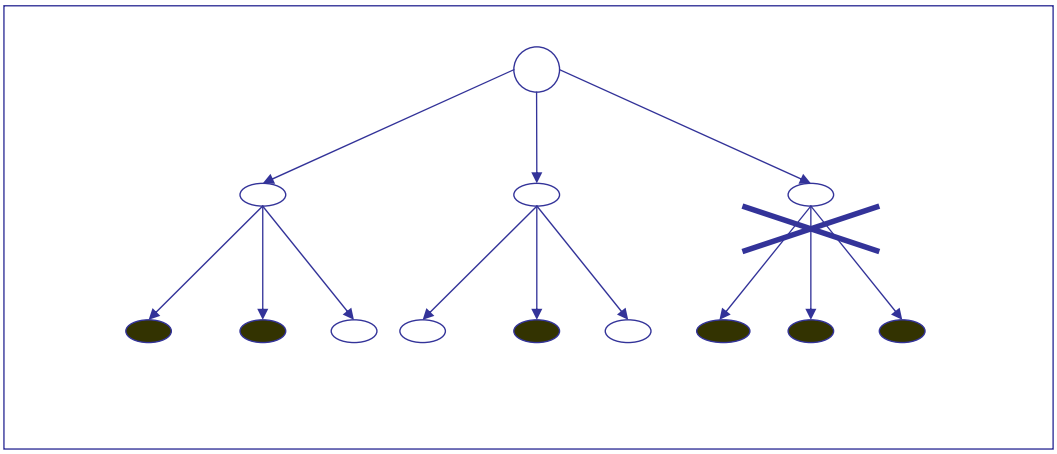
Correlation Matrix Computation

- Given data matrix X
- Naïve computation: Compute X^TX
- Compute in the frequency domain (take Fourier transformation)
- StatStream (Zhu and Shasha, 2002)

Resource Constrained Change Detection in the Correlation Matrix

- Kargupta, Puttagunta, Klein, 2004
- Data stream
- Efficiently detect changes in the correlation matrix
- Identify the region of the matrix that contain significantly changed coefficients

Divide-and-Conquer Search for Significant Correlation Coefficients



- Impose a tree-structure:
  - Leaf node: a unique correlation coefficient
  - Root of a sub-tree: set of all coefficients corresponding to the leaves in that sub-tree

Does a Sub-tree Contain Any Significant Coefficient?

Given a subset of attributes: $\{i_1, i_2, ...i_k\}$;

Is there any significantly correlated pair of attributes?

The j - th row of the data matrix X : $x_j = [x_{j,1} x_{j,2} .....x_{j,n}]$

Entries from the j - th row x_j corresponding to attributes in G

$[x_{j,i_1} x_{j,i_2} .....x_{j,i_k}]$

Continued

Consider a random vector $\sigma_p = [\sigma_{i_1,p}, \sigma_{i_2,p}, ...\sigma_{i_k,p}]$

$\sigma_{j,p} \in \{-1,1\}$ with uniform probability

$$s_{j,p} = \sum_{l=i_1,i_2,.... i_k} x_{j,l} \sigma_{l,p}$$

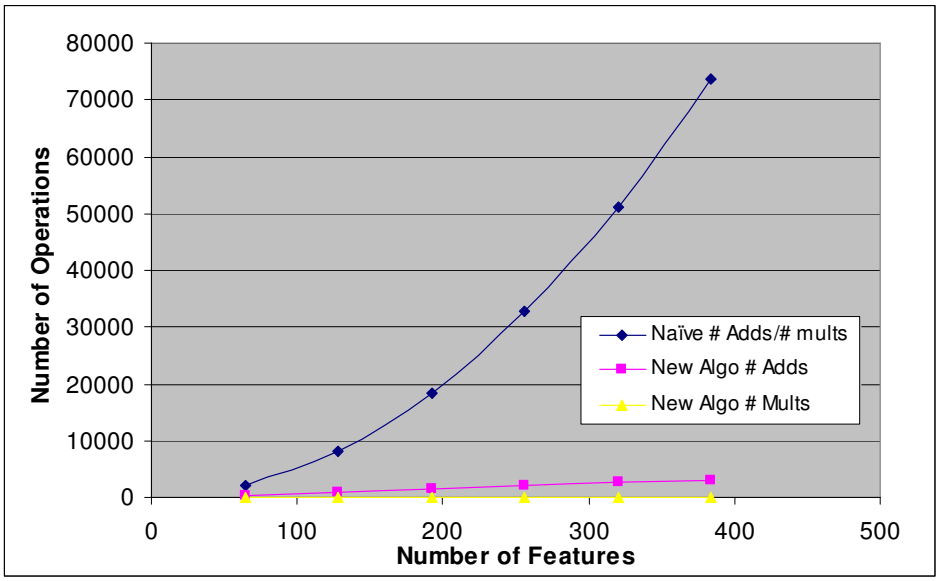
$$S_{\{i_1,i_2,....i_k\},p} = [s_{1,p} s_{2,p} \ldots s_{m,p}]^T$$

The Test at Every Node

$$\frac{1}{r}\sum_{p=1}^{r} Var(S_{\{i_1,i_2,\ldots i_k\},p})^2 \approx \sum_{l_1,q_1} Corr(x_{l1}, x_{q_1})^2$$

- Compute the left hand side at every node and proceed only if it is greater than a threshold.

Detecting No Changes



- **Number of multiplications and additions performed by the naïve and the tree-based algorithms for correctly detecting no significant changes in the correlation matrix.**

Detecting No Significant Changes

r	Number of Additions		Number of Multiplications		Number of Nodes	
	μ	σ	μ	σ	μ	σ
2	875.6	564.7	10.4	7.7	5.2	3.8
4	722.4	468.2	3.4	2.6	1.8	1.3
6	1692.6	1376.9	6	6.2	3	3.1
8	1025	0	2	0	1	0
10	1281	0	2	0	1	0

- Number of multiplications and additions (with a scaling factor of number-of-data-rows) performed by the tree-based algorithm. Threshold value is 0.6. Should be scaled by a factor equal to the number of data rows used for computing the correlation matrix
- Naïve approach requires 2016 x number-of-data-rows additions and multiplications

Incremental Representation Construction Techniques

- Incremental PCA
- Incremental randomized projection

Incremental PCA for Signal Processing Applications

- PCA is often needed for signal processing
- Example: Filtering, direction of audio signal arrival
- Covariance matrix is additive.
- Incrementally update the covariance matrix and perform eigen-analysis.

Covariance Computation from a Data Stream

- Data stream environment where data blocks $X_1, X_2, \ldots X_t$
- X_t is an $m_t \times n$ data matrix observed at time 't'
- Covariance is additive.

$$Cov_t = \frac{\sum_{j=1}^{t-1} m_j}{\sum_{j=1}^{t} m_j} \left[Cov_{t-1} + \frac{m_t}{\sum_{j=1}^{t-1} m_j} \hat{C}_t \right]$$

$\hat{C}_t$ is the covariance matrix constructed from the data set received at time t

Useful Bounds from Matrix Perturbation Theory

- (λ_1, v_1) (λ_2, v_2): Most significant eigenvalue and the corresponding eigenvector of Cov_t and Cov_{t-1} .

$$\Delta = Cov_t - Cov_{t-1}$$

$$\| v_1 - v_2 \|_2 \leq \frac{4 \| \Delta \|_F}{\delta - \sqrt{2} \| \Delta \|_F}$$

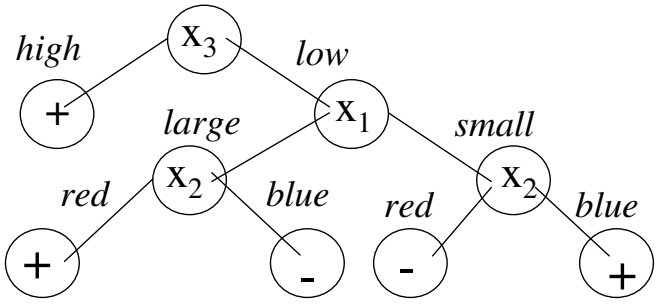
$$| \lambda_1 - \lambda_2 | \leq \sqrt{2} \| \Delta \|_F$$

$$\text{Frobenius norm} \| \Delta \|_F = \left(\sum_i \sum_j \Delta_{ij}^2 \right)^{1/2}$$

A Sampler of Other Stream Mining Algorithms

- Some more techniques
  - Decision trees from data streams
  - SVM-based classifier learning from streams
  - Bayesian networks from streams
  - Clustering data streams
  - Ensembles
- Originally proposed for high-end applications not resource constrained sensor nodes

Decision Tree from Streams



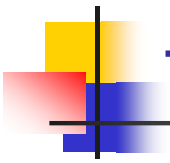
- A decision tree builds a classification tree from a labeled data-set.
- Nodes correspond to features and links correspond to feature values.
- Leaf nodes correspond to class labels

Hoeffding Tree

- (Domingos and Hulten, 2000)
- Based on Hoeffding or additive Chernoff bound
- With probability 1-δ the true mean of random variable x is at least E[x]-ε where E[x] is the mean of n samples and

$$\varepsilon = \sqrt{\frac{\beta^2 \ln(1/\delta)}{2n}}$$

- β is the range of x

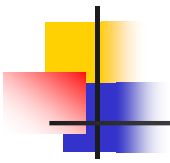


Tree Construction from Stream

- Initialize the tree with the root node.
- For each record in the stream:
  - Traverse the tree and find the appropriate leaf node, L.
  - Compute the information gain, $IG(X_i)$, for every attribute X_i , based on the subset of the data that includes the observed new record.
  - If $IG(X_i) - IG(X_j) > \varepsilon$ then split L using attribute X_i.

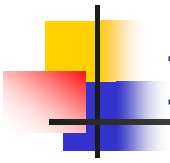
On-line Support Vector Machine (SVM) Learning

- Syed, Liu and Sung, 1999
  - Learn support vectors from one data block at a time
  - Add them to the new data set and learn support vectors from that
  - Approximate approach
- Cauwenberghs and Poggio, 2001
  - An ``exact on-line'' method to construct SVM considering one point at a time
  - Retain Kuhn-Tucker conditions on all previously seen data and ``adiabatically'' add new data points to the solution



Bayesian Networks

- (Kinouchi and Caticha, 1992; Winther and Solla, 1998; Andrieu, Freitas and Doucet, 2000)
- On-line Bayesian learning
- Updates posterior probability as new observations come in

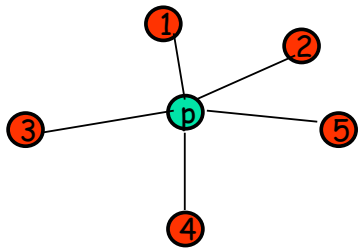


Incremental Clustering

- Many works on incremental clustering
- CF-tree-based approach BIRCH (Zhang,Ramakrishnan, Livny, 1997)
- Fuzzy incremental clustering (Joshi, and Krishnapuram, 1998)
- DSCAN-based approach for a metric space (Ester, 1998)
- Stream clustering algorithms: A Critical Perspective (Keogh, Lim, 2003)

K-Medians from Streams

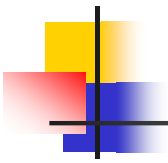
- Find k centers in a stream S so as to minimize the sum of distances from data points in S to their closest cluster centers.
- Example (k=1)



cost = $\sum_j d(p, j)$ is minimum

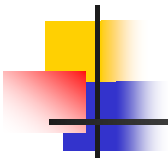
Continued

- For each successive set of records G_i find O(k) centers in G_i .
- Assign each point in G_i to its closest center.
- Let G' be the centers for $G_1 \ldots G_2 \ldots G_m$.
- Each center is weighted by the number of points assigned to it.
- Cluster G' to find k centers.



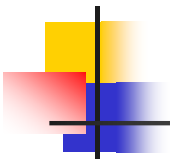
Ensemble-Based Approach: Boosting

- Freund and Schapire, 1995
- Highly accurate ensemble of weak classifiers
- Learn a classifier
- Change the ``focus'' (distribution) of the learner on the data subset that is hard to predict
- Generate a new model for this subset and continue this process until a weighted aggregation of the classifiers provide high accuracy



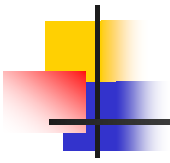
Bagging, Arching, Stacking, and Littlestone's Algorithm

- Bagging (Breiman, 1996)
  - Generate multiple models from subsets of data chosen uniformly with replacement
  - Average the outputs of the models
- Arching (Breiman, 1996)
- Stacking (Wolpert, 1992)
- The Weighted Majority Algorithm (Littlestone and Warmuth, 1990)



Boosting-Based Stream Mining

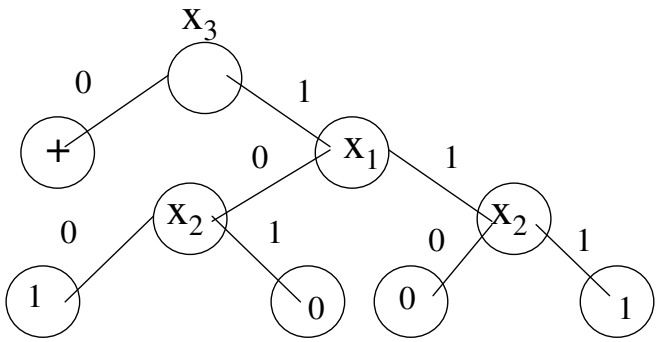
- Fan, Stolfo, and Zhang, 1998
- Learn multiple models from different incoming blocks using the Boosting framework
- Create an ensemble based on the boosting weights



Ensembles and Data Mining

- Better Accuracy with ensemble of models
- But, ensembles are hard to decipher
- On-line learning sees data incrementally and that may create multiple models that can be functionally represented by a single model

Aggregation of Multiple Decision Trees Using Fourier Analysis



Represent the symbols by integer numbers.
X_1 (large = 0, small = 1)
X_2 (red = 0, blue = 1)
X_3 (high = 0, low = 1)
Class (+ = 1, - = 0)

Decision tree can be viewed as a function

Function Representation

- Any function can be written in the decomposed and distributed form using a set of basis functions $f(x) = \sum_{k \in J} w_k \psi_k(x)$
- where,
  - J is an indexed set
  - w_k is the k-th coefficient; $w_k = \sum_x f(x) \psi_k(x)$
  - $\psi_k(x)$ is the k-th basis function.
- Fourier basis is particularly useful for decision trees: For example, in binary representation:
  $\psi_k(x) = (-1)^{k \cdot x}$

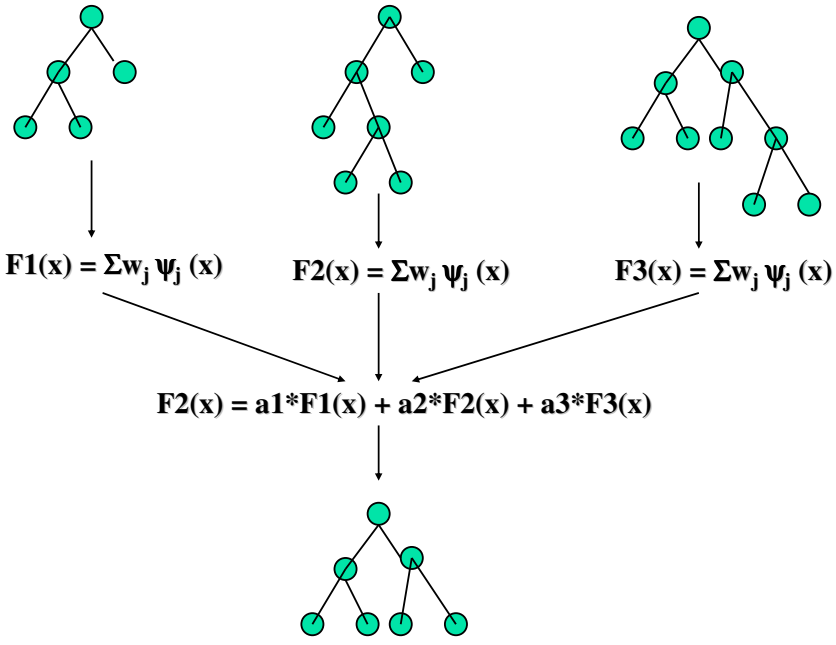
Discrete Fourier Spectrum of a Decision Tree

•Very sparse representation; polynomial number of non-zero coefficients. If k is the depth then all coefficients involving more than k features are zero.

•Higher order coefficients are exponentially smaller compared to the low order coefficients (Kushlewitz and Mansour, 1990).

•Can be approximated by the coefficients with significant magnitude.

Aggregation of Multiple Decision Trees

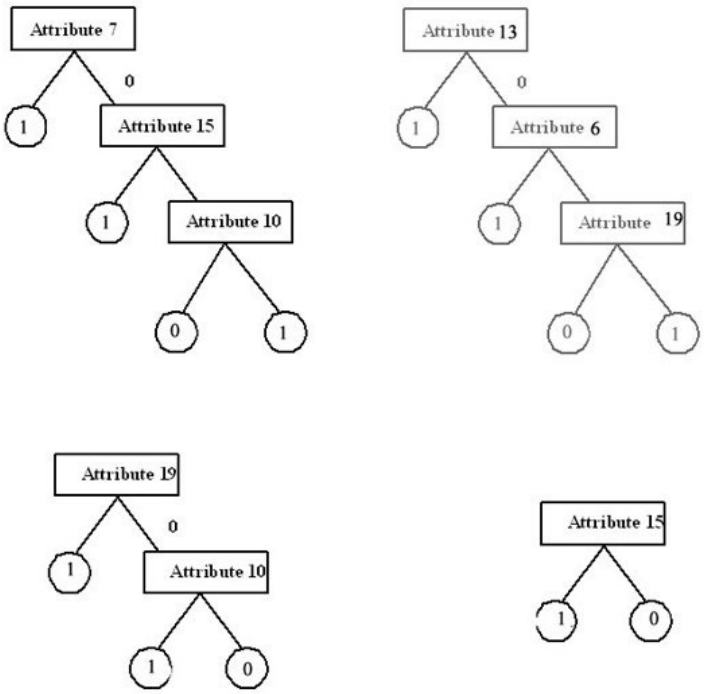


- Weighted average of decision trees through Fourier analysis

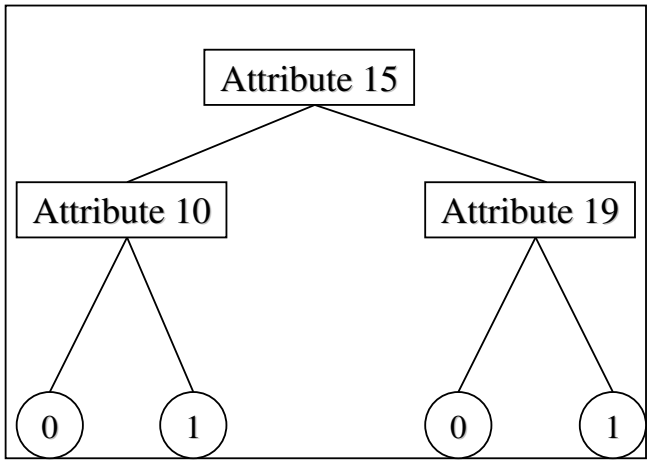
Orthogonal Decision Trees for Resource Constrained Applications

- Kargupta, Dutta, 2004
- Represent the trees in an ensemble using Fourier bases
- Perform PCA of the ensemble
- Eigenvectors represent a Fourier spectra of a decision tree
- Construct a tree from each eigenvector
- These trees are functionally orthogonal to each other and constitute a redundancy-free ensemble.

An Ensemble of Decision Trees

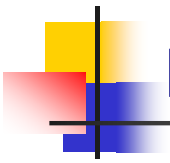


An Orthogonal Tree Generated from the Ensemble



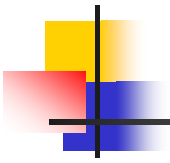
Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources



Different Types of DDM Environments

- Homogeneous Sites: Sites observing a common set of features (Stolfo, et al., 1997, Yike, et al.1998)
- Heterogeneous Sites: Sites observing different feature sets



Homogeneous Data Sites

Account Number	Amount	Location	History	Earning
11992346	99.84	Seattle	Good	High
12999333	29.33	Seattle	Good	High
45633341	34.89	Portland	Okay	Low
55567999	980	Spokane	Good	Low

Account Number	Amount	Location	History	Earning
87992364	20	Chicago	Good	Low
67845921	447	Urbana	Good	Low
85621341	19.78	Chicago	Okay	High
95345998	800	Peoria	bad	High

Different sites observe same features for different events

Heterogeneous Sites

State	Movie	Rating	Revenue
WA	Hyper Space	A+	6M
ID	Once Upon a Time	B-	2M
BC	The King and the Liar	B+	8M
CA	The Shepard	A-	10M

City	State	Size	Avg. earning	Teen pop.
Lewiston	ID	Small	Low	5K
Spokane	WA	Medium	Medium	30K
Seattle	WA	Large	High	250K
Portland	OR	Large	High	200K
Vancouver	BC	Medium	Medium	199K

Different sites observing different feature sets

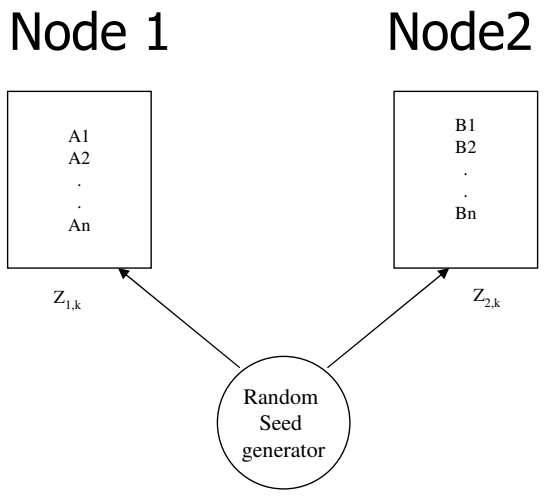
Quick Detection of Patterns Using DDM Algorithms

- Distributed Query Processing
- Distributed data pre-processing
- DDM Algorithms
  - Distributed association rule learning
  - Collective decision tree learning
  - Collective PCA and PCA-based clustering
  - Distributed hierarchical clustering
  - Other distributed clustering algorithms
  - Collective Bayesian network learning
  - Collective multi-variate regression
  - Distributed support vector machine learning
  - Distributed construction ensemble models
  - Ensemble-based aggregation

Comparing Observations for Target Tracking

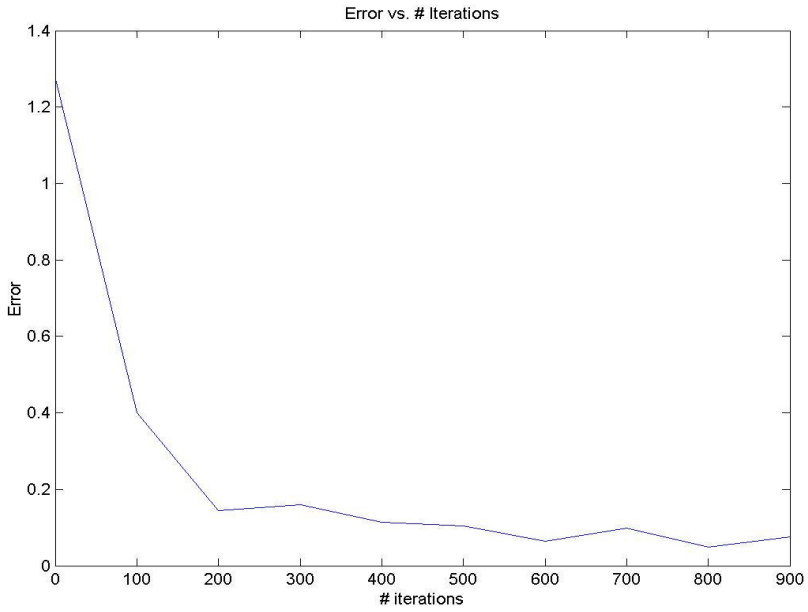
- Given two sequences observed at two different sensor nodes
- Compute the similarity between these sequences
- Inner product computation
  - Related to Euclidean distance, Hamming distance, Correlation coefficient and many other computing problems

Distributed Randomized Inner Product Computation



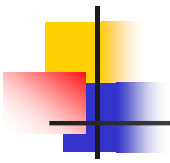
- Node 1 computes Z_{1k}
  - $Z_{1k}=A1.J_1+..+An.J_n$
  - $J_i \in \{+1,-1\}$ with uniform probability
- Nodee 2 calculates Z_{2k}
  - $Z_{2k}=B1.J_1+..+Bn.J_n$
- Compute $z_{1k}.z_{2k}$ for a few times and take the average

Relative Error vs. Communication Cost



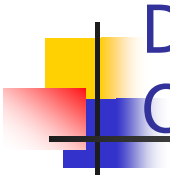
Random Sampling from a Sensor Network

- How to efficiently collect a random sample of data from a sensor network?
- Naïve approach:
  - Contact every node and collect a random sample
  - Complexity of the order of the network-size
- Needs more efficient techniques



More Efficient Sampling Algorithms

- Spatial sampling based on the distribution of the sensor nodes
- Several algorithms exist:
  - Bash, Byers, and Considine, 2004
  - Manku, Rajagopalan, and Lindsay, 1998



Distributed Majority Vote Computation in a Peer-to-Peer Mode

- 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:

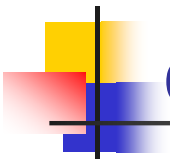
$$\Delta^u = s^u + \sum_{(v,u)\in E^u} sum^{vu} - \lambda\left(c^u + \sum_{(v,u)\in E^u} count^{vu}\right)$$

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

- Update when:
  - S^u changes, a message is received, E^u changes
- Update when:
  - A message is sent to or received from 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.



Continued

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

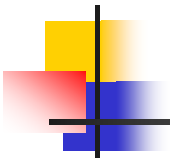
For each $(v,u) \in E^u$

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



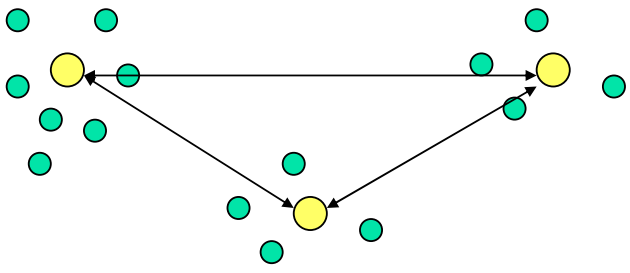
P2P Association Rule Learning

- (Wolff, Schuster, 2003)
- Based on the P2P majority voting algorithm

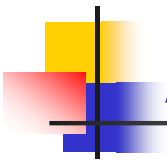
Distributed Association Rule Learning

- Cheung, et al., 1996.
- Compute locally large itemsets.
- Communicate information regarding the locally large itemsets.
- Generate the globally large itemsets.

Distributed k-Means Clustering from Homogeneous Data Nodes

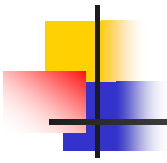


- Local gateways are used as distributed clustering nodes



A Hypothetical Algorithm

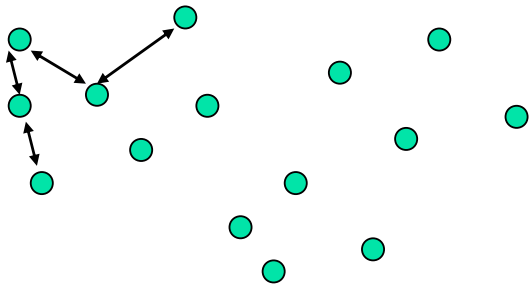
- Compute centroids at each node
- Share the centroids with every other node in the network
- Update the centroids
- Reassign the cluster members
- Repeat the above steps until convergence



Observations

- Exact algorithm that offers results same as the centralized version
- Assumes that the nodes are synchronous
- May work in a hierarchical network topology where local gateways work in tandem with the global base station

Distributed Clustering for Asynchronous Peer-to-Peer Mode



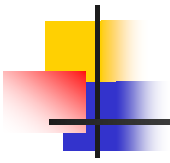
- More decentralized peer-to-peer communication

Algorithm Sketch

- Compute the local centroids
- Receive centroids from a subset of nodes:
  - Only from neighbors in a deterministic manner
  - Only from neighbors in a randomized manner
  - From a randomly chosen subset of nodes in the network that are not necessarily the neighbors
- Update the centroids
- Reassign cluster members in a probabilistic manner with some quantified confidence
- Repeat the process

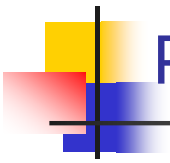
Observations

- Perfectly synchronous scenario gives exact result
- Asynchronous local computation may introduce error:
  - Cluster assignment error
  - Sampling error
- Need for probabilistic assignment of cluster members in a decision theoretic framework
  - Hoeffding, Chernoff bounds, Tail distribution statistics



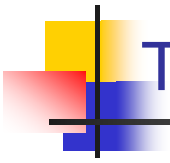
Collective Data Analysis from Heterogeneous Data Sites

- Unsupervised techniques:
  - Collective Principal Component Analysis *(Kargupta, Huang, Sivakumar, Park, and Wang, 2000)*
  - Distributed Clustering *(Johnson and Kargupta, 1999; Johnson and Kargupta, 2000)*
  - Collective Bayesian Learning *(Chen, Sivakumar, Kargupta, 2003)*
- Supervised techniques:
  - Collective Learning of decision trees *(Kargupta, Park, Hershberger, and Johnson, 1999).*
  - Statistical Modeling Techniques: Collective Multivariate regression using wavelets *(Hershberger and Kargupta,2000)*



Principal Component Analysis (PCA)

- A statistical technique to construct a ``concise'' description of data using the dominant eigen vectors of the covariance matrix.
- Very useful for high dimensional data analysis, e.g. text mining.
- Well understood techniques for PCA: Work only when the data are centrally stored.



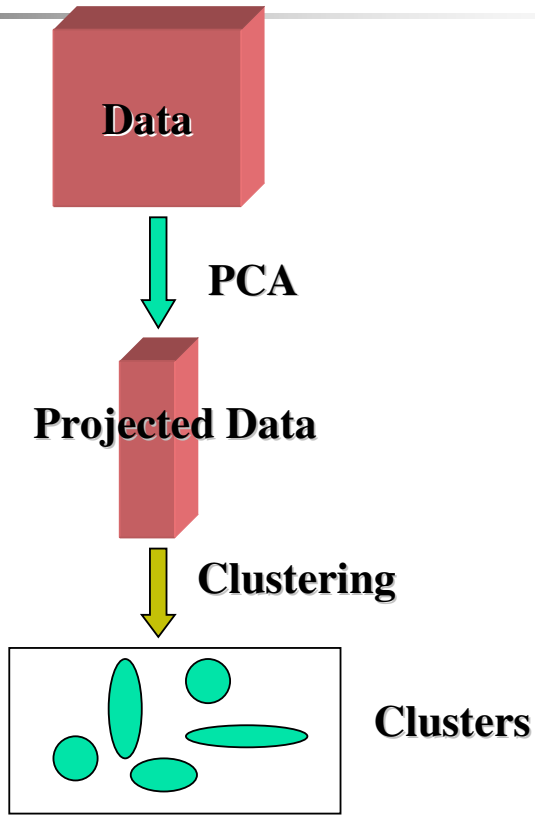
The Collective PCA (CPCA)

- Perform local PCA at each site; select dominant eigen vectors, and project the data on them.
- Send a sample of the projected data to the central site and the eigen vectors.
- Combine the projected data from all the sites.
- Perform PCA on this global data set, identify the dominant eigen vectors, and transform them back to the original space.

Bounding the Error

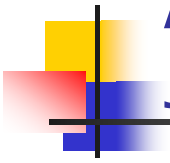
- Error introduced by neglecting the insignificant eigen vectors
- Error Introduced by the sampling of projected data
- Perturbation analysis

PCA-Based Clustering



Clustering

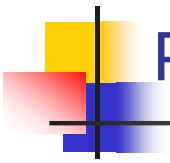
- Assumes existence of a distance metric.
- Requires computation of distances in between data points
- Finding the global distance between a pair of points requires computing the projection of portions of a data row along the global principal directions.
- How to do that in a distributed environment?



A Solution with Theoretical Justification

- The data set distributed among s sites $X = [X_1, X_2, \ldots\ldots X_s]$.
- The eigen vectors, $V = [V_1', V_2', \ldots\ldots V_s']$ ′where V_i is an $n_i \times p$ matrix; n_i is the number of features at the i-th site and p is the number of selected eigen vectors.
- Let $U = [U_1', U_2', \ldots\ldots U_s']$ ′be an $n_i \times (n-p)$ matrix corresponding to the neglected eigen vectors.
- Consider two data rows:

  $\mathbf{x} = [x_1, x_2, \ldots\ldots x_s]$ and $\mathbf{y} = [y_1, y_2, \ldots\ldots y_s]$.
- $\mathbf{x} - \mathbf{y} = \mathbf{z} = [z_1, z_2, \ldots\ldots z_s]$.



Partial Projection is Good Enough

- Projection of **z** using all eigen vectors,

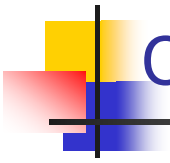
$$\mathbf{z}\,[\,\mathbf{V}\;\;\mathbf{U}\,] = [\,\underline{\mathbf{z}}_1\;\;\underline{\mathbf{w}}_1\,] + [\,\underline{\mathbf{z}}_2\;\;\underline{\mathbf{w}}_2\,]$$

  where $\underline{\mathbf{z}}_i = \mathbf{z}_i\,\mathbf{V}_i$ and $\underline{\mathbf{w}}_i = \mathbf{w}_i\,\mathbf{U}_i$

- The Euclidean distance between **x** and **y**:

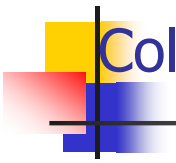
$$||\,\mathbf{x} - \mathbf{y}\,||^2 = ||\,\mathbf{z}\,||^2 = ||\,\underline{\mathbf{z}}_1\,||^2 + ||\,\underline{\mathbf{w}}_1\,||^2 + ||\,\underline{\mathbf{z}}_2\,||^2 + ||\,\underline{\mathbf{w}}_2\,||^2$$

- $||\,\mathbf{z}\,||^2 - (\,||\,\underline{\mathbf{z}}_1\,||^2 + ||\,\underline{\mathbf{z}}_2\,||^2\,) = ||\,\underline{\mathbf{w}}_1\,||^2 + ||\,\underline{\mathbf{w}}_2\,||^2$



Observation

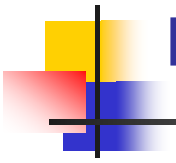
- The global distance between any two data points can be accurately approximated by adding the local distances.



Collective PCA-Based Distributed Clustering

Let C be a given centralized clustering algorithm.

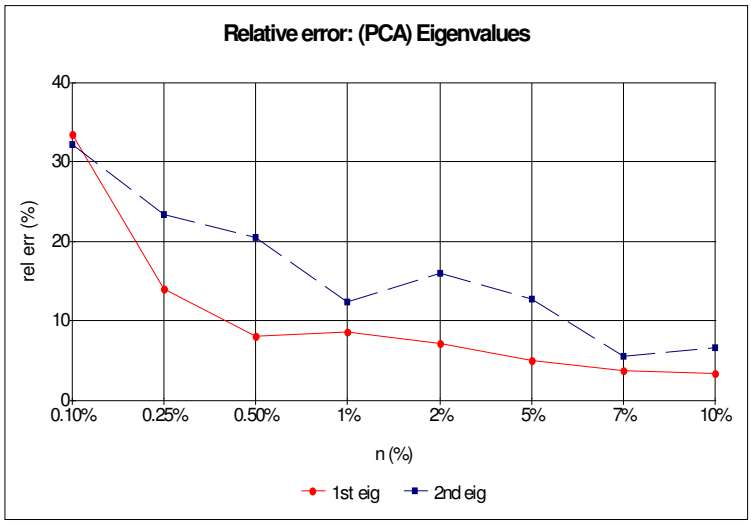
- Perform PCA followed by application of C at every site.
- Select a set of ``representative'' points from each local cluster and send them to the central site.
- Perform global PCA and broadcast the eigen vectors.
- Apply C at each site on the projection of the data and send the cluster-descriptions to the central site.
- Combine the clusters in a linearly decomposable fashion.



Random Projection-Based Approach

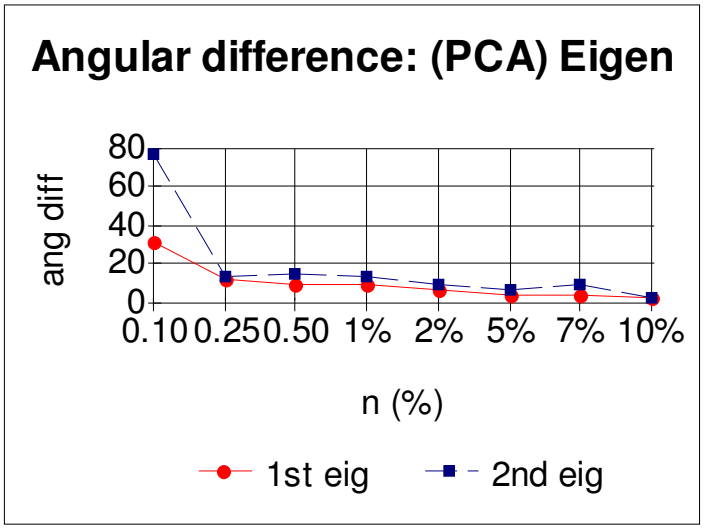
- Random projection for distributed inner product computation
- Can be used for computing the covariance matrix needed for performing PCA

Distributed PCA

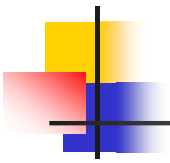


- Average relative error (%) in estimating the top-2 eigenvalues. The mean is computed over 5 independent runs.

Continued..

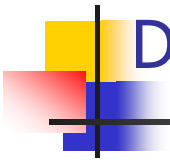


- Mean difference in angle (in degrees) between the actual and the estimated top-2 eigenvectors. The value of *n* (in percentage) reflects the communication cost for the corresponding accuracy. The mean is computed over 5 runs.



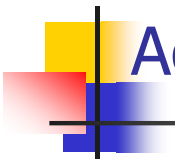
Observations

- Useful random projection-based technique for distributed computation
- Often more efficient than Collective PCA



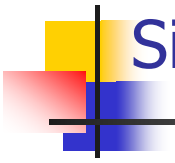
Distributed Hierarchical Clustering

- Distributed SLINK and other hierarchical clustering algorithms
- Heterogeneous Case: (Johnson, Kargupta, 1999)
- Homogeneous Case: (Samatova, et al., 2001)



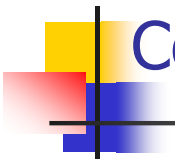
Agglomerative Hierarchical Clustering

- Transforms a set of points into a *dendrogram*
- Uses some metric of dissimilarity or distance (e.g., Euclidean distance)
- Points form the leaves of the tree
- Internal nodes represent partitions of the data



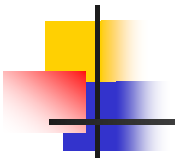
Single Link Clustering

- Initialize each individual point as a single cluster.
- Determine the shortest distance between two clusters, and merge these clusters.
- Continue to merge the least dissimilar clusters until all points are contained in a single cluster.
- SLINK algorithm (Sibson, 1973) $O(n^2)$ time and $O(n)$ space.



Communication Issues

- Cost to transmit the entire dataset is $O(mn)$ where n is the number of rows and m the number of features.
- However, if we transmit locally generated models to a facilitator site, the transmission costs are $O(n|S|)$, where $|S|$ is the number of sites.
- Need to assimilate the locally generated dendrograms.

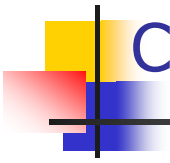


Bounding The Distances In The Dendrogram

- Not enough information in the dendrograms to determine the exact distance between two points.
- Lower and an upper bound on the distance between any two given points can be computed from the dendograms.
- Bounds for estimating the global distance and hence aggregation of the local dendograms.

Other Distributed Clustering Algorithms

- Distributed clustering algorithms
  - Parthasarathy and Ogihara, 2000
  - Dhilon and Modha, 1999
  - McClean et al., 2000
  - Sayal and Scheuermann, 2000
  - Forman and Bin Zhang, 2000



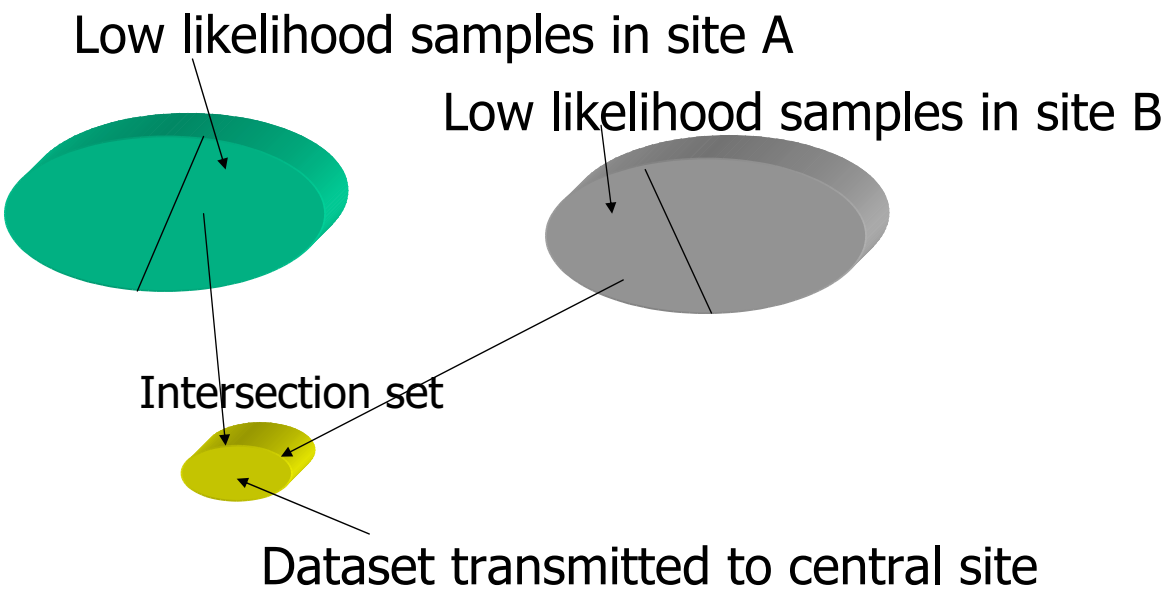
Collective Bayesian Network Learning

- Compute local BN using local dataset
- At each site, identify observations that are likely to be evidence of cross terms. Transmit part of these samples to central site
- Compute a non-local BN using these samples
- Combine local BN and non-local BN to get the collective BN

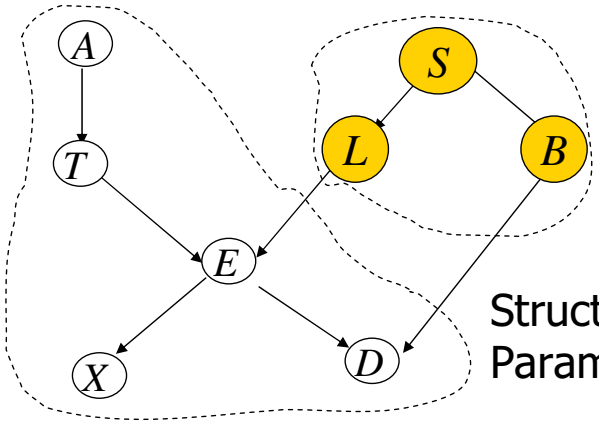
Data Filtering

- After local learning, we have a local BN at each site.
- Compute the likelihood of samples in local sites (under the local BN model)
- Samples with relatively low likelihood do not fit well with local model --- likely to be evidence of cross links.

Data Filtering



Example: ASIA model



Structure: Network topology
Parameter: Conditional Probabilities

Data Filtering: Intuition

- Consider ASIA model

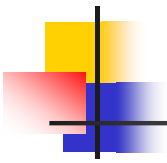
$$P(\mathcal{V}) = P(\mathcal{A},\mathcal{B}) = P(\mathcal{A} \mid \mathcal{B})P(\mathcal{B}) = P(\mathcal{A} \mid nb(\mathcal{A}))P(\mathcal{B})$$

$$P(\mathcal{A} \mid nb(\mathcal{A})) = P(A)P(T \mid A)P(X \mid E)P(E \mid T, L)P(D \mid E, B)$$

Similarly,

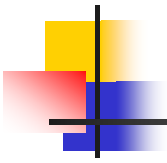
$$P(\mathcal{V}) = P(\mathcal{A},\mathcal{B}) = P(\mathcal{B} \mid \mathcal{A})P(\mathcal{A}) = P(\mathcal{B} \mid nb(\mathcal{B}))P(\mathcal{A})$$

$$P(\mathcal{B} \mid nb(\mathcal{B})) = P(S)P(B \mid S)P(L \mid S)P(E \mid T, L)P(D \mid E, B)$$



Data Filtering: Intuition

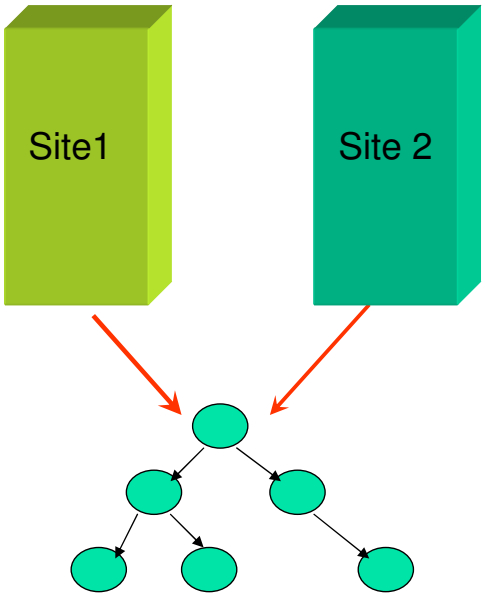
- If both $P(\mathcal{A})$ and $P(\mathcal{B})$ are small for a given row, then both $P(\mathcal{A}|nb(\mathcal{A}))$ and $P(\mathcal{B}|nb(\mathcal{B}))$ are expected to be large.
- Terms common to both $P(\mathcal{A}|nb(\mathcal{A}))$ and $P(\mathcal{B}|nb(\mathcal{B}))$ are precisely the cross terms.



Distributed BN learning for data streams

- Learn an initial collective BN using first dataset
- At step K
  - Update local BN
  - Update threshold (for low-likelihood) in local site
  - Transmit selected data to central site
  - Update parameters of cross term
  - Combine local and cross terms to get a updated collective BN
- K=K+1, repeat step 2 for new data

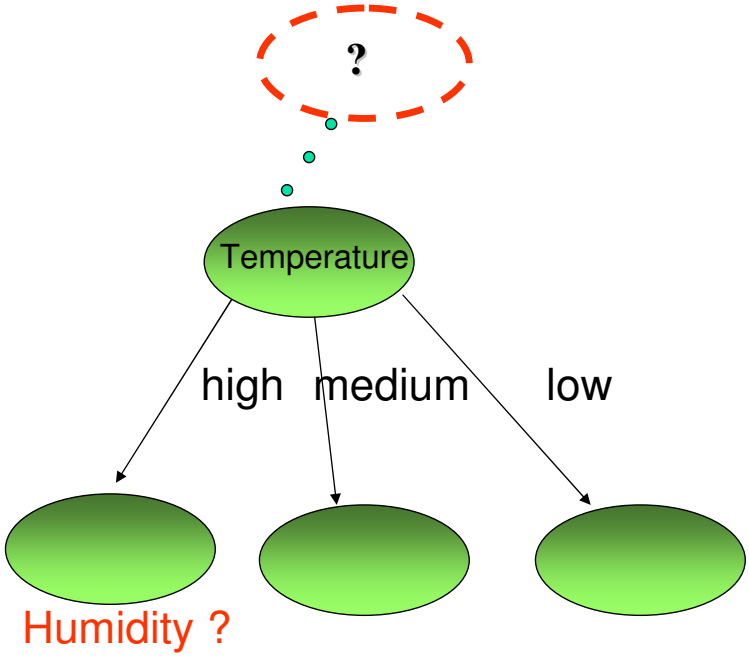
Decision Tree Induction From Vertically Partitioned Distributed Data



Heterogeneous DDM and Decision Trees

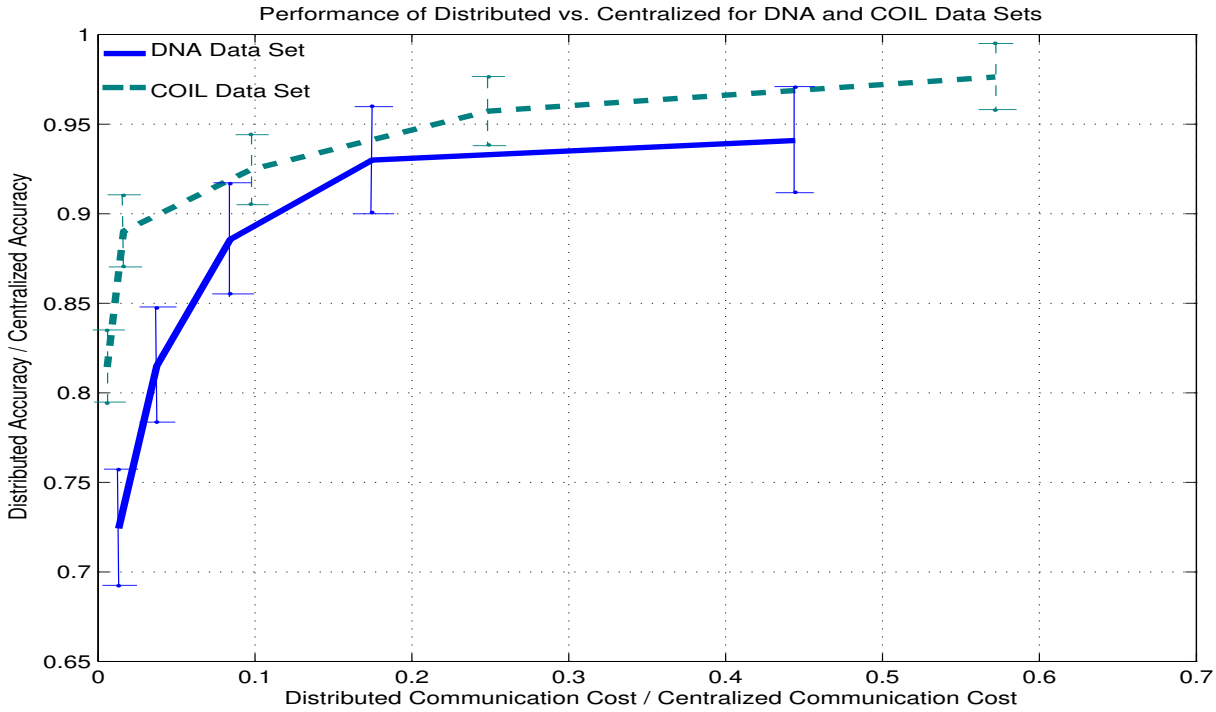
- Distributed Randomized Inner Product (DRIP) computation (Giannella, Liu, Olsen, Kargupta, 2004)
- Computing information gain using DRIP.
- Information gain computation can be posed as an inner product computation problem.

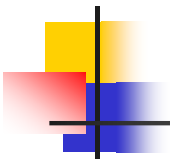
Information Gain Computation and DRIP



Information_Gain(Temp_high,Humidity) = ?

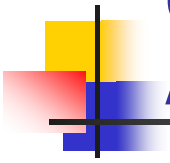
Experimental Results





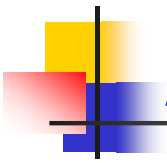
Distributed Multivariate Regression

- Take wavelet transformation of the data
- Communicate a small fraction of the significant data
- Perform regression on the wavelet coefficients
- Original regression coefficients remain invariant under wavelet transformation.



Order Statistics and Model Aggregation

- Order statistics offers a robust way to combine multiple models (Tummer and Ghosh, 2000) in an ordinal manner.
- It works better when the models have high variances.
- Particularly suitable for heterogeneous model aggregation.



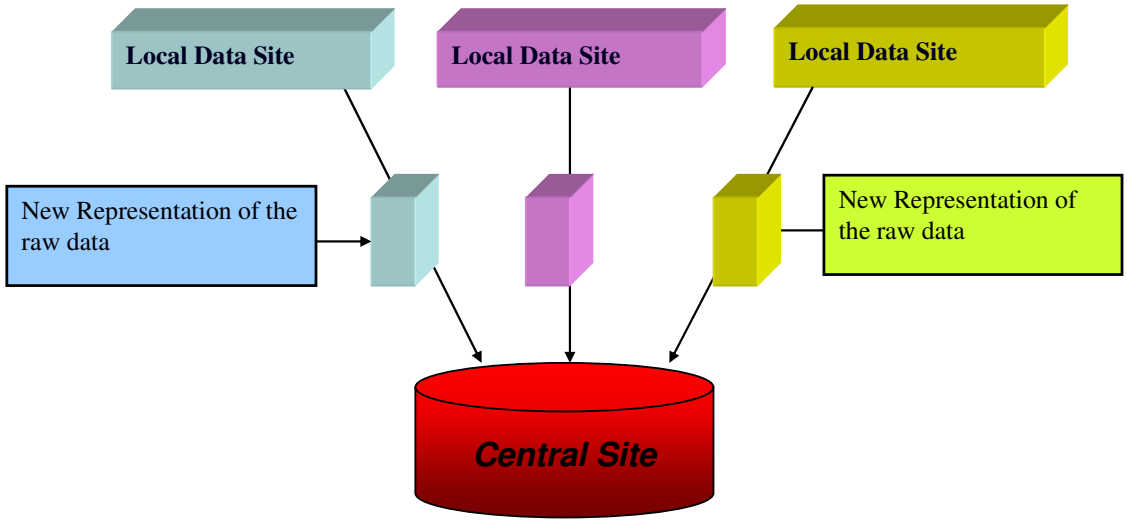
Active Mining

- Databases change frequently. Fast awareness about the changes is important in a DDM environment.
- Active mining (Parthasarahy et al., 2000) proposes several techniques to generate summary statistics at local sites.
- Summary statistics can be exchanged among the sites in order to gain understanding about incoming data.

Privacy Sensitive Distributed Data Mining

- Privacy-sensitive data.
- Multi-party ownership of data stored at different locations.
- So collecting even part of the raw data sets from different locations is not acceptable.

Data Transformation-Based Approach



Perturbation-based Transformations

- Random additive noise (Aggrwal and Srikant, 2000; Kargupta et al, 2003)
- Random multiplicative noise (Liu, Kargupta, 2004)
- Random swapping (Estivill-Castro, Brankovic, 1999).

Secured Multi-Party Computation (SMC) Approach

- Algorithms that allow different participating nodes to compute different aggregates that can be combined to mine the data without sharing the raw data.
- Examples:
  - Inner product computation
  - Association rule learning
  - Clustering
  - Decision tree learning

Illustration: Inner Product Computation

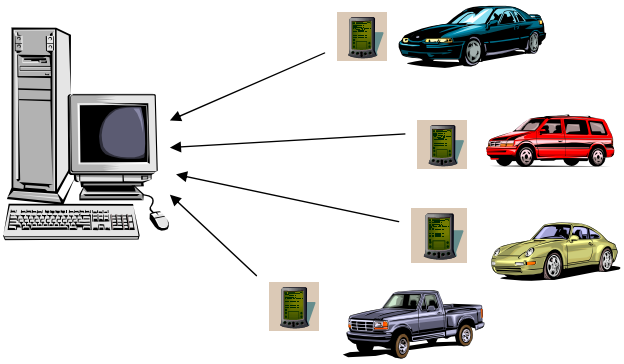
- The privacy-management server generates two random vectors R_a and R_b of size *n*, and let $r_a + r_b =< R_a, R_b >$, where (or r_b) is a randomly generated number. Then the server sends (R_a, r_a) to A, and (R_b, r_b) to B.
- Site A sends $\hat{x}_1 = x_1 + R_a$ to site B, and B sends $\hat{x}_2 = x_2 + R_b$ to site A.
- Site B generates a random number v_2, computes $\langle \hat{x}_1, x_2 \rangle + \langle r_b - v_2 \rangle$, and then sends the result to A.
- Site A computes $\left(\langle \hat{x}_1, x_2 \rangle + (rb - v_2)\right) - \langle R_a, \hat{x}_2 \rangle + r_a = \langle x_1, x_2 \rangle - v_2 = v_1$
- Sites A and B send v_1 and v_2 respectively to the node that needs the inner product and the inner product is $v_1 + v_1$

Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources

MineFleet: A Vehicle Data Stream Management and Mining Software System

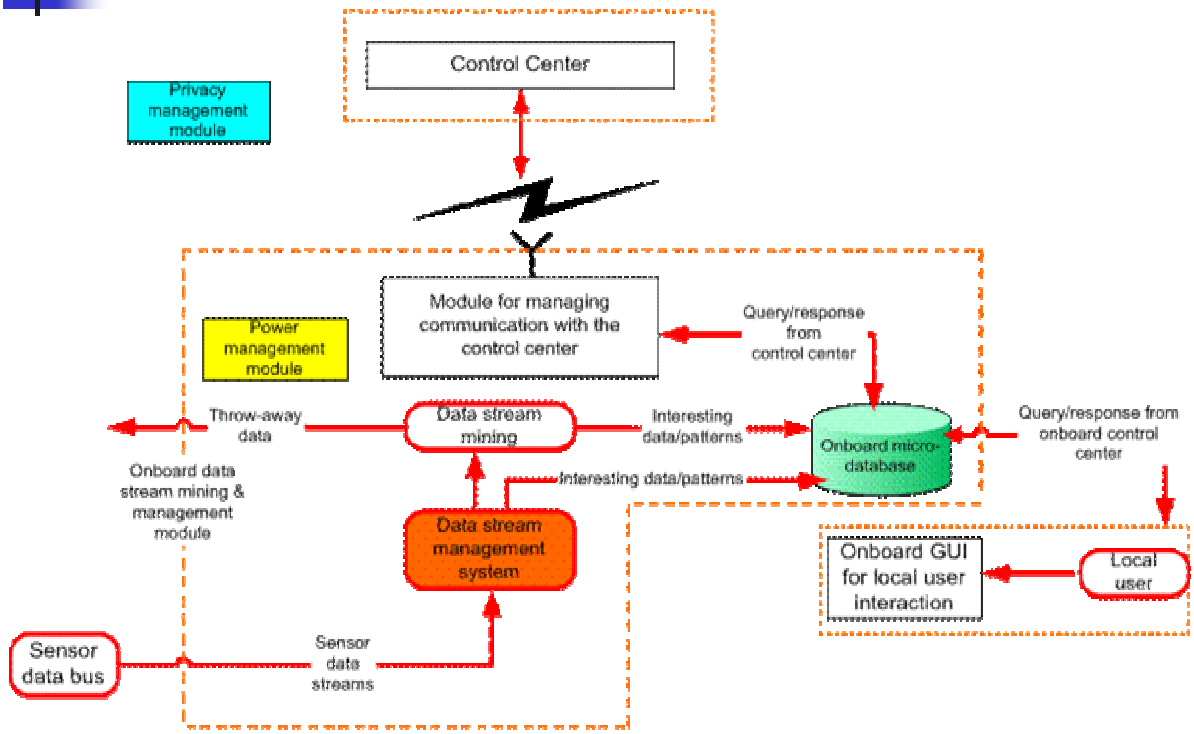
- On-board Module:
  - Continuous data streams from the vehicle data bus
  - Onboard data stream mining
  - Communicates with a remote control station
  - Privacy management
- Central control station:
  - Data Management
  - Data mining
  - Communicates with the on-board modules over wireless networks
  - Privacy management



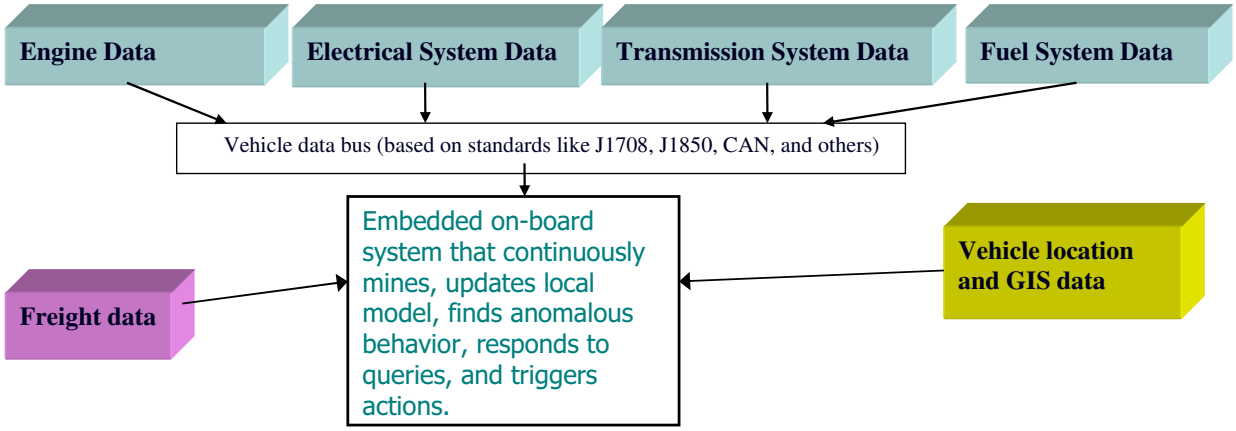
Vehicle Data Stream Management and Mining

- Supports:
  - Real-time vehicle health monitoring
  - Real-time privacy-sensitive vehicle/driver behavior characterization and monitoring
  - Fleet monitoring
- Modalities of data analysis
  - Control station-based
  - On-board
- Important Characteristics
  - Resource constrained computation
  - Distributed computation
  - Privacy sensitive

Architecture of MineFleet



Onboard Vehicle Data Streams

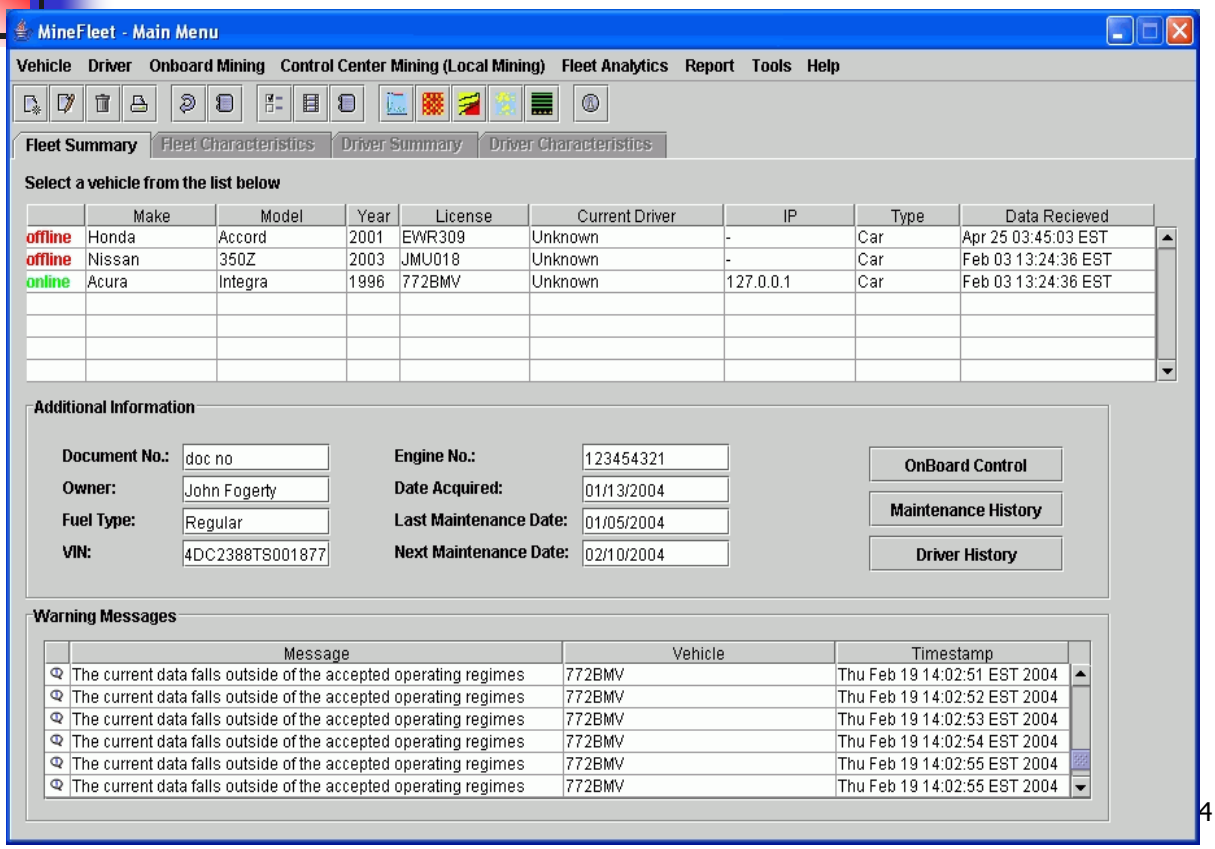


Continuous Data Streams

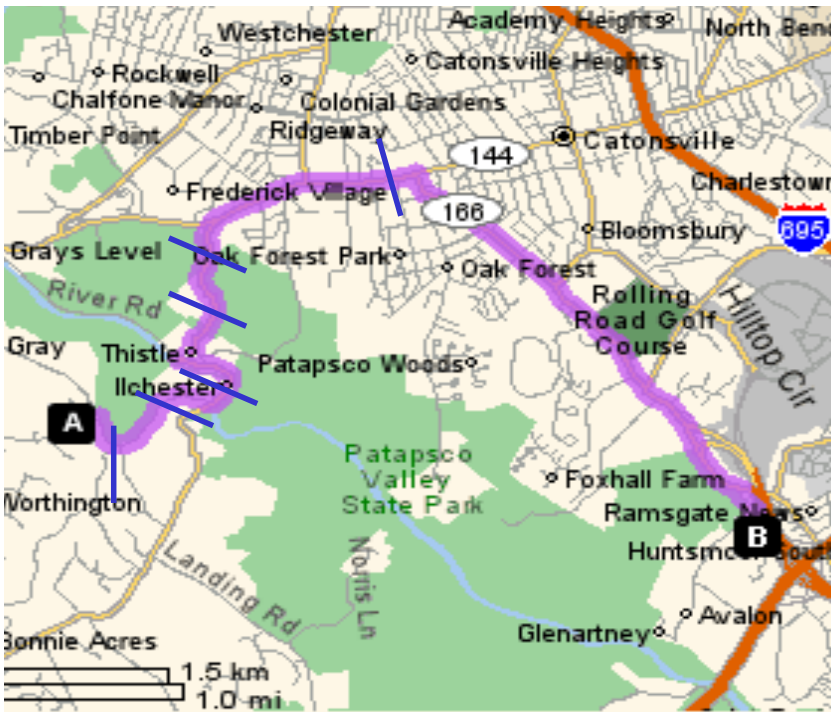
- Depends on the type of vehicle
  - Hundreds of attributes
  - Approximately one tuple per every few (e.g. 4) seconds.
- Continuous-valued attributes
- Fault-codes are Boolean

On-Board Hardware Modules (Academic Version)

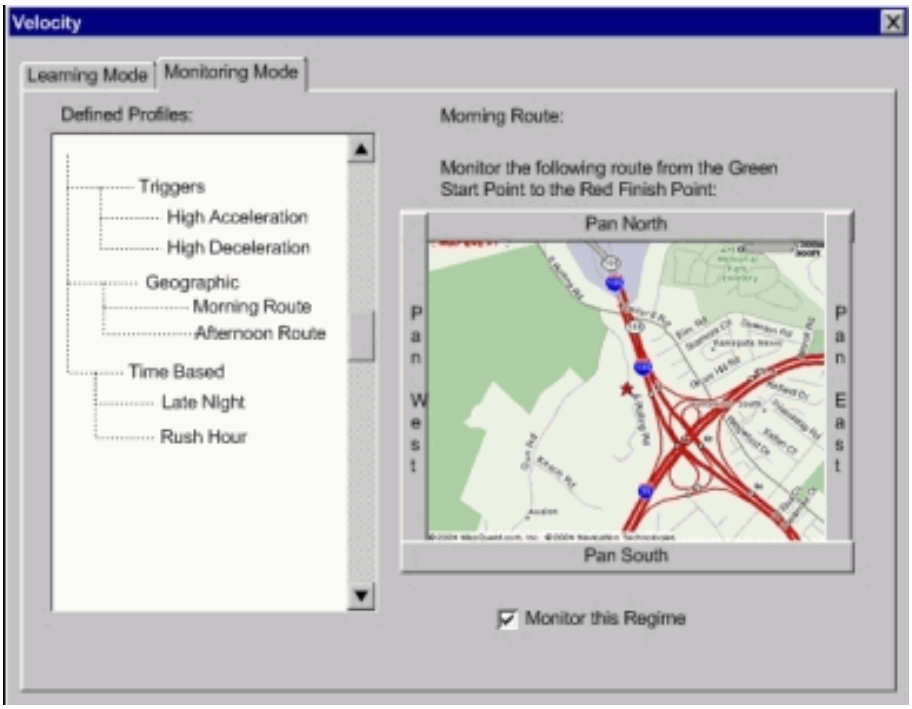
Fleet View



Trajectory Tracing and Monitoring



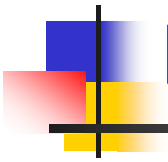
Varieties of Trigger Settings



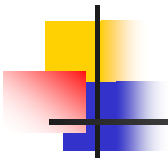
Identifying Typical Operating Regimes

- Drive around and generate typical operating regimes using a PCA-based approach
- Project normal operating data in low dimensional space using principal component analysis
- Identify clusters in the projected data as safe regimes
- Represent these safe regimes using polygons

Distributed Vehicle Stream Mining Demo

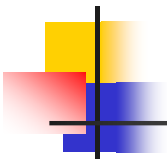


P2P Clustering Demo



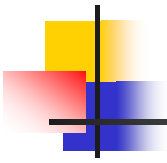
Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources



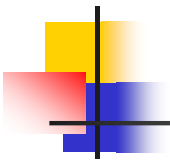
Future Research Directions

- Algorithmic Issues:
  - Peer-to-Peer DDM algorithms for homogeneous and heterogeneous data
  - Privacy-preserving DDM for analyzing distributed privacy-sensitive data
  - DDM and data stream mining for embedded and ubiquitous environments.
  - Designing resource constrained algorithms that are fundamentally designed to handle (a) limited bandwidth, (b) limited computing and storage capabilities, (c) limited battery power, and (d) specific network-communication protocols



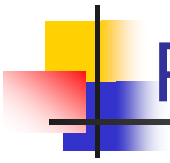
Future Research Directions

- Systems Issues:
  - Developing light-weight DDM application development environments for ubiquitous applications
  - Integration with multi-agent systems
  - Making DDM environments Grid-enabled
  - Integration of DDM systems with distributed data stream management modules



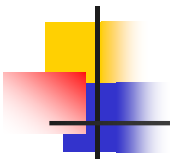
Future Research Directions

- Human-Computer Interaction Issues:
  - Collaborative interaction with sensor-networks and DDM applications
  - Alternate human-computer interfaces for distributed DDM programs that allow non-intrusive pervasive interactions
  - Privacy and social impact from integration of the DDM technology with daily life.



Roadmap

- Distributed Data Mining (DDM): An Overview
- Sensor networks: An Overview
  - Architecture
  - Resource Constraints
  - Applications
- DDM algorithms for Sensor Networks
  - Local data stream mining on-board the sensor nodes
  - Distributed analysis for collective problem solving
- A case study: Mining Vehicle Sensor Networks
- Future directions
- Pointers to more advanced material and resources



Resources

Bibliography:

- DDM Bibliography: http://www.cs.umbc.edu/~hillol/DDMBIB/

Books

- H. Kargupta, A. Joshi, K. Sivakumar, Y. Yesha. Data Mining: Next Generation Challenges and Future Directions.
- H. Kargupta and P. Chan (eds). Advances in Distributed and Parallel Knowledge Discovery, MIT Press, 2000.
- M. J. Zaki and C.-T. Ho (eds). Large-Scale Parallel Data Mining. LNAI State-of-the-Art Survey, Volume 1759, Springer-Verlag, 2000.

Workshops

Ubiquitous Data Mining Workshop:

- PKDD Workshop on Ubiquitous Data Mining for Mobile and Distributed Environments, 2001. http://www.cs.umbc.edu/~hillol/pkdd2001/udm.html

Distributed Data Mining Workshops:

- SIAM Data Mining Conference Workshop on High Performance, Pervasive, and Data Stream Mining (2001, 2002, 2003, 2004)
- ACM SIGKDD Workshop on Distributed Data Mining, 2000. http://www.eecs.wsu.edu/~hillol/DKD/dpkd2000.html
- ACM SIGKDD Workshop on Distributed Data Mining, 1998. http://www.eecs.wsu.edu/~hillol/DDMWS/papers.html

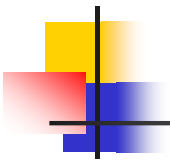
Resources (contd.)

Journal Special Issues

- Special Issue on Distributed and Mobile Data Mining, IEEE Transactions on Systems, Man, and Cybernetics, Part B, November, 2004.
- Special Issue on Knowledge Discovery from Distributed Information Sources Journal of Information Sciences
- Special section on Distributed and Parallel Knowledge Discovery, (Knowledge and Information Systems), 2000.

Survey Articles

- H. Kargupta and K. Sivakumar. Existential Pleasures of Distributed Data Mining. In Data Mining: Next Generation Challenges and Future Directions. MIT/AAAI Press, 2004.
- B. Park and H. Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. To be published in the Data Mining Handbook. Editor: Nong Ye, 2002.
- Distributed and Parallel Knowledge Discovery. Hillol Kargupta and Philip Chan. Advances in Distributed and Parallel Knowledge Discovery, xv--xxvi, MIT/AAAI Press, 2000.
- Distributed Data Mining: Scaling Up and Beyond. Provost, F. Advances in Distributed and Parallel Knowledge Discovery, 3--27, AAAI/MIT Press, 2000.

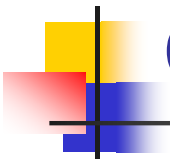


References for DDM Papers

- Please see the DDM bibliography at: http://www.cs.umbc.edu/~hillol/DDMBIB/

References for Other Related Papers Cited in this Presentation

- R. Agrawal and R. Srikant (2000), Privacy-preserving data mining, Proceeding of the ACM SIGMOD Conference on Management of Data', ACM Press, Dallas, Texas, 439--450.
- I. Akyildiz, W. Su, Y. Sankarasubramaniam, and E. Cayirci. (2002) A Survey on Sensor Networks. IEEE Communications, August, 102-114.
- B. Bash, J. Byers, and J. Considine. (2004). Approximately uniform random sampling in sensor networks. Proc. of the 1st Workshop on Data Management for Sensor Networks.
- N. Alon, P. Gibbons, Y. Matias and M. Szegedy. Tracking join and self-join sizes in limited storage, PODS, 1999.
- L. Breiman. (1996). Bagging Predictors. Machine Learning, vol. 24, no. 2, 123—140.
- V. Estivill-Castro, L. Brankovic. (1999). Data swaping: Balancing privacy against precision in mining for logic rules. Proc. of the first Conference on Data Warehousing an Knowledge Discovery, 389-398, Florence, Italy. Springer Verlag.
- M. Datar, A. Gionis, P. Indyk, and R. Motwani. Maintaining Stream Statistics Over Sliding Windows. SIAM Journal on Computing, Vol 31, No. 6, 1794-1813.
- P. Domingos and G. Hulten, (2001). Mining high speed data streams, Proc. of the Sixth International Conference on Knowledge Discovery and Data Mining, 71--80, ACM Press.
- D. Estrin and R. Govindan and J. Heidemann and S. Kumar. (1999). Next century challenges: scalable coordination in sensor networks. Proc. of the 5th annual ACM/IEEE international conference on Mobile computing and and networking. 263-270.

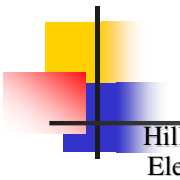


Continued

- W. Fan, S. Stolfo and J. Zhang. (1999). The Application of AdaBoost for Distributed, Scalable and On-line Learning, Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining.
- Y. Freund, Boosting a Weak Learning Algorithm by Majority, Information and Computation, volume 121, no. 2, 256-285.
- Gifford. (1979). Weighted Voting for Replicated Data. Proc. ACM Symp. Operating Systems Principles.
- P. Gibbons. Distinct sampling for highly-accurate answers to distinct values queries and even reports, VLDB 2001.
- H. Kargupta, S. Datta, Q. Wang, and K. Sivakumar. (2003). On the Privacy Preserving Properties of Random Data Perturbation Techniques. Proceedings of the IEEE International Conference on Data Mining. Melbourne, Florida, USA, pages 99-106.
- E. Keogh, J. Lin, and W. Truppel., Clustering of time series subsequences is meaningless: Implications for past and future research, Proc. of the 3rd IEEE International Conference on Data Mining, 115--122, 2003.
- K. Liu, H. Kargupta, and J. Ryan. (2003). Multiplicative Noise, Random Projection, and Privacy Preserving Data Mining from Distributed Multi-Party Data.
- N. Littlestone, (1991). Redundant noisy attributes, attribute errors, and linear threshold learning using Winnow, Proc. 4th Annual Workshop on Computational Learning Theory, Morgan Kaufmann, 147-156.

Continued

- G. Manku, S. Rajagopalan and B. Lindsay. (1998). Approximate medians and other quantiles in one pass and with limited memory, Proc. of ACM SIGMOD, 426-435.
- Y. Zhu and D. Shasha. (2002). StatStream: Statistical Monitoring of Thousands of Data Streams in Real Time, VLDB.
- R. Thomas. (1979). A Majority Consensus Approach to Concurrency Control for Multiple Copy Databases, ACM Trans. Database Systems, vol. 4, 180—209, 1979.
- D. Wolpert. (1992). Stacked Generalization, Neural Networks, volume 5, 241-259.



Hillol Kargupta

Hillol Kargupta is an Associate Professor at the Department of Computer Science and Electrical Engineering, University of Maryland Baltimore County. He received his Ph.D. in Computer Science from University of Illinois at Urbana-Champaign in 1996. He is also a co-founder of AGNIK LLC, a ubiquitous data intelligence company. His research interests include mobile and distributed data mining and computation in gene expression.

Dr. Kargupta won a National Science Foundation CAREER award in 2001 for his research on ubiquitous and distributed data mining. He along with his co-authors received the best paper award in the 2003 IEEE International Conference on Data Mining for a paper on privacy-preserving data mining. He won the 2000 TRW Foundation Award and the 1997 Los Alamos Award for Outstanding Technical Achievement. His dissertation earned him the 1996 Society for Industrial and Applied Mathematics (SIAM) annual best student paper prize.

He has published more than eighty peer-reviewed articles in journals, conferences, and books. He is an associate editor of the IEEE Transactions on Systems, Man, Cybernetics, Part B. He served as the Associate General Chair of the 2003 ACM SIGKDD Conference. He is also the Program Co-Chair of the 2005 SIAM Data Mining Conference. He has co-edited two books: (1) Advances in Distributed and Parallel Knowledge Discovery, AAAI/MIT Press, and (2) Data Mining: Next Generation Challenges and Future Directions, AAAI/MIT Press. He is in the program committee of almost every major data mining conference (e.g. ACM, IEEE, SIAM). He has been a member of the organizing committee of the SIAM data mining conference every year from 2001 until 2005. He hosted many workshops and journal special issues on distributed data mining and other related topics. He regularly serves as an invited speaker in many international conferences and workshops. More information about him can be found at http://www.cs.umbc.edu/~hillol.