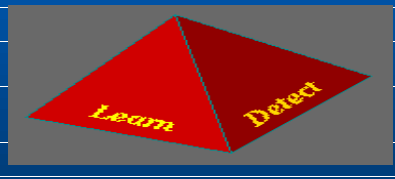
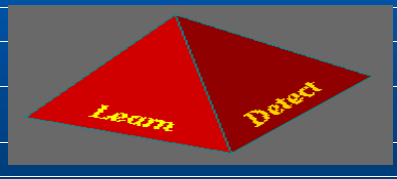
JAM: Java™ Agents for Meta-Learning over Distributed Databases

- Salvatore Stolfo
- Andreas L. Prodromidis
- Shelley Tselepis
- Wenke Lee
- Dave W. Fan
- Philip K. Chan



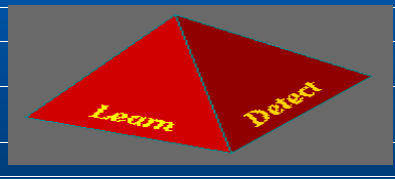
Topics of Discussion

- Scaling Problem for Machine Learning
- Overview of Meta-Learning
- Architecture of JAM
- Fraud and Intrusion Detection
- Future Research



Scaling Problem for Machine Learning

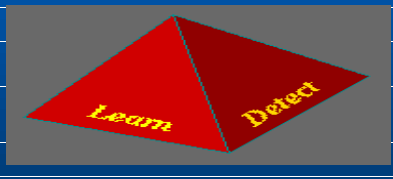
- Objective
  - Acquire knowledge efficiently and accurately from large and distributed databases
- Problems
  - Limited resources
  - Information is distributed



Overview of Meta-Learning

Age	*Thyroxine?*	*TSH*	*... ..*	*Class*
45	False	1.7	... ..	Negative
73	False	10	... ..	Primary
43	False	4.5	... ..	Secondary
...	... ..	... ..	... ..	... ...

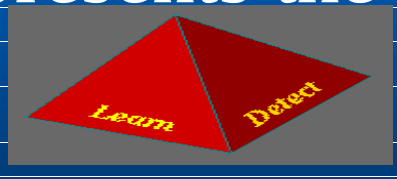
Site-1	Site-2	Site-3	...	Class
Negative	Primary	Negative	...	Primary
Negative	Negative	Negative	...	Negative
Secondary	Negative	Secondary	...	Secondary
...	...	...	...	...



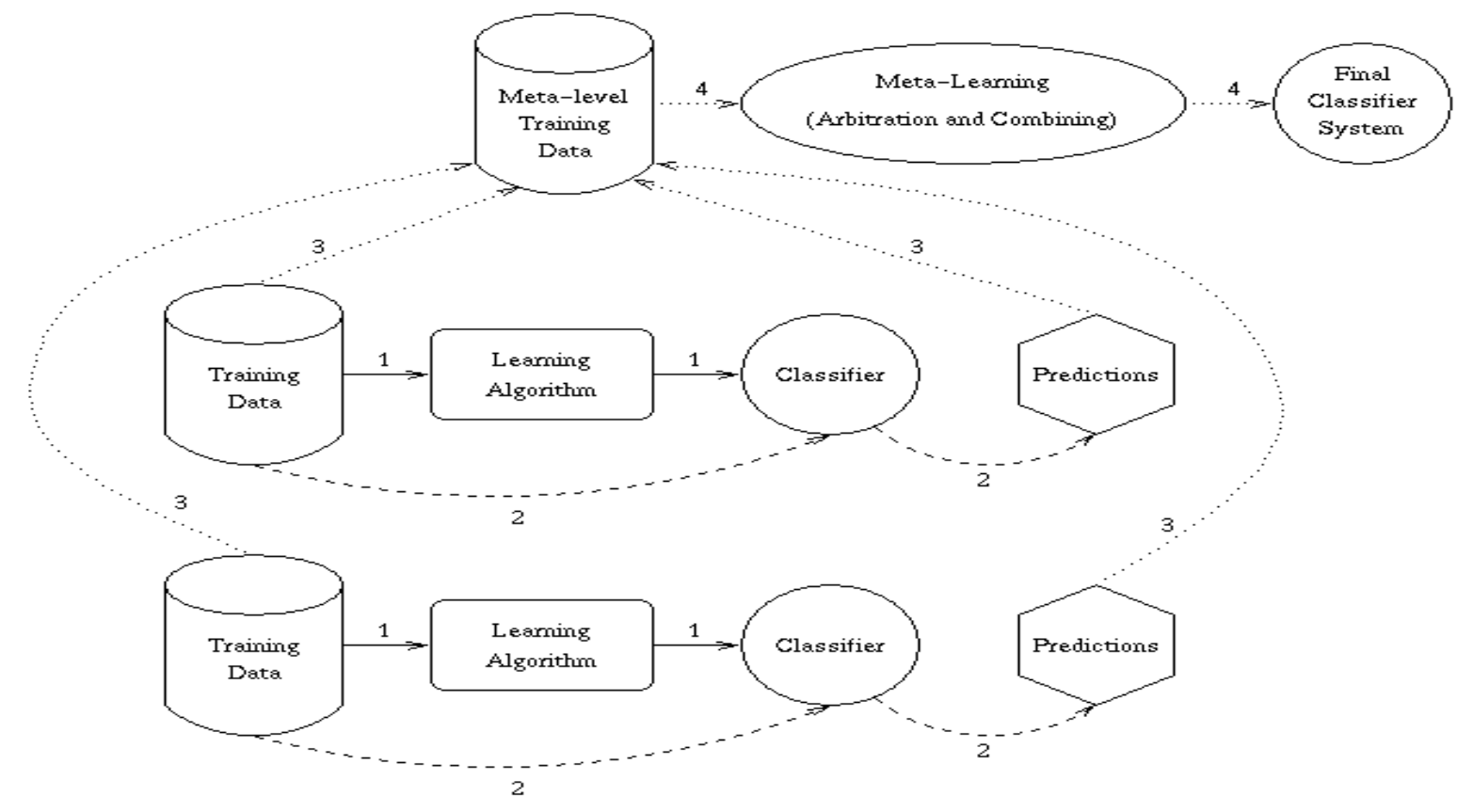
Overview of Meta-Learning

Deploys Machine learning techniques to combine knowledge acquired by separately learned classifiers

- Input: Classified predictions of individual classifiers
- Output: A meta-classifier that represents the collective knowledge

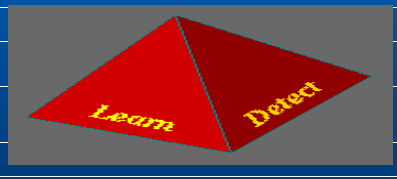


Overview of Meta-Learning



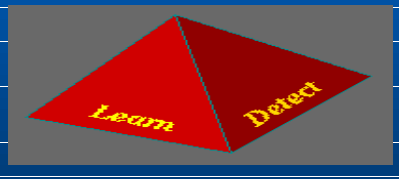
Overview of Meta-Learning: Advantages

- Unifying and extensible approach
  - algorithm and platform independent
- Improves efficiency
  - data reduction
  - parallel execution of serial learning processes
- Improves accuracy

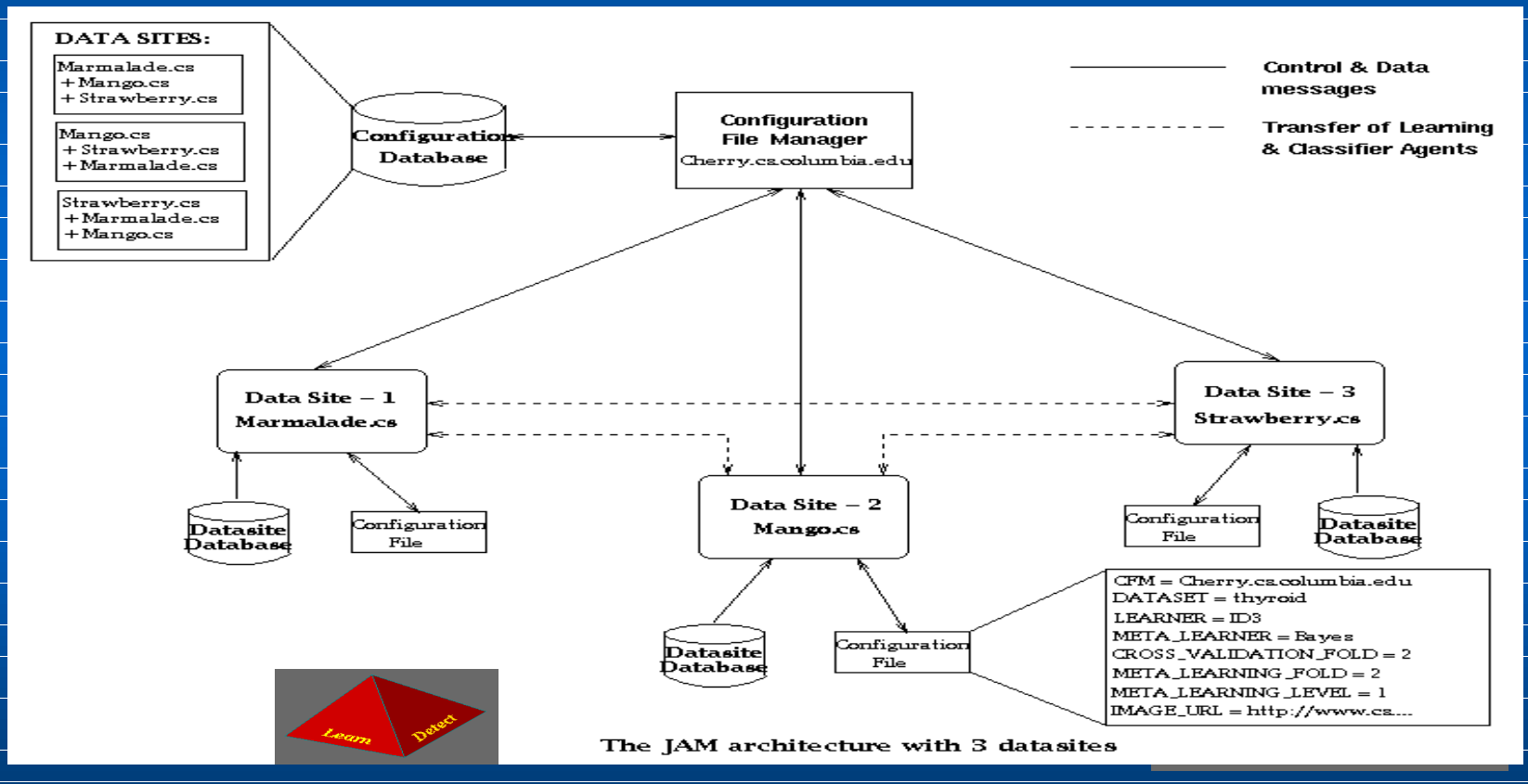


The JAM architecture

- Several distributed database sites
- A collection of Java learning agents
- A collection of Java Classifier agents
- Java based infrastructure for launching and exchanging agents
- A set of meta-learning techniques

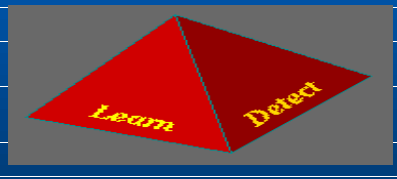


The JAM architecture



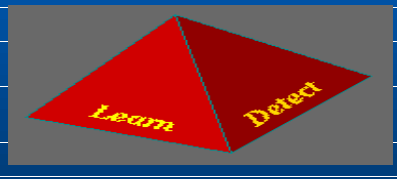
The JAM architecture: Configuration File Manager

- System configuration Manager
- Equivalent to a name server in network systems
- Provides registration services
  - JOIN, QUERY and DEPARTURE requests



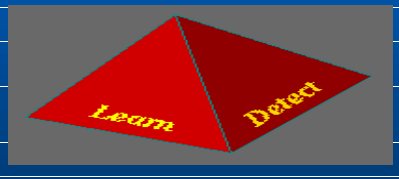
The JAM architecture: Datasites

- Manage local databases
- Build local classifiers
- Obtain remote classifiers
- Build local meta-classifiers
- Interact with User (GUI, or text based UI)

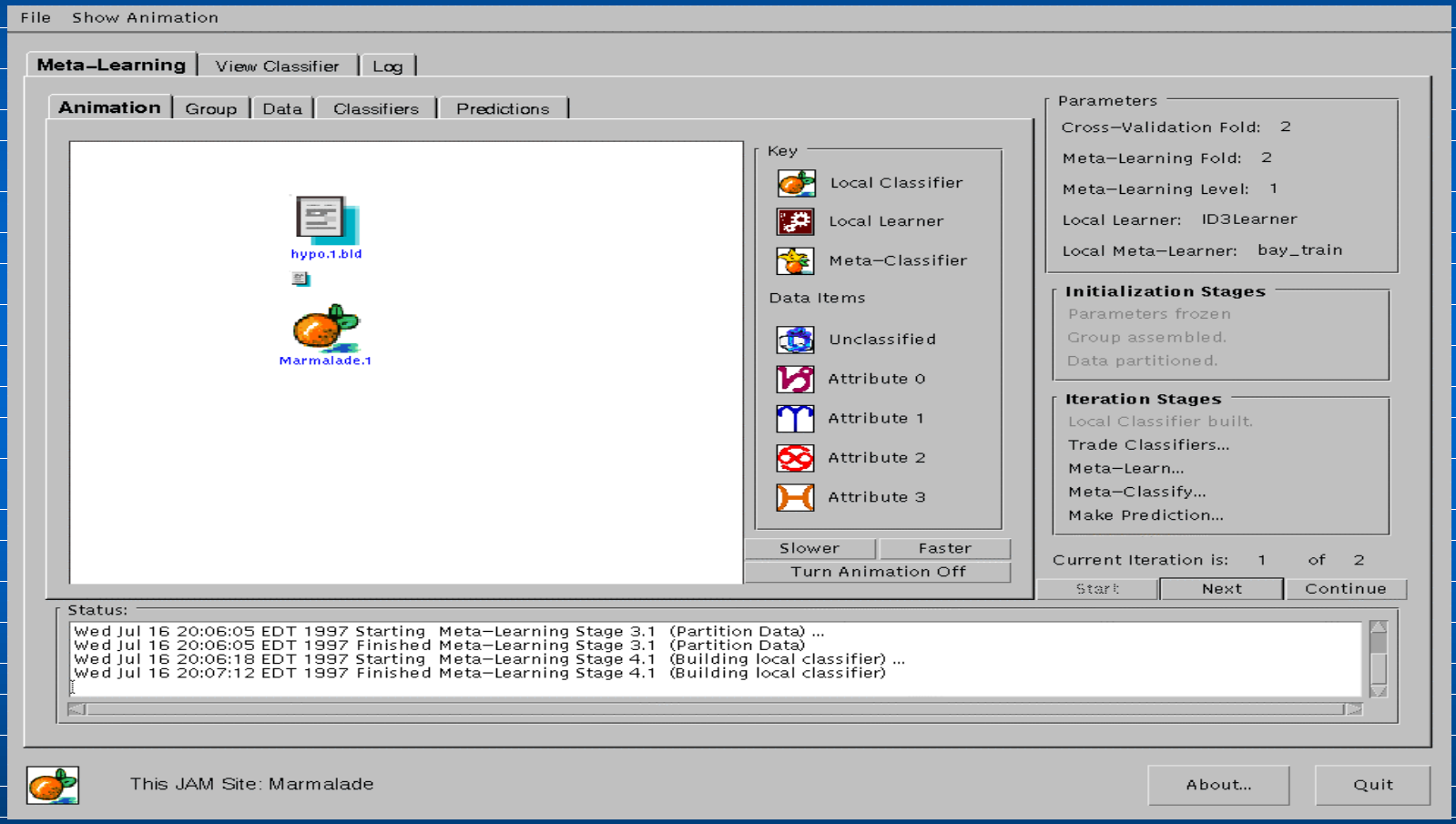


The JAM architecture: Datasites

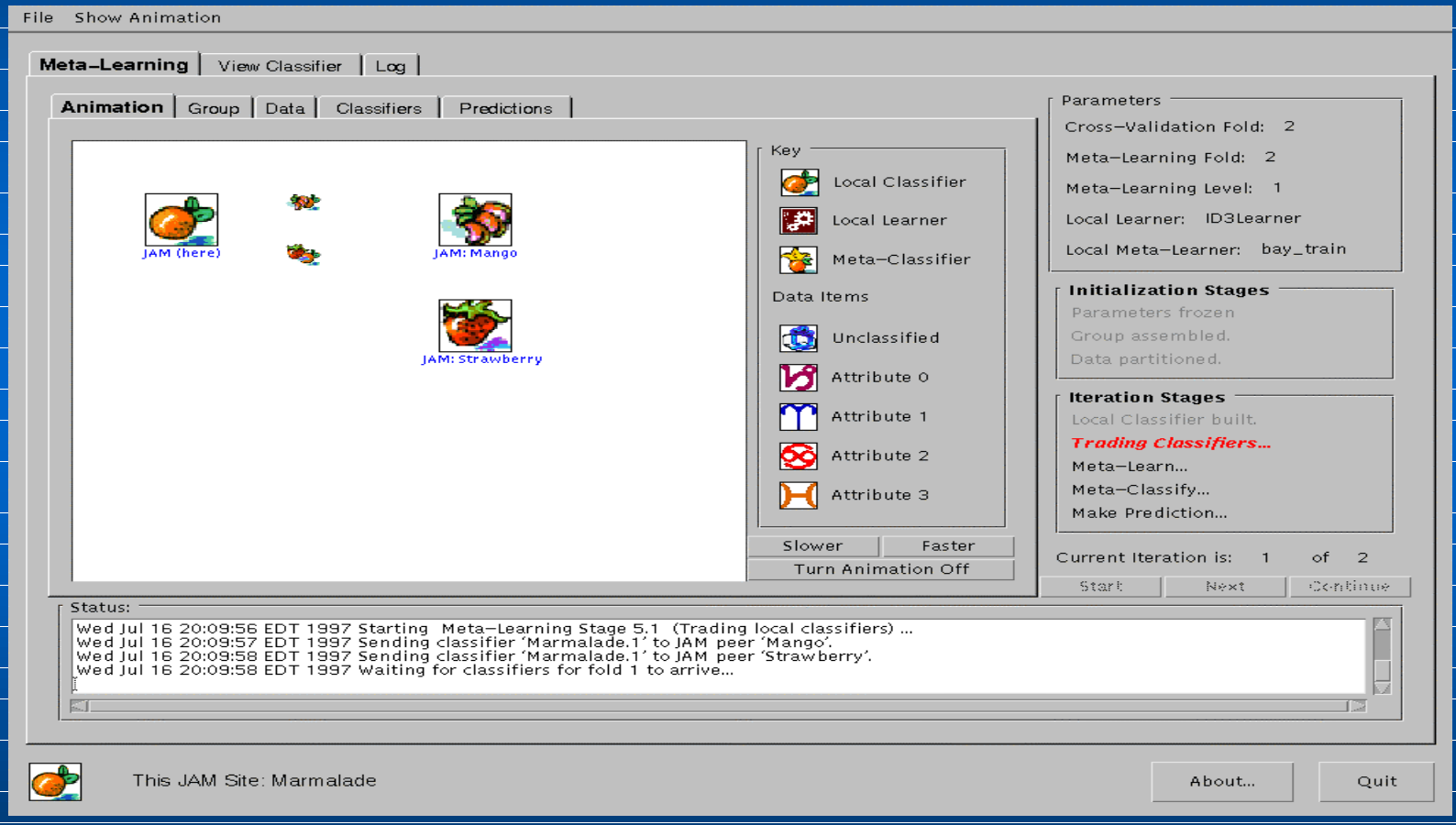
- Active components of JAM
- Multi-threaded Java program
- Command and message driven
  - by user via the GUI
  - from peer Datasites via open sockets.



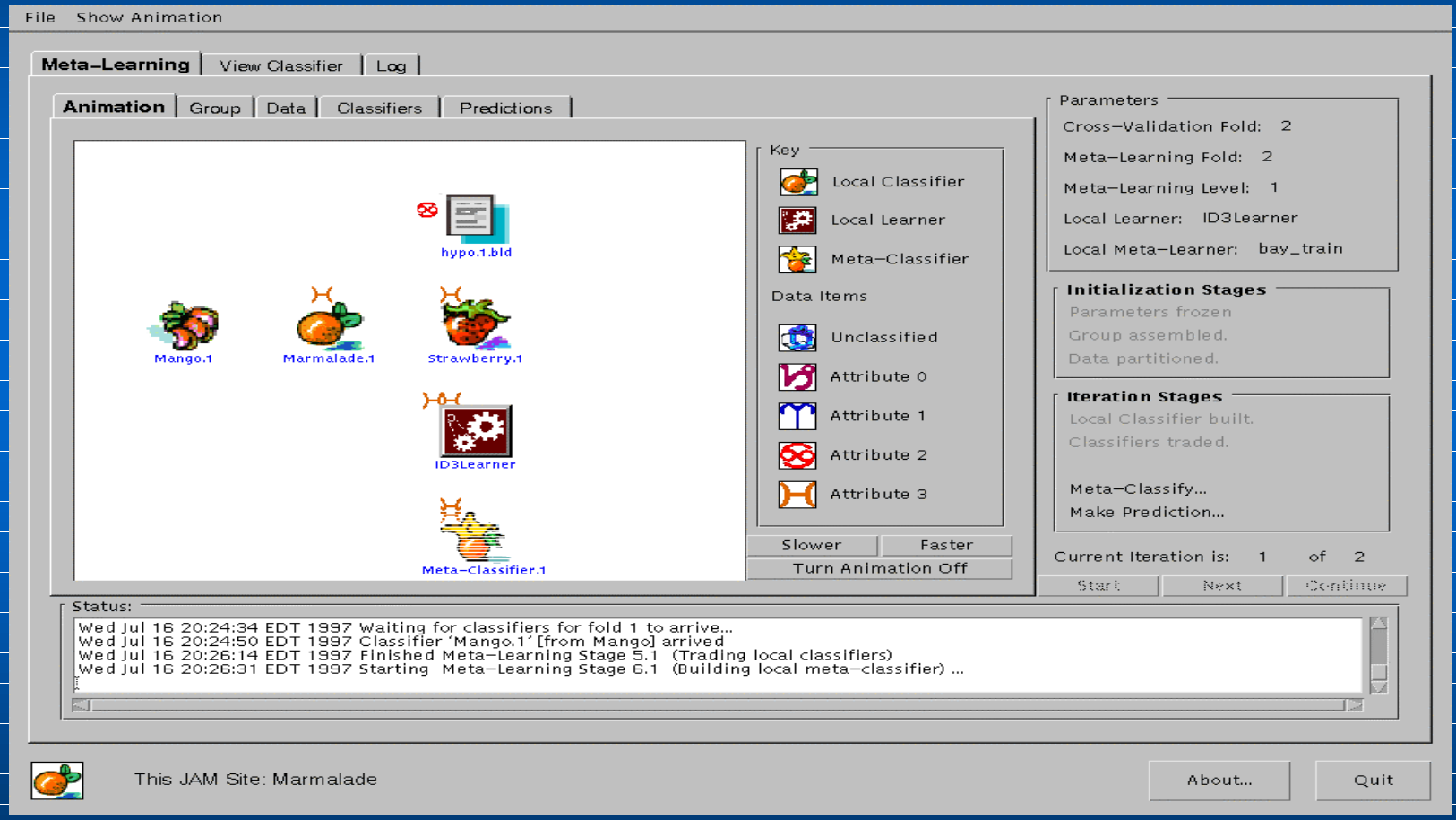
The JAM architecture: Datasites



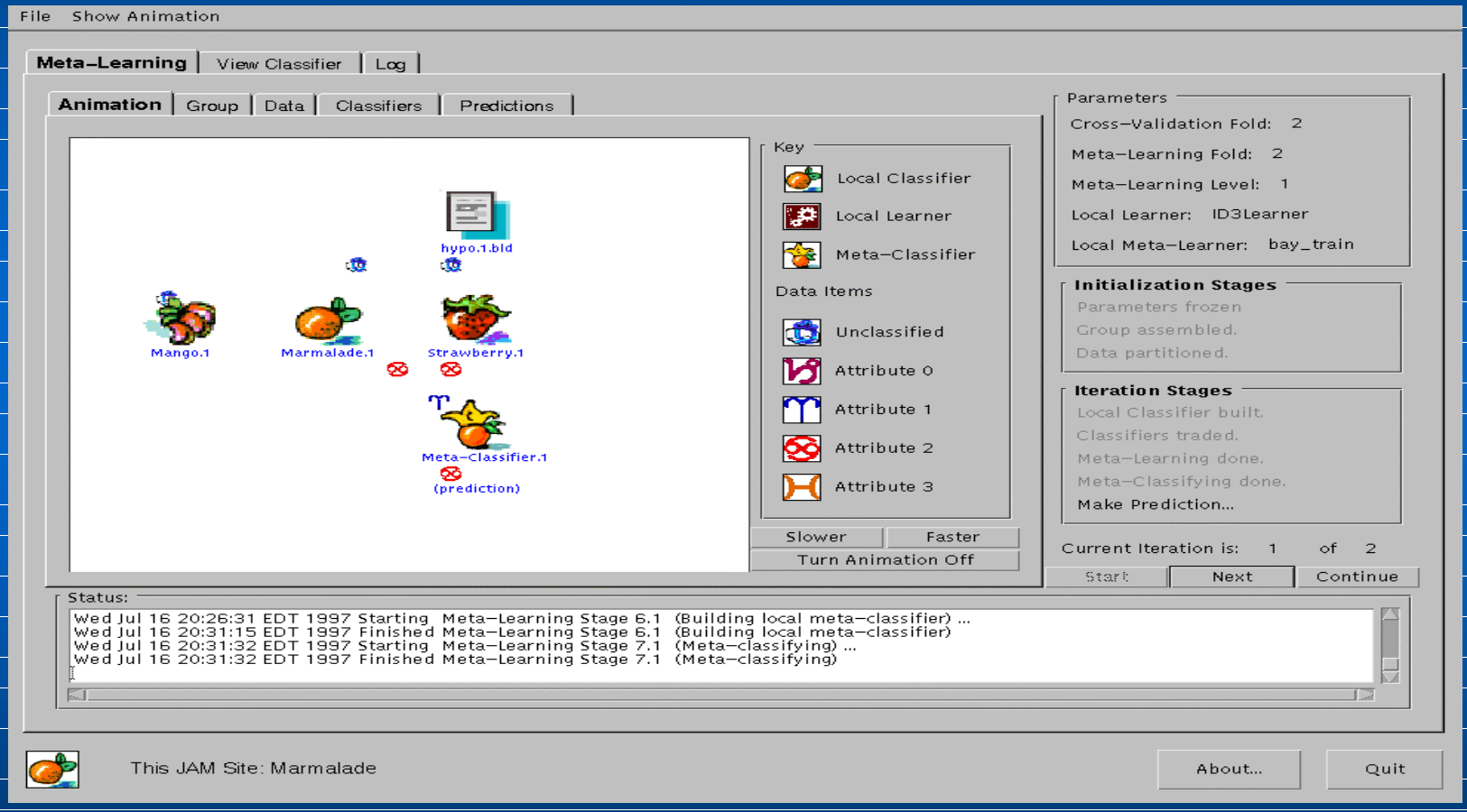
The JAM architecture: Datasites



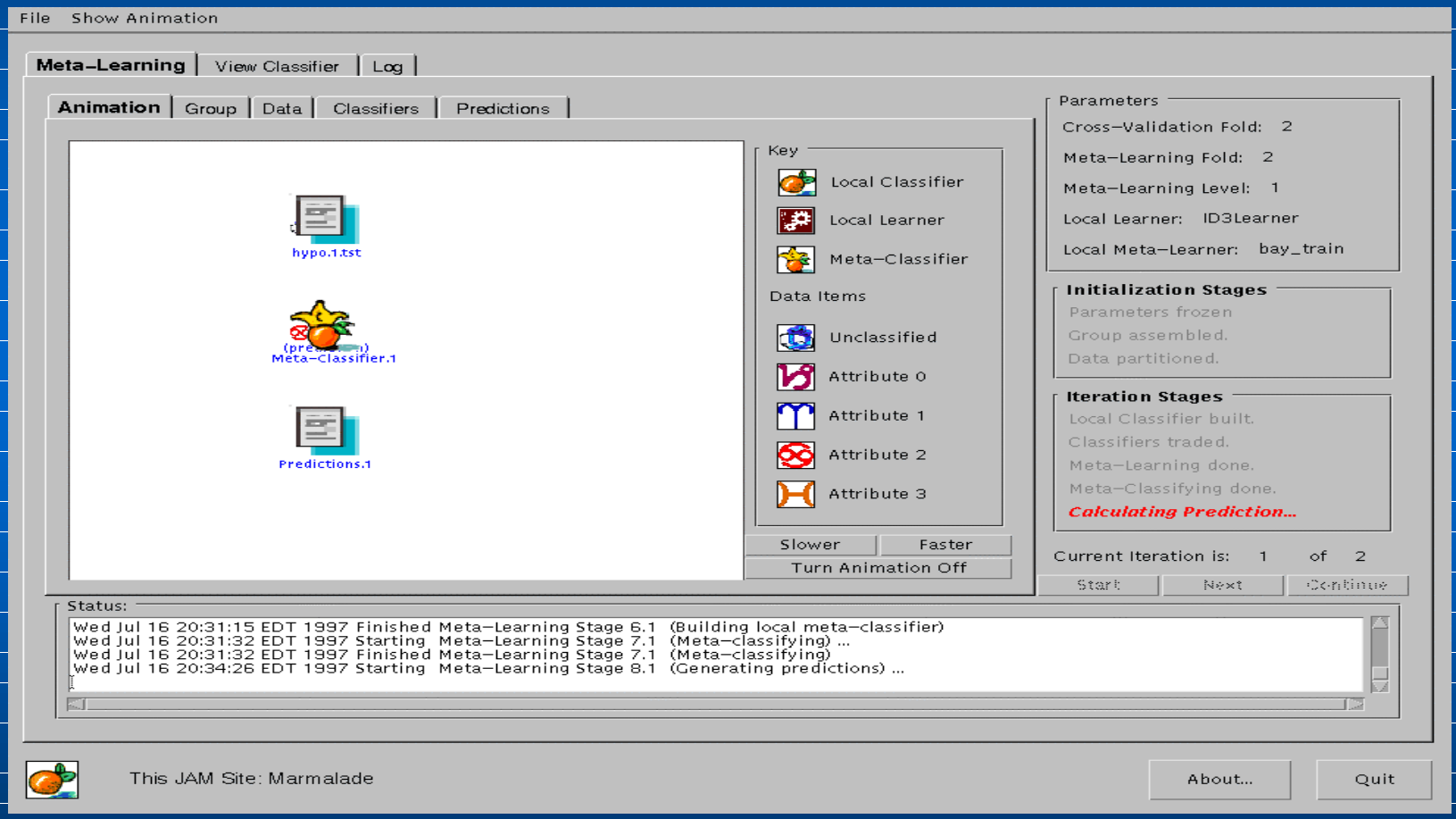
The JAM architecture: Datasites



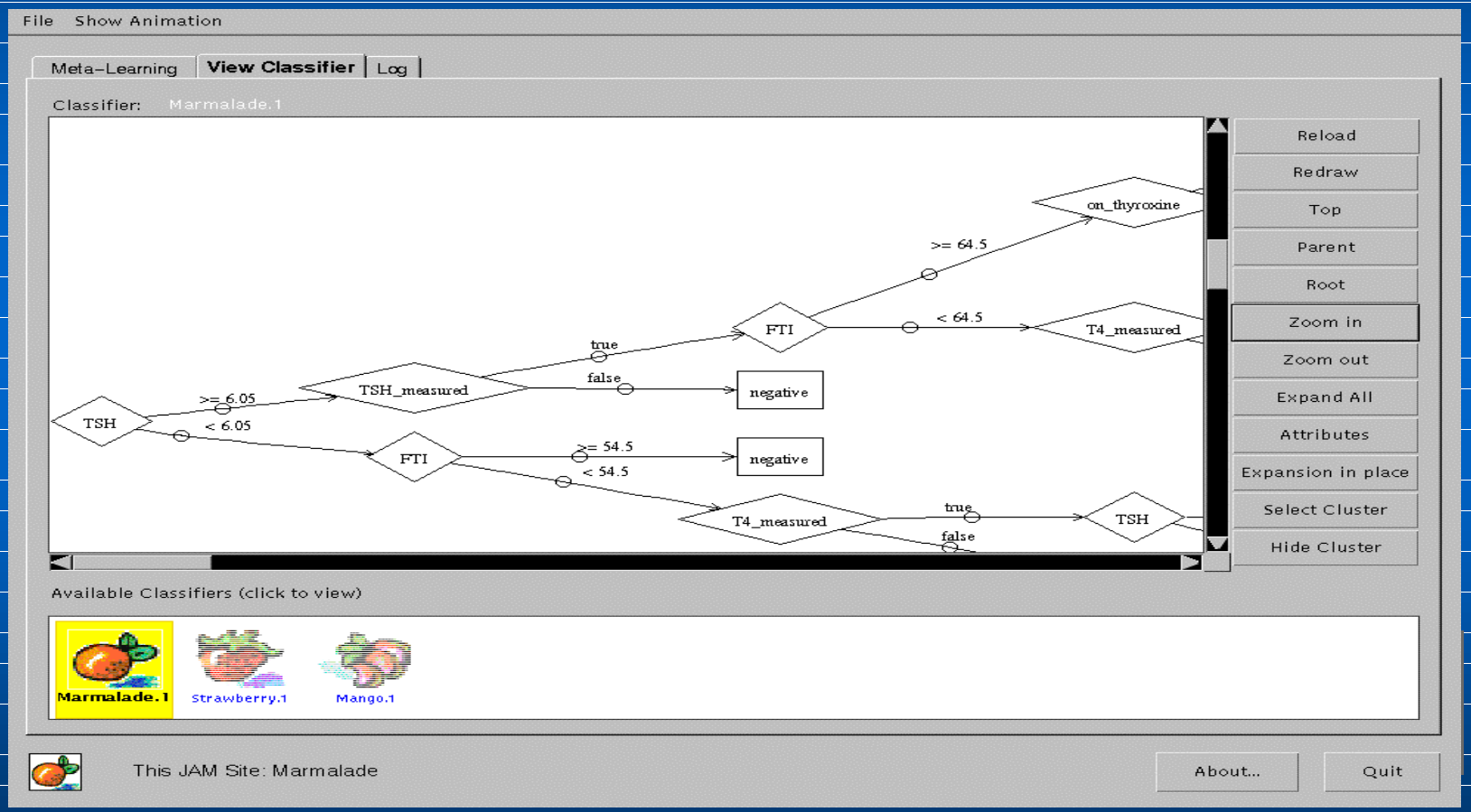
The JAM architecture: Datasites



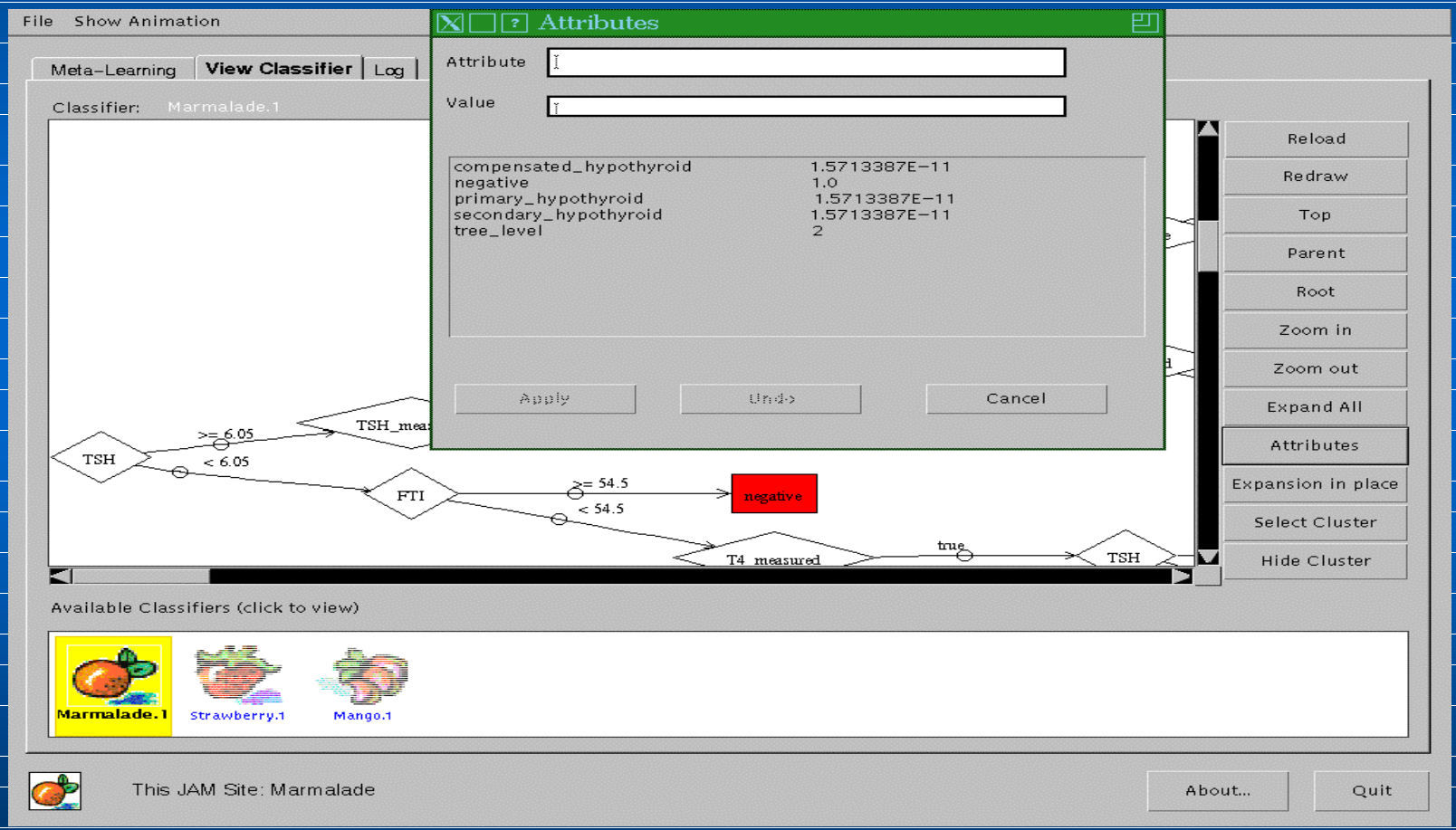
The JAM architecture: Datasites



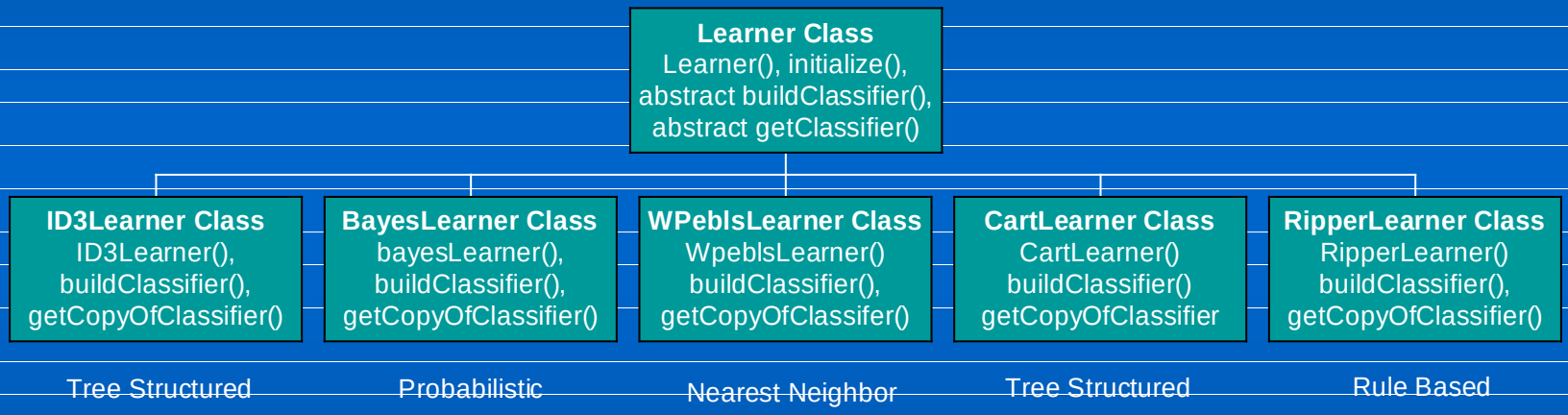
The JAM architecture: Classifier Visualization

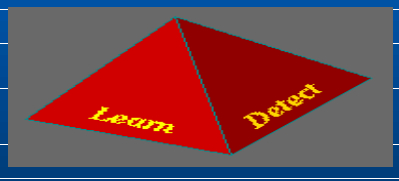


The JAM architecture: Classifier Visualization



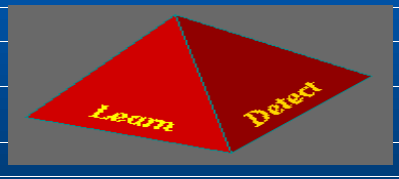
The JAM architecture: Agents





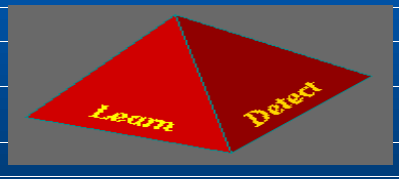
The JAM architecture: Agents

- No assumption on internal representation
- Extensibility
  - Sub-class other defined classes
  - Encapsulate native programs as objects with minimal interface requirements
  - Example of plug and play characteristic :
    - Learner l = new (Learner)
    - Class.forName(name).newInstance();



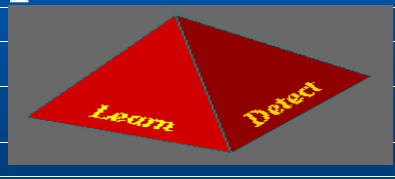
The JAM architecture: Java Implementation

- Portable, platform independent
- Easy to delegate agents to other sites
- Supports distributed programming
- GUI Capabilities
- Connectivity with Databases (JDBC)
- Good relations with WWW
- Security capabilities



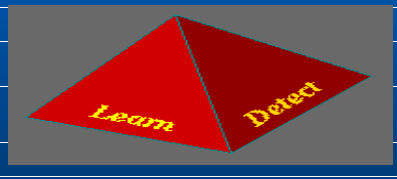
Fraud and Intrusion Detection: Current Status

- Volume of electronic transactions increases
- Number of access points in financial information system grows.
- Financial institutions don’t share their data
- Isolated and custom fraud detection systems
- Databases on transaction behavior are huge and growing

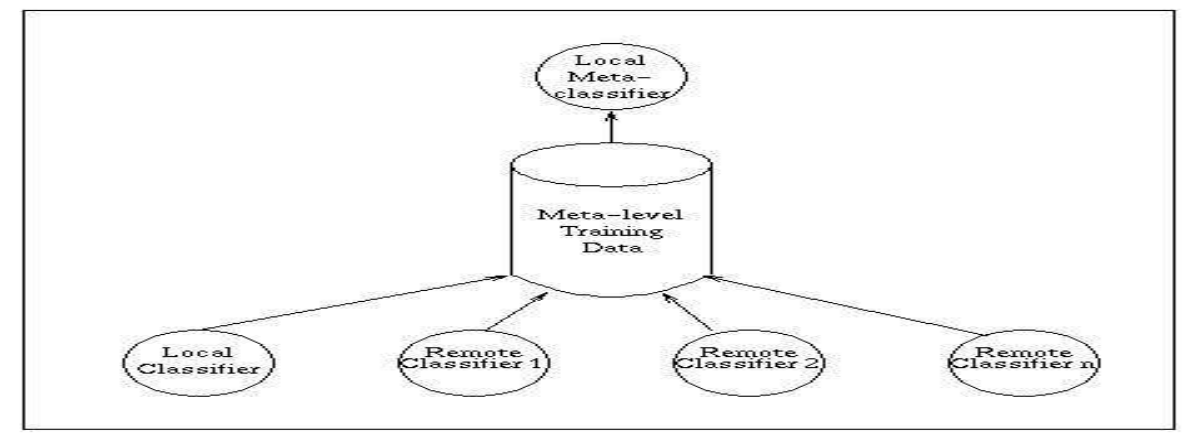


Fraud and Intrusion Detection: JAM solution

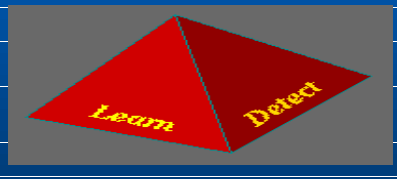
- New wall of protection: Detection
- Employ pattern-directed inference systems
- Share knowledge, NOT data
- Unified and global solution to the problem of fraudulent electronic transactions
- Global or local meta-learned detection systems



Fraud and Intrusion Detection: JAM solution

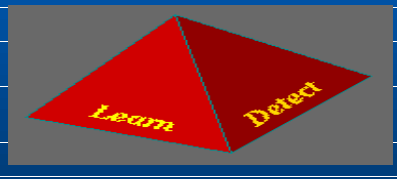


- Compute local fraud detection agents
- Integrate agents (locally or globally)
- Install the resultant meta-fraud-detectors as sentries to warn fraud.



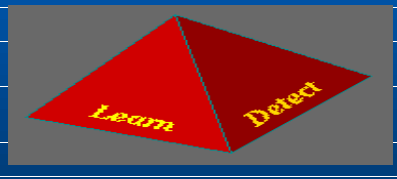
Fraud and Intrusion Detection: Credit card data

- 500,000 record spanning one year
- 30 attributes (numerical, categorical, binary classification) in 137 bytes per record
- Information:
  - Hashed credit card account number, date, time
  - Type of entry of transaction, type of merchant
  - Amount, validity codes, etc
  - The fraud label



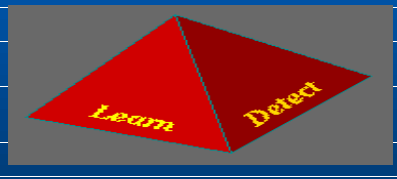
Fraud and Intrusion Detection: Schema Integration Problem (I)

- Bank B imports an additional agent from Bank A that computes the missing values of Bank's B data (if possible)
- Bank B creates additional fields with "null" values. Remote classifiers may not be as predictive, but the meta-learning system presumably detects that



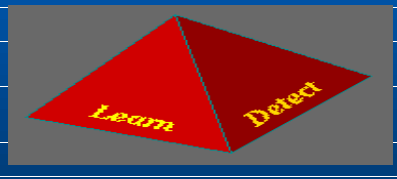
Fraud and Intrusion Detection: Schema Integration Problem(II)

- Bank A learns two models, one for the local and on for the remote schema.
- Bank A learns two models, one for the local and one with attributes that belong to the intersection of the two schemas.



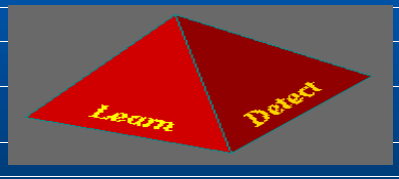
Fraud and Intrusion Detection: Summary of first results

- Number of Datasites: 6
- Training sets: 50-50% Fraud/Non-Fraud
- Testing sets: 20-80% Fraud/Non-Fraud
- Base classifiers: ID3, CART
- Meta classifiers: ID3, CART, Bayes, Ripper
- Base classifiers: 81% TP, 29% FP
- Meta-classifiers: 86% TP, 25% FP



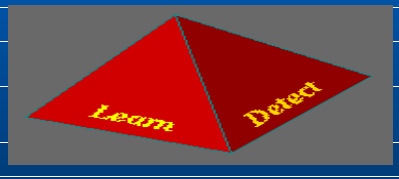
Future Research

- Intelligent and efficient selection of classifiers (TP: 80%, FP: 13%)
- Additional meta-learning techniques (Conflict Resolver : TP: 87%, FP: 9%)
- Mechanism for integrating new knowledge
- Scalable and distributed protocols
- Attribute selection experiments
- Experiments with more data



Summary

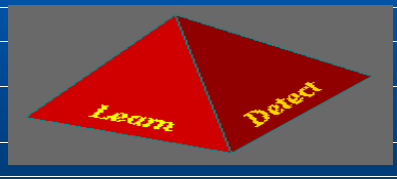
- Meta-Learning provides a scalable solution for learning from massive databases
- JAM is a portable, distributed, extensible agent-based system that uses meta-learning.
- JAM supports the launching of learning and classifier agents to distributed sites.
- JAM is used in fraud and intrusion detection



More information

For more information visit our web site:

http://www.cs.columbia.edu/~sal/JAM/PROJECT

If you are interested send a note to

- sal@cs.columbia.edu or
- andreas@cs.columbia.edu