A Mathematical and Data-Driven Approach to Intrusion Detection for High-Performance Computing

David H Bailey

Lawrence Berkeley National Laboratory

http://crd.lbl.gov/~dhbailey

With input from

Deborah Agarwal, Juan Meza, Scott Campbell, Orianna DeMasi, Mark Woods (LBNL)

Matt Bishop, Sean Whalen (UC Davis)

Vern Paxson (UCB, ICSI, LBNL)

Sean Peisert (UC Davis, LBNL)

Robin Sommer (ICSI, LBNL)

Funded via DOE/ASCR program on Complex, Distributed, Interconnected Systems

Approach

Challenge: Detect intrusion and misuse of HPC systems.

- Mathematics and statistics:
  - Apply some known mathematical and statistical techniques to perform the equivalent of credit card fraud detection on the access and usage of high-performance computer systems.
  - Investigate some new promising techniques that potentially might be more effective.
- Data:
  - Historical and near-real-time data from NERSC networks and systems.
    - Network connection and protocols.
    - System logs.
    - Process accounting.
    - Batch system and job scheduler data.
    - Integrated Performance Monitoring (IPM) logs.

Funded through the DOE/ASCR program on Complex, Distributed, Interconnected Systems.

The challenge of anomaly detection in HPC cybersecurity

Intrusion/anomaly detection on HPC systems has the advantage of being a specialized arena -- much less diverse than a general computing environment -- but there are also some special challenges as well:

- Outlier detection and the high costs of errors:
  - Finding a stable notion of “normal” is hard in a dynamically changing network and HPC environment.
  - Each mistaken false positive wastes the scarce time of system analysts.
- Interpretation of results:
  - What does that anomaly mean?
- Evaluation of techniques:
  - How do we make sure it actually works?
- Training data:
  - What do we train our system with?
- Evasion risk:
  - Can the attacker mislead our system?

Fingerprinting communication and computation on HPC systems

- How do we identify what is running on HPC systems?
  - Is a job similar to what a user normally runs?
  - Is a job what a user should be running, based on their project?
- We need metrics for application-level data.
- We use MPI calls for now (logged via IPM library).
- In the future, we plan to also use performance counters.

IPM data:

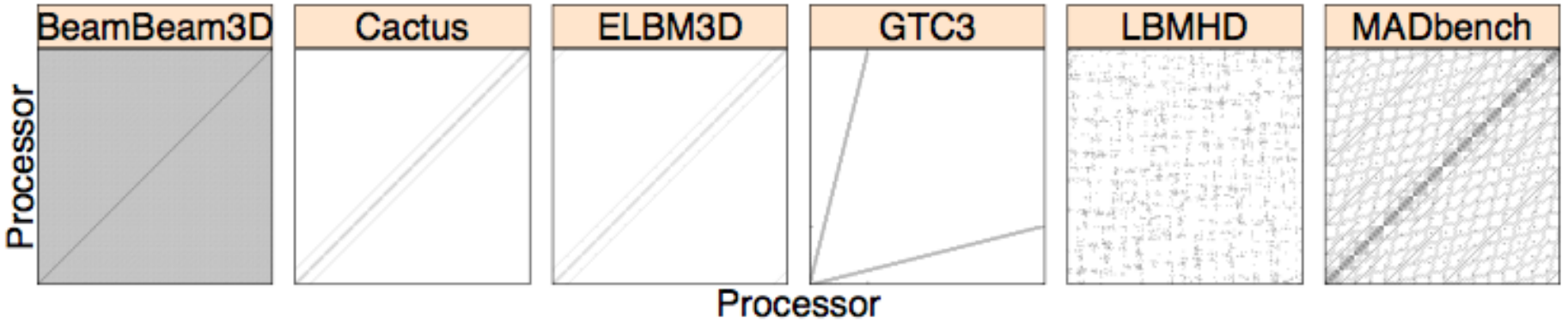
- Current data is aggregated, not ordered.
- Data shows type of call, source and destination nodes, bytes sent, number of times called.

We have used machine learning techniques to attempt to “fingerprint” various applications running on large systems.

Early fingerprinting results

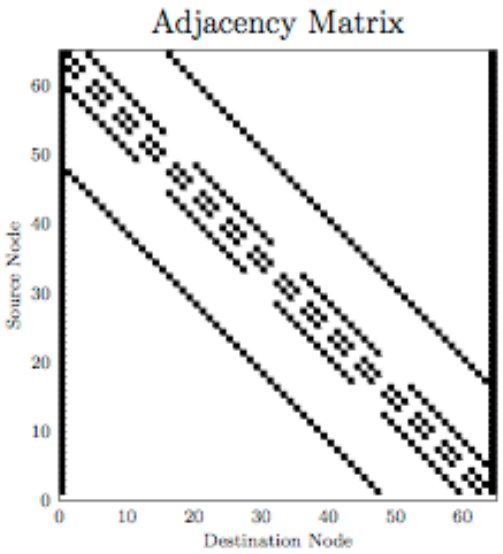
- Machine learning algorithms include kNN, naive Bayes, naive Bayes multinomial, locally weighted learning (LWL), and “boosting.”
- Amount of data currently is very limited, but we have some early results.
- Analyzing data for all nodes of a run were poor -- less than 50% chance of correctly identifying a particular code.
- Analyzing data for the first two nodes of a run (nodes 0 and 1) did much better: kNN, naive Bayes multinomial, and LWL showed accuracy scores of 80%.

Communications characteristics of various numerical computations

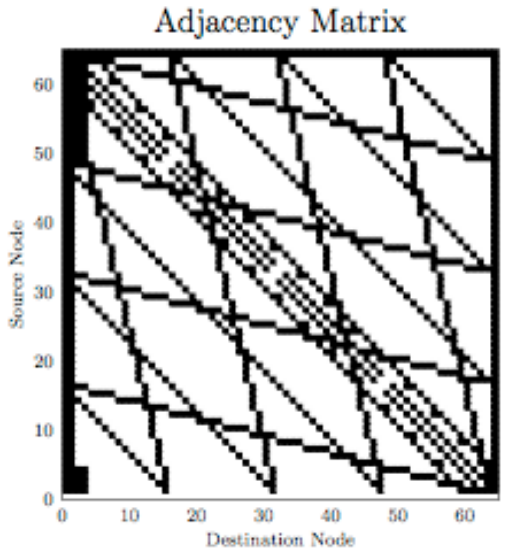


Shoaib Kamil, Ali Pinar, Daniel Gunter, Michael Lijewski, Leonid Oliker, John Shalf, "Reconfigurable Hybrid Interconnection for Static and Dynamic Scientific Applications," available at http://crd.lbl.gov/~oliker/papers/CF07.pdf.

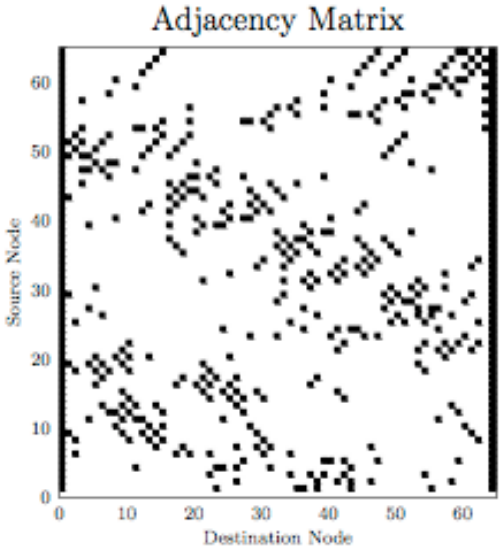
General Relativity



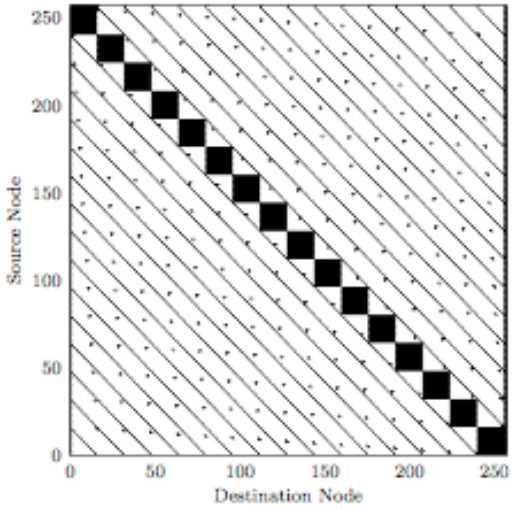
Atmospheric Dynamics



Plasma Physics



Performance Benchmark



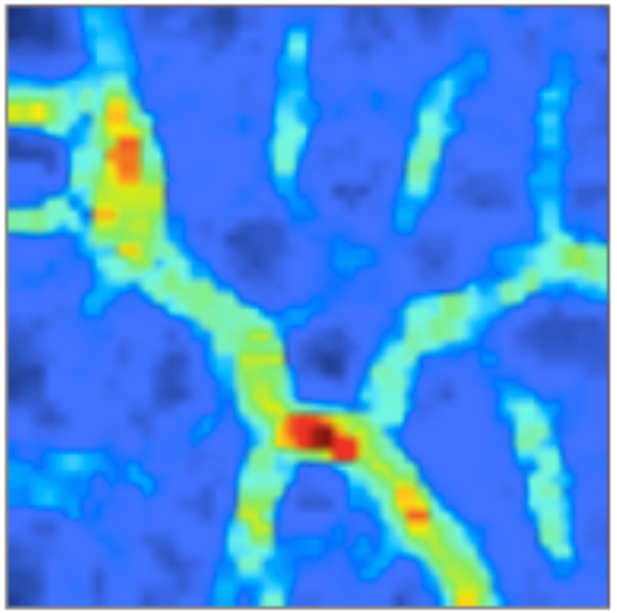
BERKELEY LAB

Second approach: Unsupervised clustering using self-organized maps

- Projects high dimensional input space to two dimensions.
- Preserves topology of input space.
- Few parameters to specify: learning rate, map size.

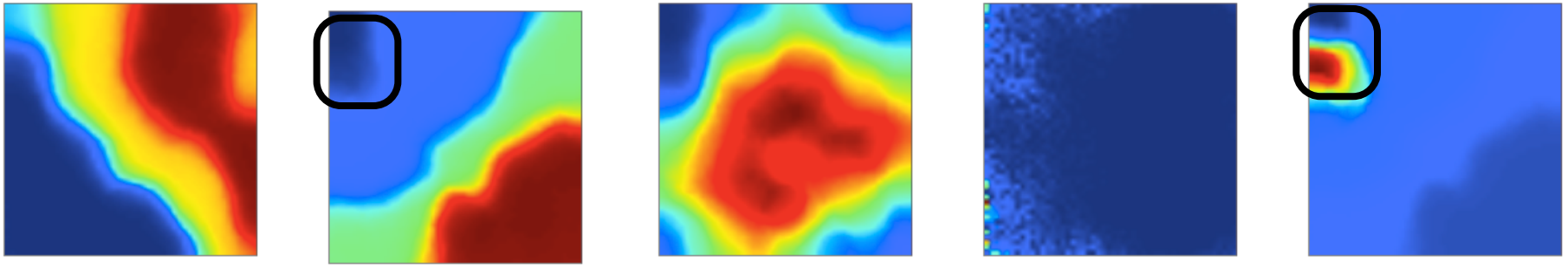
Figure shows the clustering of the input vectors. High intensity(green/yellow/red) lines indicate borders between clusters. There are roughly 6 clusters there.

The advantage of the SOM approach is we don't have to specify the number of clusters -- it finds them for us.



Mapping from high-dimensional feature space to 2-D preserves topological relationships

Function Call | Bytes Sent | Destination | Region | # of Repeats



These figures are graphs for individual features in the input vector: function call, bytes sent, destination node, region code, and number of repeats.

The highlighted features show that the map allows one to visualize correlations between features. Here, the portion of the “number of repeats” graph with high intensity has a correspondingly low intensity in the “bytes sent” graph -- i.e., messages that are often repeated don’t send much data.

Mathematical details of cluster scheme

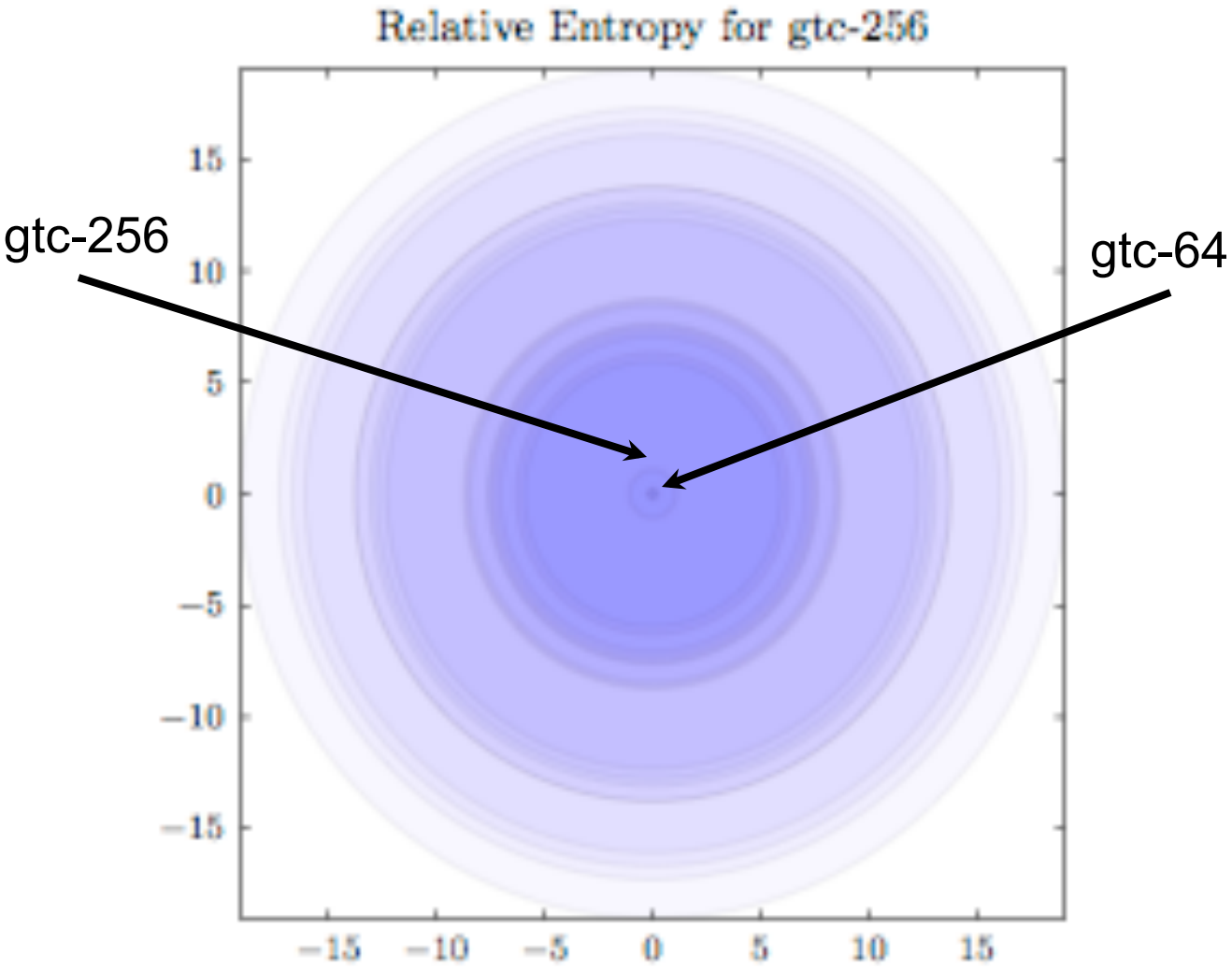
- Pairwise distances between each code's distribution of communications
- Relative entropy (Kullback-Liebler Divergence) between discrete distributions P and Q:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)}$$

Distances visualized as concentric circles

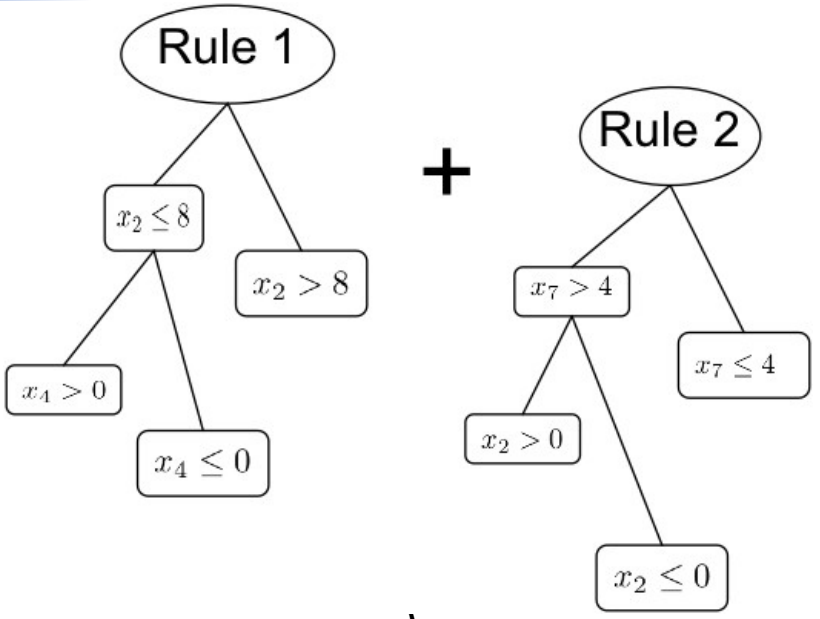
This shows the relative entropy (Kullback-Liebler divergence) between gtc-64 and the other codes in our test data, visualized as concentric circles. Note that gtc-256 is closest to gtc-64 (center point).

This shows that relative entropy is small between runs of a code with different number of nodes, and is large between different codes.



The "rule ensemble" scheme

- Train on data to build a series of decision trees. Each tree is a rule r_k.



- Find coefficients a_k by using steepest descent method to solve the optimization problem:

$$\{\hat{a}_k\}_0^K = \arg\min_{\{\hat{a}_k\}_0^K} \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, a_0 + \sum_{k=1}^{K} a_k r_k(x_i) + \lambda \sum_{k=1}^{K} |a_k|\right)$$

Combine rules into model F(x):

$$F(X) = a_0 + \sum_{k=1}^{K} a_k r_k(X)$$

This scheme has been implemented, and is being tested on both "toy" and real cybersecurity data.

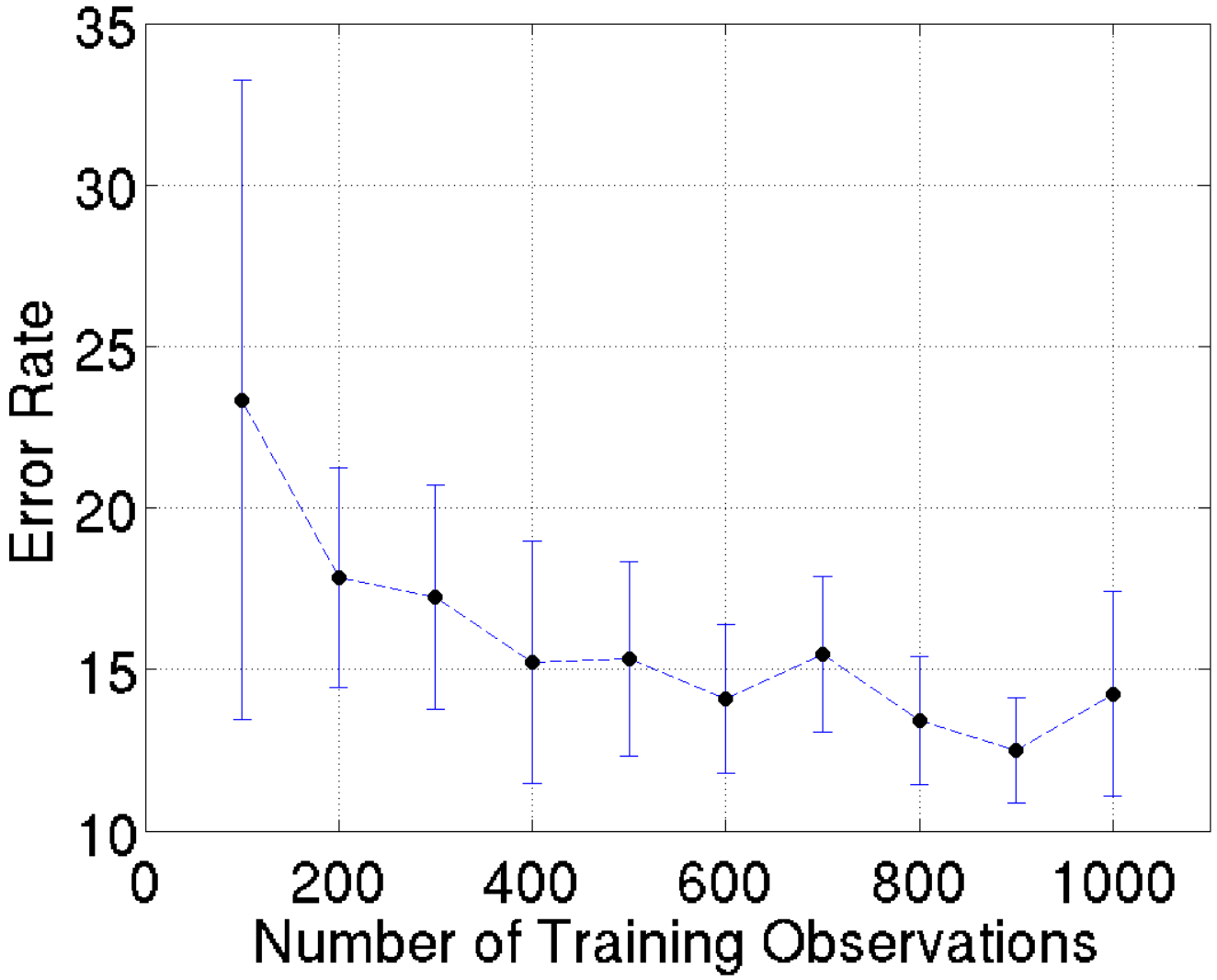
Some challenges in obtaining good data for rule ensemble scheme

- Logs come in unwieldy .xml files (up to 800 MByte for large runs).
- Preprocessing is required, as many GByte of data are required just to get enough observations for the training set.
- Many logs are unlabeled -- called by username not name of the code.
- Scaling is required -- the number of total nodes affects the number of times a call is made and the number of bytes.
- Distinguishing which runs are the duplicates is a challenge -- results may land in both training data and testing data.
- Which information is important? Is it necessary to study the communication behavior of all nodes, or just a few?

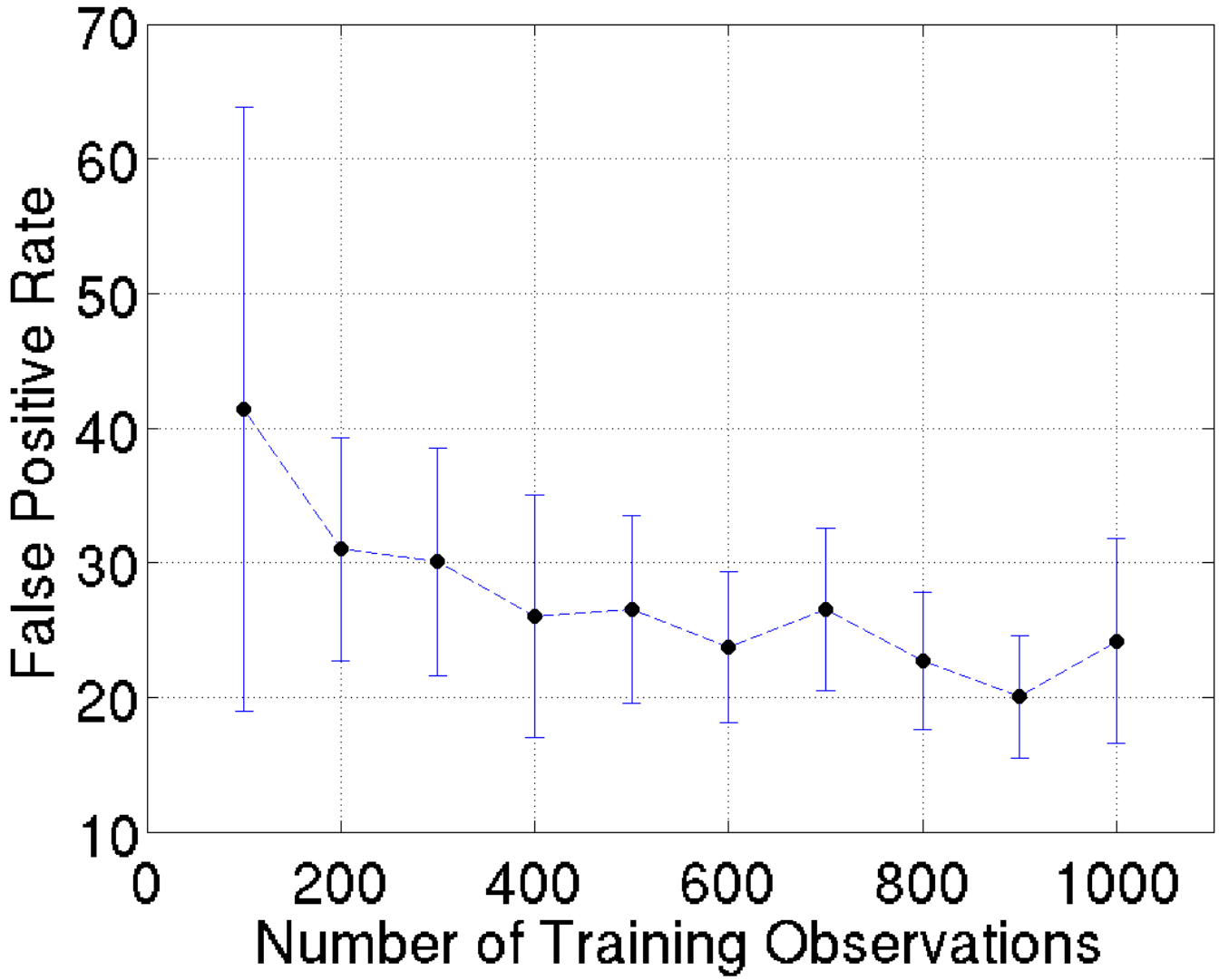
Tests of the rule ensemble scheme using supernova data

- The Nearby Supernova Factory obtains images from the JPL Near Earth Asteroid Tracking (NEAT) project and processes them at the NERSC Parallel Distributed Systems Facility (PDSF). Each object is scored for eccentricity, brightness, signal to noise ratio, and 21 other features.
- 100 decision trees are built for each run.
- Each tree keeps building until a certain number of terminal nodes (drawn from an exponential distribution) is reached.
- In these experiments, the average number of rules used was 6, so since 100 trees were built, on average 600 rules were used per ensemble.
- Each point in the following plots is the average of 25 runs.

Error rates for rule ensemble scheme on supernova data



False positive rates for rule ensemble scheme on supernova data



The challenge of data sanitization

One challenge is how to “redact” data -- remove personal information, so as to facilitate open-community research and development.

- Permission issues -- must have explicit cooperation of those involved.
- Privacy requirements -- determine what can and cannot be revealed.
- Analysis requirements -- what must be revealed to enable appropriate use.
- Threat model -- what access to unredacted data does adversary have?
- What external knowledge does adversary have? (We may not know this.)

Relationships and attacks:

- Attacks succeed by discovering relationships:
  - Within the redacted data set itself.
- Among redacted data and externally available data.
- Describe relationships before sanitizing:
  - Build ontology.
  - Determine what relationships will allow adversary to undo redaction.

M. Bishop, J. Cummins, A. Singh, D. Agarwal, S. Peisert, D. Frincke, and M. Hogarth, “A Data Sanitization Framework,” to appear in *Proceedings of the 2010 New Security Paradigms Workshop.*

Summary

- A new cybersecurity project is addressing the unique challenges (and opportunities) of a large high-performance computing environment.
  - Unique network access patterns.
  - Unique usage patterns.
- Initial work has focused on:
  - Better understanding the challenges of machine learning in anomaly detection.
  - “Fingerprinting” of communication and computation.
  - Clustering with self-organized maps.
  - “Rule ensemble” methods for automatic classification.
- Future plans:
  - Improvement of data preprocessing.
  - Improvement of fingerprinting, anomaly detection and rule ensemble schemes.
  - Data sanitization -- to facilitate open-community research and development.
  - Integration of these techniques into usable tools.