# Loadstar: Load Shedding in Data Stream Mining

Yun Chi[†] Haixun Wang[‡] Philip S. Yu[‡]

[†]Computer Science Dept, UCLA
ychi@cs.ucla.edu

[‡]IBM T. J. Watson Research Center
{haixun,psyu}@us.ibm.com

## Abstract

In this demo, we show that intelligent load shedding is essential in achieving optimum results in mining data streams under various resource constraints. The Loadstar system introduces load shedding techniques to classifying multiple data streams of large volume and high speed. Loadstar uses a novel metric known as the quality of decision (QoD) to measure the level of uncertainty in classification. Resources are then allocated to sources where uncertainty is high. To make optimum classification decisions and accurate QoD measurement, Loadstar relies on feature prediction to model the data dropped by the load shedding mechanism. Furthermore, Loadstar is able to adapt to the changing data characteristics in data streams. The system thus offers a nice solution to data mining with resource constraints.

## 1 Motivation

Consider the following scenario. Two cameras $A$ and $B$ set up on highways transmit streams of snapshots to a central server. One snapshot is taken by each camera in each time unit. But the central server is only able to investigate one snapshot in each time unit. How do we design a load shedding scheme to catch as many speeding cars as possible in real time?

### Naive Approaches

We assume: i) during a certain time period, snapshots from camera $A$ contain speeding cars with probability $p_A$ and snapshots from camera $B$ with probability $p_B$; and ii) the classifier deployed on the central server to capture speeding cars from snapshots is 100% accurate (without false negatives or false positives). We now consider the following two load shedding schemes.

Scheme 1

The probabilities $p_A$ and $p_B$ are unknown to the load shedding mechanism. At each time unit, we randomly select one stream to investigate (i.e., each stream has a probability of $\frac{1}{2}$ to be selected). The expected number of speeding cars caught in one time unit is

$$E_1 = \frac{p_A + p_B}{2}$$

Note that this scheme gives the same result as a deterministic round-robin scheme.

Scheme 2

At each time unit, we select streams based on previous investigation results. We always choose the stream whose last snapshot was investigated and classified as positive (i.e., it contains a speeding car). If none or both streams qualify, we choose one of them randomly. We derive $E_2$, the expected number of speeding cars caught in each time unit, using a Markov model:

$$E_2 = \frac{p_A + p_B - 2p_Ap_B}{2 - p_A - p_B}$$

Unless $p_A = p_B$, we find that Scheme 2 is always better than Scheme 1 as $E_2 - E_1 = \frac{(p_A - p_B)^2}{2(2-p_A-p_B)} \geq 0$.

### Analysis

Scheme 2 is not necessarily the best load shedding scheme, but it provides some intuition behind the goal of load shedding. When we have limitted resources, we shall allocate them to tasks that are most likely to provide the best results. Scheme 2 contemplates that past results are indications of future performance. Based on this belief, it gives priority to the task that was successful last time. Furthermore, it does not assume prior knowledge about the distribution ($p_A$ and $p_B$ are unknown), and therefore if data characteristics change with time, the scheme will adapt to the new environment.



**Proceedings of the 31st VLDB Conference, Trondheim, Norway, 2005**

## 2 Challenges

Our motivating example made many simplified assumptions. For example, in real applications, including network monitoring, credit-card fraud detection, and biosurveillance, we typically monitor hundreds and thousands of data streams simultaneously. Also, the data characteristics of a stream is often changing with time, and it is not enough to model them using simple parameters such as probabilites ($p_A$ or $p_B$).

### State of the Art

Load shedding in mining data streams is a new topic and it raises many challenges. Many approaches assume that a set of Quality-of-Service (QoS) specifications are available [1, 2, 4]. A load shedding scheme decides when and where to discard data and how much data to discard according to the QoS specification. In other words, the assumption is that the impact of load shedding on performance is known *a priori* through the QoS specifications.

This assumption might be valid for simple queries (e.g., aggregation) on data streams. It is often safe to assume that the quality of the query result depends only on the sample size. In contrast, in mining data streams, sample size itself cannot guarantee good mining result, because the quality of mining often depends on specific feature values in a non-monotonic way. For example, within certain regions of the feature space, a classifier may have very high confidence in its classification decision, even if the feature value is only known approximately. But in other regions, a small variation in a feature value may change the decision. In this case, resources (i.e., CPU cycles to compute the exact feature values) should be allocated to a data source if its classification is more sensitive to the exact feature value.

Furthermore, data mining applications are often more sensitive to changes in data characteristics [5]. It means feature value prediction is important to load shedding design for mining data streams. Fortunately, many feature values (e.g, the readings of temperature sensors, the water level of a river, or the feature values extracted from consecutive satellite images) have strong time-correlation and we can build models to take advantage of such correlation. Thus, the challenge lies in building a feature predictor that is able to capture the time-correlation and adapt to the time-variance of the feature values.

### Our Contributions

To the best knowledge of the authors, Loadstar is the first system for load shedding in mining data streams [3]. The system is advantage in several aspects. (1) It employs a novel *quality of decision* (QoD) measure for classification based on the predicted distribution of the feature values in the next time unit. (2) It consists of a feature value prediction model using Markov chain whose parameters can be updated in real time to reflect data distribution changes. (3) Experiments on both synthetic data and real-life data show that Loadstar is effective in improving the accuracy of data stream classification in the presence of system overload.

## 3 System Architecture

We illustrate the major system components of Loadstar in Figure 1. A detailed description of the system can be found in [3].

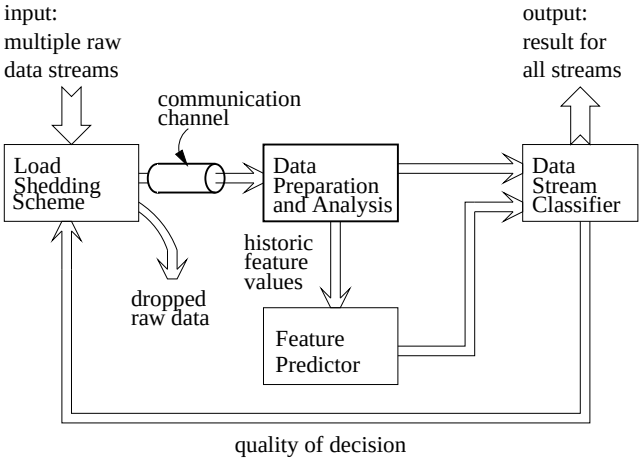



Figure 1: Loadstar System Architecture

Raw data from multiple streams are sent through communication channels to the data preparation and analysis block, which is responsible for data cleaning, feature extraction, feature composition, etc. The derived features are passed to the classifier. Our system assumes that data preparation and analysis (e.g., feature extraction for multimedia data) is CPU intensive. In comparison, the classification requires few CPU cycles. An equivalent scenario is when the bandwidth of the communication channel is limited and therefore not all raw data can go through.

Thus, when the system is overloaded, the data are dropped before they enter the communication channel. The feature values of dropped data are recovered by the feature predictor block based on historic feature values. Therefore, the classifier will handle both the real feature values generated by the data preparation and analysis block, and predicted feature values for data that has been dropped.

## 4 Theoretical Foundation of Loadstar

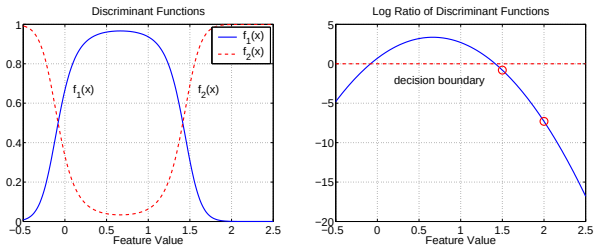
Load shedding takes place when data from multiple streams exceeds the processing capacity. We are interested in load shedding schemes that ensure dropped load has minimal impact on the benefits of mining. In order to do this, we need

- a measure of benefit loss if we discard data $\mathbf{x}$ from a certain stream, and
- the ability to measure the benefit loss without seeing the exact values of $\mathbf{x}$.

### Measuring the Quality of Classification

We view a classifier as a set of *discriminant functions* $f_i(\mathbf{x}), i = 1 \ldots K$. The classifier assigns class label $c_k$ to $\mathbf{x}$ if $f_k(\mathbf{x}) \geq f_i(\mathbf{x}), \forall i$. Consider an example where there are two classes and the data is one dimensional (i.e., there is a single feature $x$). Figure 2(a) shows the two discriminant functions and Figure 2(b) shows their log ratio.

When feature values are not exact, classification decisions have different levels of certainty. For example, assume that $x = 2$ and $x = 1.5$ are *current* feature values and we believe $x$ will not change dramatically in the *next* step. If the classifier has to make a classification decision for the next step, it may assign class label $c_2$ to both data streams; however, for the data stream with $x = 2$, the classifier is much more *certain* about its decision than for the data stream with $x = 1.5$. Intuitively, the *quality* of the classification decision for the first data stream is higher than that of the second data stream.




(a) (b)

Figure 2: Certainty of a Classifying Decision

Assume we have derived a probability density function for $\mathbf{X}$, the feature value in the next time unit:

$$\mathbf{X} \sim p(\mathbf{x}) \tag{1}$$

At point $\mathbf{x}$ in feature space $\mathbf{X}$, if we decide the class is $c_i$, then the conditional risk of our decision is

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

where $\sigma(c_i|c_j)$ is the loss function, i.e., the penalty incurred when the real class is $c_j$ and our decision is $c_i$.

Because we have the distribution of the feature value $\mathbf{x}$ at the next time unit, we can compute the expected risk for a decision for next time unit as

$$E_{\mathbf{X}}\left[R(c_i|\mathbf{x})\right] = \int_{\mathbf{x}} R(c_i|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

We use $\delta$ to represent the best-effort decision rule that minimizes this expected risk:

$$\delta: \; k = \arg\min_i E_{\mathbf{X}}\left[R(c_i|\mathbf{x})\right] \tag{2}$$

Let $c^*$ denote the optimal decision[1] for $\mathbf{x} \in \mathbf{X}$. Because $\mathbf{x}$ is unknown, $c^*$ is infeasible to realize in load shedding. The risk associated with $c^*$ is

$$E_{\mathbf{X}}\left[R(c^*|\mathbf{x})\right] = \int_{\mathbf{x}} R(c^*|\mathbf{x}) p(\mathbf{x}) d\mathbf{x}$$

This risk is the Bayesian lower bound based on distribution $p(\mathbf{x})$. We then define the QoD based on the difference between the expected risk and the lower bound:

$$Q = 1 - \left(E_{\mathbf{X}}\left[R(c_k|\mathbf{x})\right] - E_{\mathbf{X}}\left[R(c^*|\mathbf{x})\right]\right) \tag{3}$$

Intuitively, the larger the $Q$, the higher the quality of the decision. Detailed discussion of how to compute QoD given by Eq 3 based on different loss functions such as 0-1 loss can be found in [3].

### Feature Value Prediction

The computation of the QoD is based on the assumption that we know the distribution of the feature values. In reality, $p(\mathbf{x})$ of Eq 1 is unknown. If the current feature values are independent of those in the next time unit, the best we can do is to use the prior distribution of the feature values. However, in many real life applications, feature values often have short-term temporal correlation.

Loadstar uses discrete-time Markov-chains with a finite number of states for feature value prediction. Consider any feature $x$ and its corresponding Markov-chain. Assume the feature value at time $t_0$ is known to us, and we have $x = s_i, \; 1 \leq i \leq M$. Thus, the distribution of the feature value at $t_0$ is $p_0(x) = e_i$, where $e_i$ is a $1 \times M$ unit row vector with 1 at position $i$ and 0's at other positions. The distribution of the feature value in the next time unit $t_1$ is $p_1(x) = p_0(x)P = e_i P$, where $P$ is the state transition probability matrix. In the next time unit $t_2$, the distribution of the feature value becomes $p_2(x) = p_1(x)P = e_i P^2$.

We refer readers to [3] for a detailed description of the theoretical foundation of load shedding in classifying data streams.

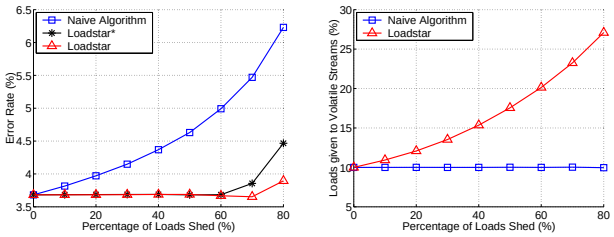
## 5 About the Demo

Our VLDB 2005 demo will illustrate the key features of the LoadStar system. In particular, we will demonstrate:

[1] More rigorously, $c^*$ should be written as $c^*(\mathbf{x})$.

### Penalty of Load Shedding

A good load shedding scheme allows system performance (e.g., accuracy in classification) degrade gracefully under system overload. Our demonstration will show how Loadstar achieves this goal. Here, we give an example, where we apply load shedding to change detection. Assume we have a maximum load of 100 data streams, among which 10 are volatile from time to time. During system overload, the system capacity is reduced from 100 data streams to 20 data streams.




(a) (b)

Figure 3: Performance Comparison

Figure 5(a) shows the error rates of the classifier under different levels of overload[2]. Note that the maximum error rate is 10%, as only 10 streams are volatile. Using Loadstar, performance will not degrade until load shedding is beyond the 60% level. Figure 5(b) explains the reason behind the good performance: as load shedding builds up, increasing percentage of resources are devoted to streams that are volatile, while in the naive approaches, the percentage is always fixed at 10%.

### Resource Adaptive Load Shedding

In our demo, we will show how Loadstar dynamically adjusts to changes in available system resources. Figure 4 is a snapshot of the monitor screen of our load shedding system. Loadstar monitors the fluctuation of classification error (the screen on the left hand side) in response to changes in system capacity (the screen on the right hand side). Both of the screens are in fact moving windows on the time axis. They demonstrate how the load shedding system manages to keep the fluctuation of the error rate under a relatively low level when system capacity is going through very dramatic changes. This proves that an intelligent load shedding scheme can make a stream management system more robust to external disturbances.

### Data Streams with Concept-drifts

We demonstrate how Loadstar copes with concept drifts in the data stream. Loadstar dynamically learns a Markov model to predict feature values of unseen data. Our demonstration uses the Kullback-Leibler

[2]Loadstar* is a Loadstar variant that handles data streams with evolving concepts.

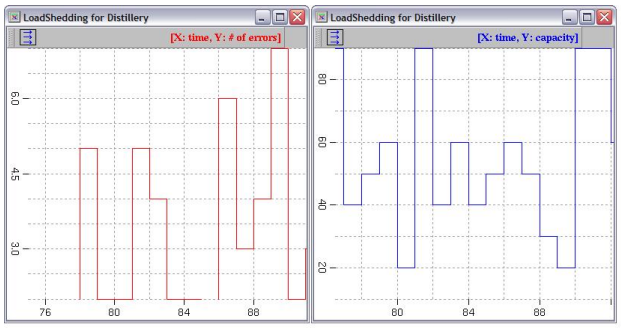
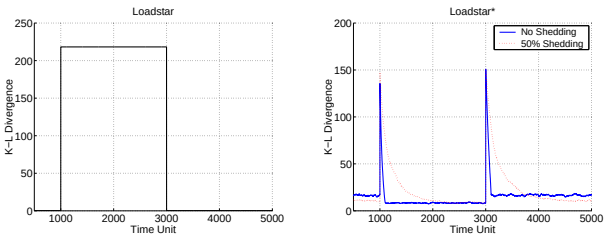



Figure 4: Real Time System Montioring

divergence as the measure of error. Figure 5 shows the effect of learning. When data distributions change at time unit 1000, the system will experience a sudden increase in classification error. However, in a system with learning capability, this increase is merely temporary.




(a) Loadstar without Learning (b) Loadstar with Learning

Figure 5: Learning the Markov-Chains

## References


[1] D. J. Abadi, D. Carney, U. Cetintemel, M. Cherniack, C. Convey, S. Lee, M. Stonebraker, N. Tatbul, and S. Zdonik. Aurora: a new model and architecture for data stream management. *The VLDB Journal*, 12(2):120–139, 2003.

[2] B. Babcock, M. Datar, and R. Motwani. Load shedding for aggregation queries over data streams. In *20th International Conference on Data Engineering*, 2004.

[3] Yun Chi, Philip S. Yu, Haixun Wang, and Richard Muntz. Loadstar: A load shedding scheme for classifying data streams. In *SIAM International Conference on Data Mining (SDM)*, 2005.

[4] N. Tatbul, U. Cetintemel, S. Zdonik, M. Cherniack, and M. Stonebraker. Load shedding in a data stream manager. In *Proc. of the 29th Intl. Conf. on Very Large Databases (VLDB'03)*, 2003.

[5] H. Wang, W. Fan, P. S. Yu, and J. Han. Mining concept-drifting data streams using ensemble classifiers. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2003.