



Loadstar: Load Shedding in Data Stream Mining

Yun Chi¹, Haixun Wang², Philip S. Yu²
¹Department of Computer Science, UCLA
²IBM Thomas J. Watson Research Center



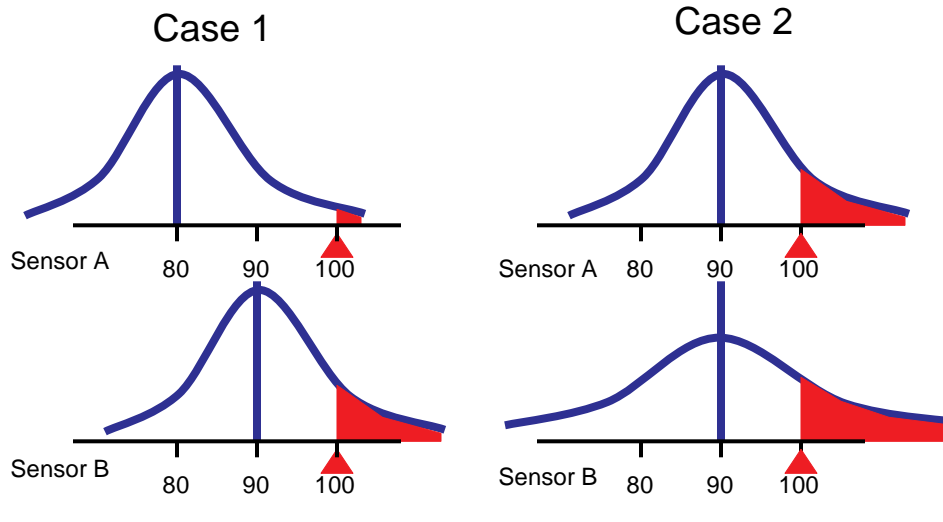
Introduction

- n Data stream systems
 - Data from embedded sensors
 - Financial and retailer data
 - Network traffic data
- n Resources are limited
 - CPU cycles
 - Bandwidth
 - Memory

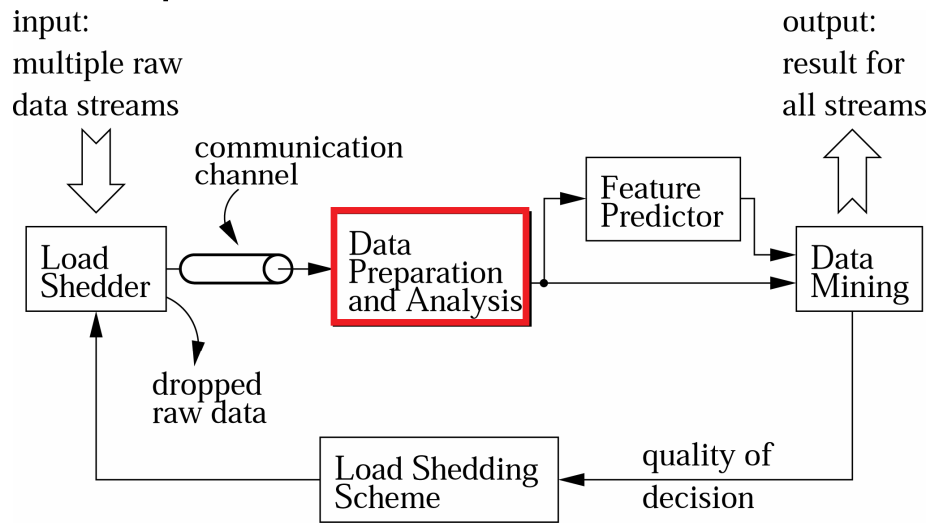
Load Shedding—Which to Drop?

- Load shedding
 - Dropping certain amount of loads
- Which to drop?
 - Randomly
 - Intelligently

Load Shedding—An Example of Temperature Sensors



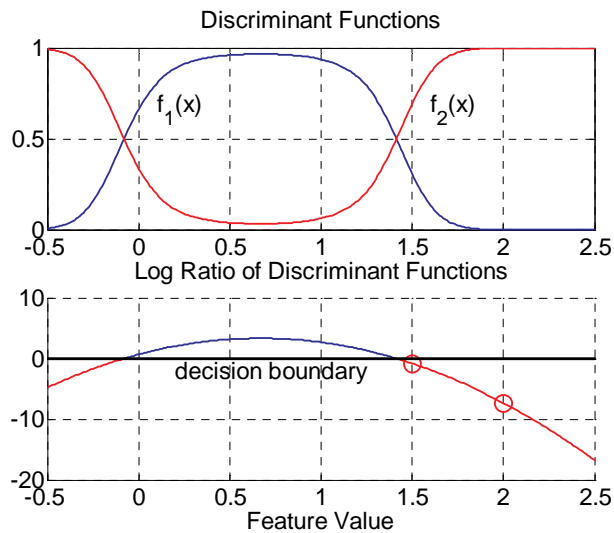
Load Shedding in Classifying Multiple Data Streams—Introduction



Our Main Contributions

- n A Novel *Quality of Decision* (QoD) measure
 - Discriminant functions
 - Predicted feature distribution
- n A feature prediction model based on
 - Markov-chains
 - Real-time parameters update
- n Loadstar

Quality of Decision —Discriminant Functions



Quality of Decision —Based on Overall Risk

- n Feature distribution in the next time unit

$$\vec{X} \sim p(\vec{x})$$

- n At a point \mathbf{x} , the conditional risk for c_i

$$R(c_i | \vec{x}) = \sum_{j=1}^K \sigma(c_i | c_j) P(c_j | \vec{x})$$

- n The expected risk

$$E_{\vec{x}}[R(c_i | \vec{x})] = \int_{\vec{x}} R(c_i | \vec{x}) p(\vec{x}) d\vec{x}$$

- n The decision based on expected risk

$$\delta_2 : k = \arg \min_i E_{\vec{x}}[R(c_i | \vec{x})]$$

Quality of Decision —Based on Overall Risk

n The Bayesian risk:

$$E_{\vec{x}}[R(c^* | \vec{x})] = \int_{\vec{x}} R(c^* | \vec{x}) p(\vec{x}) d\vec{x}$$

n The Quality of Decision (QoD) :

$$\begin{aligned} Q_2 &= 1 - (E_{\vec{x}}[R(c_k | \vec{x})] - E_{\vec{x}}[R(c^* | \vec{x})]) \\ &= 1 - \int_{\vec{x}} [P(c^* | \vec{x}) - P(c_k | \vec{x})] p(\vec{x}) d\vec{x} \end{aligned}$$

Quality of Decision —Based on Overall Risk

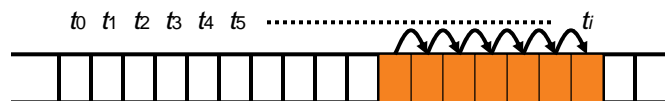
n $0 \leq Q_2 \leq 1$, the higher the Q_2 , the more confident we are.

n $Q_2=1$ if and only if c_k is the minimum-risk decision at all region of the feature space.

Feature Prediction

- n Feature distribution: $\vec{x} \sim p(\vec{x})$
- n Take advantage of temporal locality
 - Stock price data
 - Consecutive snapshots from satellites
- n Discrete-time Markov-chains
- n Real-time parameter updating

Parameter Updating



$$\text{MLE: } \hat{P}_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$$



EM?



$$\text{Approximation: } \hat{P}_{ij} = \frac{n_{ij}}{\sum_k n_{ik}}$$

The Loadstar Algorithm

Algorithm **Loadstar**(N', C)

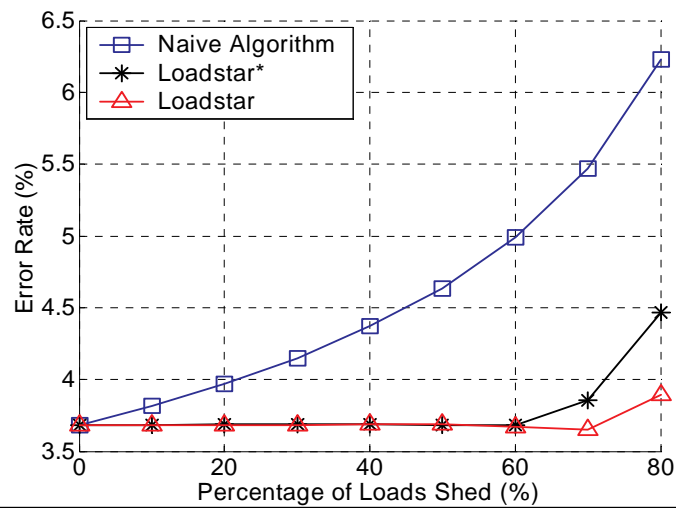
inputs: data from N' streams,
and system capacity C ;
outputs: decisions $(\delta_1, \dots, \delta_N)$;
static variables: feature distributions $p(\mathbf{x})$'s,
Markov-chains MC's,
COR flags (f_1, \dots, f_N) ;

- 1: update $p(\mathbf{x})$ for each feature \mathbf{x} using its MC;
- 2: compute decisions $(\delta_1, \dots, \delta_N)$
and QoDs (q_1, \dots, q_N) using $p(\mathbf{x})$'s;
- 3: select C streams from N' data streams
based on (q_1, \dots, q_N) and (f_1, \dots, f_N) ;
- 4: **for each** selected stream i **do**
- 5: observe the feature value for stream i ;
- 6: revise δ_i for stream i ;
- 7: revise $p_i(\mathbf{x})$ for stream i ;
- 8:* **if** stream i has had load in the
 previous time unit **then**
- 9:* update MC's for stream i ;
- 10:* $f_i \leftarrow false$;
- 11:* **else** $f_i \leftarrow true$;
- 12: **return** $(\delta_1, \dots, \delta_N)$;

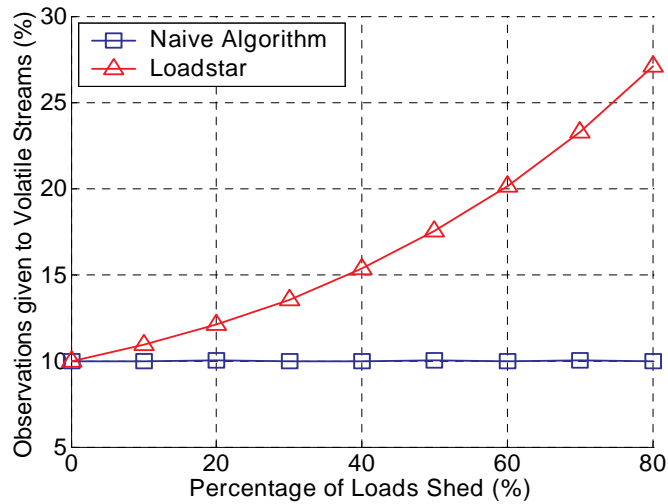
Demonstration

- n Penalty of Load Shedding
- n Resource Adaptability
- n Data Streams with Concept-drifts

Penalty of Load Shedding— Performance



Penalty of Load Shedding—Resources Assigned to the Volatile Streams



Resource Adaptive Load Shedding



Data Streams with Concept-drifts

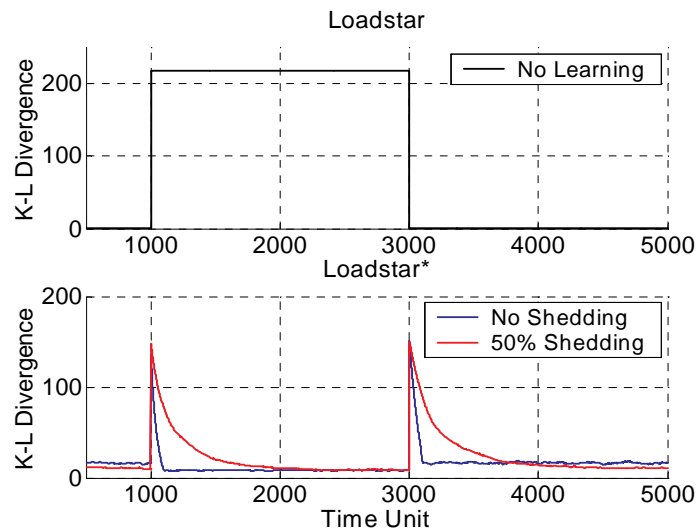
- Use two Markov-chains:

$$P_A = \begin{bmatrix} .91 & .03 & .03 & .03 \\ .03 & .91 & .03 & .03 \\ .03 & .03 & .91 & .03 \\ .03 & .03 & .03 & .91 \end{bmatrix}, \quad P_B = \begin{bmatrix} .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \\ .25 & .25 & .25 & .25 \end{bmatrix}$$

- First use P_A ; at time 1,000 switch to P_B ; at time 3,000 switch back to P_A ;
- Report the total Kullback-Leibler divergence:

$$\sum_i d(P_i, \hat{P}_i) = \sum_i \sum_j P_{ij} \log \left(\frac{P_{ij}}{\hat{P}_{ij}} \right)$$

Data Streams with Concept-drifts



Future Directions

- n Certain time units reserved for learning
- n Networks of dependent data streams
- n Data mining as an intermediate block
- n Control the communication rates