Intelligent instance selection of data streams for smart sensor applications

Magdiel Galan[a], Huan Liu[a], Kari Torkkola[b]

[a] Computer Science and Engineering Department, Arizona State University, Tempe, AZ 85287
[b] Motorola Labs, 2900 South Diablo Way, Tempe, AZ 85282

ABSTRACT

The purpose of our work is to mine streaming data from a variety of hundreds of automotive sensors in order to develop methods to minimize driver distraction from in-vehicle communications and entertainment systems such as audio/video devices, cellphones, PDAs, Fax, eMail, and other messaging devices. Our endeavor is to create a safer driving environment, by providing assistance in the form of warning, delaying, or re-routing, incoming signals if the assistance system detects that the driver is performing, or is about to perform, a critical maneuver, such as passing, changing lanes, making a turn, or during a sudden evasive maneuver. To accomplish this, our assistance system relies on maneuver detection by continuously evaluating various embedded vehicle sensors, such as speed, steering, acceleration, lane distance, and many others, combined into representing an instance of the "state" of the vehicle. One key issue is how to effectively and efficiently monitor many sensors with constant data streams. Data streams have their unique characteristics and may produce data that is not relevant or pertinent to a maneuver. We propose an adaptive sampling method that takes advantage of these unique characteristics and develop algorithms that attempt to select relevant and important instances to determine which sensors to monitor and how to provide quick and effective responses to this type of mission critical situations. This work can be extended to many similar sensor applications with data streams.

Keywords: Data Mining, Data Streams, Instance Selection, Adaptive Sampling

1. INTRODUCTION

The purpose of our work is to develop techniques that will allow us to mine streaming data from hundreds of automotive sensors of various types that continuously monitor a vehicle and its driving environment. Our goal is to capitalize on this information to help us identify when a driver is executing a critical maneuver, in order to develop methods that will aid in minimizing driver distraction from in-vehicle communications systems such as DVD and game players, iPod, cellphones, PDAs, Fax, eMail and other messaging, entertainment, or communication devices. A critical maneuver may include actions such as passing another vehicle, performing a left or right turn, or a sudden evasive maneuver, to name a few. The possibilities of an accident are significantly increased if a driver were to be distracted by any of these devices when performing one of the critical maneuvers.

According to statistics by the National Highway Traffic Safety Administration (NHTSA) in 1996, driver distractions contributed to 20-30 percent of all crashes. A more recent study by SAVE-IT (Safety Vehicle Using Adaptive Interface Technology) project funded by the NHTSA showed in their 2003 report that this statistic has increased to 20-50 percent. A contributing factor may be the incorporation of Information and Mobile Technology advances into our vehicles, mainly a result of our demand for constant information. We as individuals maintain busy schedules and often want to maximize the usage of our time, including the time we spend on communication and entertainment in our vehicles. It is not unusual to see drivers having calls or even participating in meetings while driving on the road. Consequently, our vehicles will soon be transformed into a truly mobile information center in order to satisfy our craving for information on anything that might affect our work, business, family, health, or personal finance. Unfortunately, this information-intensive environment significantly increases the cognitive load of a driver, with both the road and communication or entertainment devices competing for the driver's attention, stretching thin its ability to handle multiple tasks. If not managed properly, a distracted driver could get involved in an accident.

Our endeavor is to maximize the usage of our time while driving, minimizing distractions, creating a safer driving environment, by providing assistance in the form of warning, delaying, or re-routing, incoming signals if the assistant system recognizes that the driver is performing, or is about to perform, a critical maneuver. An example to illustrate this scenario could be that of the case of an incoming cellphone call that gets transferred into a voicemail, if the driver is in the midst of passing another vehicle, or to turn for an exit. The critical aspect to accomplish this feat is the ability to detect or recognize when a critical maneuver is being executed, or preferably, before it is about to start. Our approach is to explore the myriad of sensors that could be embedded in an automobile such as speed, steering, acceleration, lane distance, and many others, are combined into representing a sequence of the "states" of a vehicle in order to use data mining techniques to recognize critical maneuvers.

As we know, most of these embedded systems operate under tight memory and time constraints, which are characteristics of streaming data. It is impractical to hold an entire stream in a computer memory as such applications cannot wait for a lengthy stream to be captured before performing analysis. In addition, there are hundreds of sensors available to track and monitor. It is not efficient, nor practical, for a central embedded system to process such a vast number of multiple streams[13]. But if managed properly, we may not have to do so. Many sensors may produce data that are not always pertinent to a maneuver. Sensors continue spewing data even when no maneuvers are in effect, as in the cases of driving straight ahead on a freeway, or while waiting during a traffic jam or for a traffic light change. While in other cases like braking, it may not be relevant when passing a car on the freeway, but relevant when slowing down to execute a sharp turn. Hence, it will be most effective if we can capture only relevant data specific to a maneuver, and skip segments of a stream, or even an entire stream.

The implications of knowing which segments of the stream are relevant can have benefits in data reduction, storage and algorithm execution efficiency. Since only the most relevant data points are kept from the original data set, processing time is spent only on generating results that would matter the most. As such, the objective of our work is to explore the feasibility to develop a methodology that delivers a synthesized version of these streams, where only critical and relevant data is delivered to the central unit. The exercise of just extracting relevant information is a form of data reduction and sampling known as instance selection. There have been many approaches to generate a reduced data set from streaming data. Some include methods to generate a synopsis of the data, others implement some forms of feature selection, and a third group attempts to extract the relevant data points. Our work tries to highlight some of the stream characteristics necessary for instance selection, and take advantage of them to develop intelligent machine learning algorithms that attempt to select relevant and important instances through an adaptive sampling methodology implemented to determine which sensors to monitor and how to provide quick and effective responses to critical situations.

The rest of the paper is organized as follows. In Section 2 we provide an overview of instance selection. In Section 3 we describe the environment that generates the stream sensor data and describe in detail a sample stream for one of the sensors and its correlation to a changing lane maneuver. In Section 4 we describe some of the issues and challenges associated with streaming data in regards to our application. Section 5 provides a summary of related work associated with instance selection for streaming data. In Section 6 we describe our proposal for an adaptive sampling window And its implementation. Section 7 describes an empirical case study using the proposed adaptive sampling window method. Section 8 provides conclusions and plans for future work.

2. INSTANCE SELECTION

Instance selection is one of the effective means to dealing with massive data[2, 13], which denotes finding relevant data points among massive data, resulting in a data reduction, or scaling down, while still maintaining the essence of the data. From a data mining perspective, an ideal outcome for instance selection would be that the selected instances of the original data points should yield the same results or conclusions as if the entire data set were used. Instance selection should not be confused with another popular reduction technique known as feature selection[12], which is to select relevant features from a data set, reducing the number of dimensions. Feature selection has been the focus of many data mining developments as it facilitates the generation of decision trees, association rules and other classifier techniques. Instance selection instead tries to achieve data reduction by reducing the number of instances (or tuples) in the data. Both techniques, instance selection and feature selection, seek to yield benefits on both data storage reduction and algorithm execution efficiency, yet instance selection is most often overlooked, as many more works are generated

addressing feature selection, than there are for instance selection. Feature selection is very appropriate for some applications like BioInformatics, since the number of features represented as ***M*** greatly exceeds the number of instances or tuples ***N***, that is, ***M >> N***. This application can gain very little from instance selection since it is typical to have less than a few hundreds of samples containing information for thousands of genes. It is quite the contrary for other applications such as those involving streaming data, where there is a lot to be gained due to the endless number of instances or data points generated which greatly exceeds the number of features for the application, that is, ***N >> M***.

Some of the metrics that can be used to measure instance selection performance include level of compression and reproducibility. We define the data stream S and R as continuous sequences of elements such that $S = (s_1, s_2, \ldots, s_n, \ldots)$ and $R = (r_1, r_2, \ldots, r_k, \ldots)$ where subscript 1, 2, k, n, represent the order in which the elements are generated and observed where S will represent the original full data stream, and R represents the resulting reproduced stream after instance selection or reduction. We define the level of *compression* C as the ratio of the number of instances that remain, from an original data set after reduction, represented by R, to the number of instances originally present in S, expressed as a percentage . *Reproducibility* refers to how closely the reduced data resembles the original data or that the original data can be reproduced from the reduced data. It is usually measured in terms of how much the reduced data R deviates from the original data S, and typically expressed as some form of absolute or cumulative *error* (using the Euclidean distance as an example) shown next.

$$D(S,R) = \sqrt{\sum_{i=1}^{n} (s_i - r_i)^2}$$

As such, closeness or similarity of a stream S to the reproduced sequence R is represented by the cumulative squared difference between the two sequences at each corresponding moment in time, for the length n of the sequence. Both compression and reproducibility metrics are interdependent, that is, there is usually a trade-off between the two metrics. The greater the reproducibility, the lower the level of compression, and viceversa. A significant goal of instance selection would be to achieve maximum compression with the minimal amount of error in reproduction.

In a sense, instance selection is a form of sampling. Although it could be easily thought of as if performing sampling at regular intervals or just randomly, we strive to do better than that. Streaming data imposes several challenges for instance selection due to some of its inherent properties such as unbounded-ness, typical fast rate generation, and temporal component not experienced by traditional databases. Most significantly, traditional data bases have the advantage that data processing can be performed off-line, and can be executed over several passes of the data if needed to achieve maximum performance, whereas streaming data analysis has to be performed on-line, and usually have one shot at the data, so it has to be performed in one pass of the data. Nevertheless, many of the techniques developed for traditional database analysis have been adapted to streaming data using a sliding windows approach. A window is a snapshot in time for temporal data. Window length or size is expressed in terms units of time. A sliding window can be viewed as a sequence of transitioning snapshots. A sliding window is defined with the following sliding window model. At every time t, a data record arrives and is added to the window, while the oldest element within the window is removed, that is, the oldest element expires at time $t+N$, where N is the window length expressed in units of time.

3. TEST-BED ENVIRONMENT

The streaming data generated for this research was obtained from a driving simulator from the Motorola Human Interface Lab located in the Motorola Labs facilities in Tempe, AZ. The simulator, by GlobalSim, includes a full size, four-door sedan, with three front video projections providing a 150 degree view of traffic and road ahead, with realistic feedback for the steering wheel, pedals and other normal driving controls. The simulator records simulation parameters, driving conditions (speed, acceleration, braking, etc.), vehicle systems such as engine parameters (coolant temperature, oil pressure, etc.), drive controls (transmission gear selection, steering angle, window and seat belt status, etc.) and every action executed by the driver. Video cameras capture the road ahead, driving behaviors, utilization of in-vehicle devices, driver's gaze, blink rate, as well as hand positions and foot pedal actions. The system was extended to include possible future sensor devices such as radar, GPS, and a camera system for lane positioning and lane marking. In all, the

system is capable of recording up to 400 variables of combined continuous and discrete data; many of them sampled at 60 Hz.

Various forms of simulated driving environments were generated to provide a varied set of freeway and suburban roads, populated with a mix of normal and erratic vehicles. Four drivers were asked to simply navigate through this virtual world and perform unrestricted actions or maneuvers as they normally would on a real environment. Driving sessions lasted between 12-18 minutes. Streaming data for each driver/session was recorded on a binary file. Data can be extracted from the binary files at any desired sampling rate. As an example, extracting the data from one the binary files using a 50mS sampling rate can generate approximately 20,000 instances for one of the sensors. The data was later labeled with the following 12 maneuvers using the visual information captured by the video cameras: ChangingLeft, ChangingRight, CrossingShoulder, Not-OnRoad, Passing, Reverse, SlowMoving, Starting, Stopping, Tailgating, TurningLeft, TurningRight, and U-Turn.

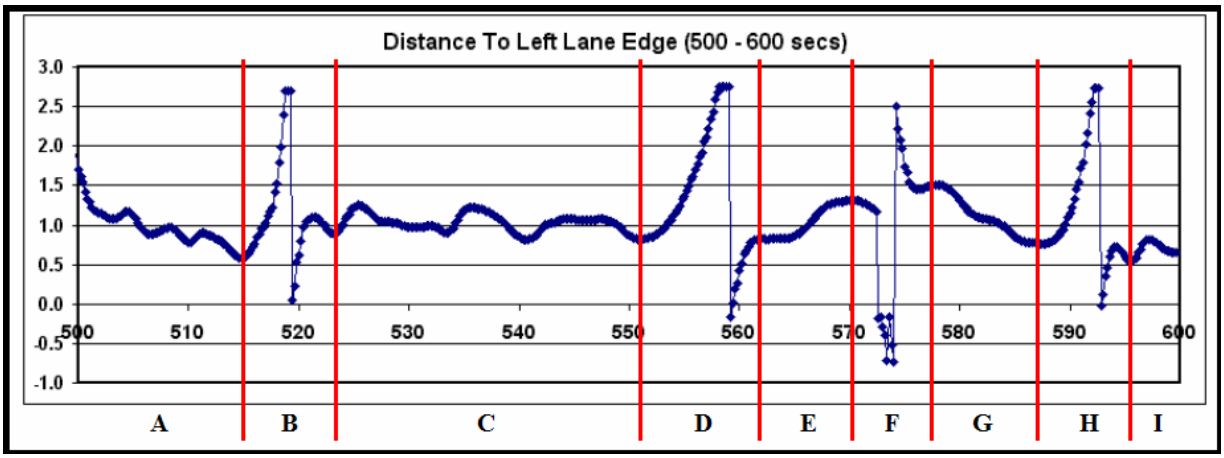

Figure 1: distToLeftLaneEdge stream sample segment.

Figure 1 shows a 100 seconds sample segment of one such stream parametric data for sensor labeled distToLeftLaneEdge, from one of the driving sessions. The horizontal axis represents elapsed time in seconds, and the vertical axis represents the sensor readings indicating the distance in meters that the vehicle's center is from the lane's left edge boundary. A similar, almost mirror image exists for the right edge recorded by a different sensor (distToRightLaneEdge). The vertical bars along the curve are close approximations to events occurring along the sequence, which are to be explained next.

The distToLeftLaneEdge stream sensor data tracks a maneuver where the vehicle intends to cross from his lane to an adjacent lane. In the case where the maneuver is to change to a lane at its right, the farther the vehicles center distances itself from the left lane edge, the greatest the distance recorded by the sensor as it progresses throughout the maneuver. When the vehicle's center crosses the right edge lane boundary, the distance from the left edge is at its maximum value just before it crosses the right edge, and then re-starts at zero after crossing this threshold point, marked by the big discontinuity in the signal, indicating its relative position with respect the new lane.

For the sequence of Figure 1, the vehicle is on a three-lane highway, which we will just designate as leftmost, center, and rightmost lanes for this discussion. This sequence portion of the drive run is at a stage where the driver is cruising in the leftmost lane (segment A). The driver then crosses into the center lane, its maneuver captured in segment B. As can be seen in segment B, the value for the sensor keeps increasing as an indication that the vehicle is moving farther and farther away from the left edge, until it crosses the threshold point, when it then re-starts the reading with respect to the new lane. The rest of the sequence captures how the driver continues cruising along the center lane (segment C), until it crosses into rightmost lane (D). It cruises briefly in the rightmost lane (E) for a few seconds, as it is returns to the center lane (F) to pass a slow moving vehicle that was in the way in the rightmost lane. It passes the vehicle (G) and returns to the rightmost lane (H), after which continues cruising in the rightmost lane (I). Note that sequence F-G-H represents the

passing maneuver. Also note that the effect in segment F is opposite to what we have seen before for segments B and D, since the vehicle is moving to the left, closing the gap to the left lane edge rather than distancing away from it.

One of the reasons for which this particular stream sensor data was selected among the rest is because its graphical representation can clearly visually show a changing lane maneuver as it happens. Few other sensors can show a strong correlation between a maneuver and a sensor stream output, like the brake sensor and the Stopping maneuver. The changing lane maneuver is among the top most common and studied maneuvers. It is manifested in many ways during a normal drive, whether for passing another vehicle, changing to a faster moving lane, to have better view of the road ahead, to take an exit ramp, or even during a drastic evasive maneuver to maybe avoid an obstacle or debris on the road. Data shows that the average changing lanes maneuver is of very short duration, in the order of 2-or-3 seconds range. It provides an excellent study case to help illustrate some of the issues and challenges associated with streaming data, in pursuing of an instance selection solution for our application. As such, we will make references to Figure 1 throughout the rest of the paper discussion.

4. CHALLENGES

As mentioned earlier, instance selection denotes finding relevant data points among massive data[2, 13], resulting in a data reduction, or scaling down, while still maintaining the essence of the data. From a data mining perspective, an ideal outcome for instance selection would be that the resulting subset of the original data points should yield the same results or conclusions as if the entire data set were used. In the case of streaming data, unfortunately, we would never have access to the entire data set, nor can afford to wait for it. Streaming data consists of a continuous and unbounded flow of input data, reported from sensors at regular intervals. The amount of data generated cannot be contained within memory bounds. Traditional data bases have the advantage that they have a snap-shot of the data where processing can be performed off-line, and can be executed over several passes of the data if needed, whereas streaming data analysis has to be performed on-line, and usually have one shot at the data, so in most cases it has to be performed in one pass. As such, the typical expectation from streaming data algorithms is that results are a close approximation, as it is not possible to have all the data, and several summarization algorithms have been developed to provide such approximations. We will review some of them in the Related Work section. Throughout the discussion, we will also review some of the requirements and expectations for our application.

Since we cannot collect, store, and process the entire stream, we study portions of a stream using a window as a temporary snapshot of the data. A window can normally be referred by its size, that is, the number of data points be referred for the stream few instances at a time, by creating a window, of size N, where N represents the number of instances within the window. The data within the window is evaluated and results generated, such as statistics, threshold checks or some data distribution. Typically for streaming sensor data, this is implemented as a sliding window, as it allows focusing on most recent values that are more indicative of current conditions. In sliding windows, any new input data point is added to the window, requiring that an existing point be removed through some replacement scheme, typically an older data point, or some less representative point based on some statistics. Our challenge is to determine what is an appropriate window size, how we take advantage of this technique into detecting relevant instances, and in a smaller note, how to keep low the overhead created by the addition and replacement of data points to the window. In general, with new data constantly arriving even as old data is being processed, amount of computation time per data element must be much lower than the data rate.

Most significant challenge regarding implementation of windows for our application is the determination of the window size. Window size is typically bounded by memory constraints, but for our application, there are several more profound considerations. One characteristic of any window is that they inherently introduce a smoothing effect and delay into the signal. Although smoothing is a desirable byproduct, as streaming data is often noisy, it also causes that shifts and trends in the data take longer to take notice. If the window is too small, it is more susceptible to fluctuations in the data and noise. If the window size is too big, it will take longer to detect a change. This is most significant, as this is a mission critical type application. Prompt detection is a must if we are to try minimizing distractions to the driver in order to minimize the likelihood of an accident. As such, we should minimize introducing many delays into the system.

Another consideration, with implications for the window size, is that mission critical applications like ours carry very high expectations with respect to response time. For example, that in the best cases, it may take at the very least 2-to-3

instances or data points in a sliding window to detect a noticeable change in the data, assuming no outliers. If we were to be using a resolution of 200mS per data point, this would translate to about half a second delay before being capable of detecting a change. This is quite significant considering that some maneuvers may take as little as 2-to-3 seconds to execute. The challenge is fast detection, and fast response with little delays. One solution to the delay is to implement a higher resolution window. But care should be taken as a higher resolution implies more execution time for the processing unit. For example, in the 200mS case above, if we were to use instead 50mS, this will automatically quadruple the number of previously available instances or data points than there were when using 200mS. In addition, the processing algorithm should be even faster enough to handle the load, as it will be now doing four times as much work for hundreds of streaming sensors data. One inherent property of streaming data not present in traditional databases is the temporal component of the data. This is one key aspect for our application, in particular when considering that a maneuver correspond to a sequence of events. As such, a challenge for our resulting instance selection algorithm is that it should guarantee the capturing of the minimum moments needed to define or distinguish a sequence of events.

We now examine the data variability by simply contrasting the differences between segments B, D and H in Figure 1. These three segments represent the same ChangingRight lane maneuver. Segments B, and H are quite similar, though B seems to be sharper and drastic in its execution, while H seems to have had smoother actions. They both significantly differ from segment D, which seems to be 50% longer in duration than B and H. In general, the variability of the sequences depends on many factors, such as the traveling speed, road conditions, traffic load, time of the day, different users for same vehicle, and many others, causing that signals could be either shrunk, or expanded in time relative to each other for the same maneuver.

One of the main objectives of the application is to detect maneuvers but there is one caveat. The earlier we can detect a maneuver, the earlier we can prevent a distraction from one of the communications device from occurring. It helps little if we could only discern halfway through the maneuver, or when it is almost over, hence, it is preferable that we detect the maneuver before, or just after, it starts. Referring to Figure 1, this implies that to be successful in minimizing distractions, we need to be able to detect the maneuver somewhere before or just after the boundary points between A-B, C-D, E-F, and G-H, which bring us to the final challenge and consideration to be addressed in this section. Noticeed in Figure 1, almost always when a maneuver is about to start, there is a little dip in the signal, which can better be observed in the C-D boundary. This characteristic could be utilized to indicate or predict when a maneuver is about to happen, except that similar dips can be observed during regular driving while cruising in the same lane (segment C). However, using this clue alone could lead to many false-positives due the variability observed from the weaving in and out within a lane, maybe the result of road conditions, driving habits, or simply a pre-indication of an intend to change lanes. As such, this sensor alone cannot be used solely to detect a maneuver by itself, in particular, if trying to identify the maneuver before it starts. In reality, a maneuver is a combination of factors and conditions. This implies that we cannot just consider streams individually but we should somehow include the influence of some other streams. This imposes a significant challenge as the amount of variability and combinations could be limitless. Up to this point we have addressed a single stream, but in reality, whichever instance selection method is implemented, it should be one capable to interact with the selection for other streams as well.

5. RELATED WORK

Methods that generate a smaller representation of the data we have divided in two main groups: sampling and data transformations. Sampling refers to methods that generate a summarization of the data through some form of counting over a stream. These methods maintain statistics and can be used in detecting level shifts and outliers for streaming data.. Data transformation refers to methods developed such as wavelets and linear transformations, which have been very successful in achieving significant data reduction. A sampling algorithm tries to collect an un-biased representative subset of a large data set (a stream in our case) in order to make generalizations about the entire data set[17]. Much work has been generated in this area, as many data streams are suitable for general counting methods such as lossy counting[14] or to maintain stream statistics over sliding windows[4]. From a sampling perspective, these methods are useful for identifying most frequent item sets or elements. They are mostly focused in maintaining distributions or histograms for single streams. They would require that parametric continuous data be grouped into bins or buckets, each treated as an individual item, which depending on the size of the bucket could contribute to additional delays into detecting a change.

Other similar methods are implemented with different window levels[1] or as and adaptive window sampling for multidimensional[8] and multiple data streams[21]. Their common characteristic is the implementation of sub-windows, to form or add to a larger window. For the first two methods[1, 8], each sub-window is half the size of its previous window, very similar to our implementation of splitting in half our window to increase resolution, whereas the first method was developed to addresses approximate counts and quantiles and the multiple data streams method captures density distributions, to identify areas called cold (low activity or sparse distribution) and hot (high activity or dense distribution) spots, concepts analogous to our finding interesting or relevant points over the stream. For the last one method[21], there are also similarities to our work, such as adding a basic smaller window to a larger window, and it is optimized for calculating the best-fit slope. It has a fixed response time for fast execution of thousands of streams with response time measured in a few seconds. Our implementation requires resolution in the order of milliseconds, and we implement various time resolutions.

The implementation of the sliding windows facilitates the mechanisms for detecting level changes, critical points or events that we implemented in our work. This was seen in the multiple data streams method in their slope calculation to help identify changes, and can be seen from the implementation of the two window paradigm[9], one window contains current data, the other window the reference data for comparison. Their algorithm is implemented for detecting change in distributions while our concern is change of rate, and their reference window is fixed while ours changes with every basic window. A common theme for sampling methods is that they are geared toward generating synopsis for query optimization as they can provide statistics about their data such as frequent items sold in a store, areas of major network traffic for re-routing, or movement of related stocks from a stock tickler. Although some of them can provide quick updates, in the matter of seconds, they usually they try to keep statistics over a larger window timeframe. Ours being a mission critical application requires of very fast response. Our application requires very short synopsis, as most maneuvers are over in a few seconds. For example, a U-Turn maneuver has little bearing with the next maneuver, like changing lines or making a turn.

Regarding transformations, much methods and techniques have been developed to support mostly finite time series. These techniques decompose the series into subcomponents through mathematical transformations such as Fourier transforms or generation of Haar wavelets. The success of these techniques relies in significantly reducing the data size by capturing a set of coefficients that summarizes the signal. These coefficients can be used for similarity comparisons to other signals using distance measurements such as Euclidean distance. The coefficients can be also used as input parameters into regression algorithms such as ARIMA (auto regressive integrated moving average) models to predict future data points.

Despite these desirable properties, there are various drawbacks with respect to our application. For the Fourier transform, its computation using the fastest algorithm takes $O(N\log N)$, too expensive for our application. Haar wavelets transformation[15, 7, 5, 6, 16] seems to be more suitable for our application, as their computation can be performed linearly. Still, both Fourier and Haar transformations require a significant amount of data points for it to become efficient. For example, in the case of wavelets, the window size normally has to be an even power of 2, that is, 2k. Their implementation for sliding windows also imposes few challenges of their own. An alternative to Haar wavelets, with respect to data reduction and final waveform, can be achieved using a piecewise aggregate approximation [KCPM00]. This method can be implemented on-line by simply calculating an aggregate such as mean or median after a set number of data points calculated every 2^{nd}, 4^{th}, 8^{th}, 16^{th}, …, 2^{k} data point.

Other time series techniques that grasp the essence of our intend, which is to capture most relevant data points in a stream, include methods such as the landmark algorithm [19], or by adaptively approximating the stream through linear approximations such as the piecewise constant approximation[3] , and piecewise linear approximation[10, 18] . These techniques mostly use an error bound criteria whether with respect to a similarity Euclidean distance error, or a cumulative statistic boundary. The level of reduction achieved by these techniques is a direct consequence of the selected error bound criteria. The common theme for the transformation methods has been mostly of similarity search for sequence matching. This is very difficult to implement for our application of detecting a maneuver, at least before the maneuver happens. Referring to Figure 1, these methods can be developed to easily identify segments B, D, F and H as a ChangingLeft or ChangingRight maneuver, but only after their completion, which would be too late for our application.

6. ADAPTIVE SAMPLING

A maneuver consists of a combination of many different factors such as speed, heading, lateral acceleration, road conditions and others. We established during the challenges section that one sensor alone could not be trusted to detect a maneuver by itself. As a result, it would seem that we would have no choice but to monitor all sensor streams in order to detect a maneuver. But we are only interested in capturing those streams that have relevant data as an indication of a maneuver. From the related work, we have shown that methods that can be applied to instance selection such as the transformations treat each sensor individually, and do not address multiple stream interaction, as they are mostly concerned with similarity sequence matching, whereas various sampling methods are mostly geared toward generating synopsis for query optimization for summarization of most frequent items.

The above suggests that sampling or transformations alone are not sufficient for our instance selection. There are several reasons (or challenges) (1) not all intervals of a stream need to be treated equally; (2) in order to catch critical events before or when they are about to happen, denser sampling may be needed for certain segments (i.e., simple sampling does not work here as it often treats all segments equally); and (3) a single stream alone may not be sufficient to determine a critical event.

The figure below, built from synthetic data, will help illustrate our approach. It represents the relation between a change in the acceleration data stream with respect to the output to a lane edge detection edges sensor. This is a pattern that has been observed, with some variability and skewness, in some of the samples of our driving sessions data. It relates to those occasions when a driver might accelerate just before changing lanes. The signals are represented normalized and the horizontal axis represents elapsed time in seconds.

The figures illustrates promises of instance selection: (1) it is possible to sample in different rates as events do not occur all the time; (2) an adaptive window size may serve the purpose for instance selection; and (3) a maneuver may be a combination of various events. The left figure is before applying linear approximation transformation[10] and the right one is after transformation. As can be seen from the figure, with so few instances, it may be difficult to define a pattern in the sequence and when it determines a trend, it could be too late to act. The interesting segments are between 1 and 4, those segments before 1 or after 4 are not really relevant as no events occur. This suggests that an adaptive sampling approach should be appropriate.

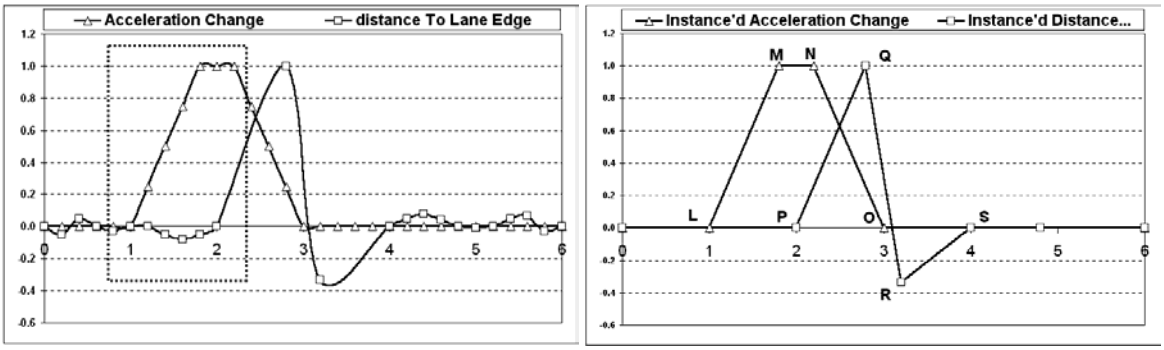

Figure 2: Pre-transformation Post-transformation

For simplicity, we will refer to the two streams in the figure as A-stream (acceleration) and D-stream (distance to lane edge). The window size is defined by some time unit (e.g., in seconds), and determines the number of data points that can be processed in a particular time period. Given a fixed processing power or fixed memory size, a smaller window allows more instances to be sampled and processed. With this property, we propose a concept of elastic window to realize adaptive sampling: the window size can change between Wmin and Wmax predefined for each stream, and it increases slowly when no event is detected, and reduces quickly when a potential event is spot so that more instances can be sampled. When some transformation (e.g., linear approximation) is combined with adaptive sampling, we expect a significant reduction in data to be processed without compromising detection capability. Considering the two events (LM and PQ) in the above right figure, because an elastic window allows denser sampling, a trend can be determined

way before it reaches M or Q by shrinking the window to a 1/4th of its original size. This will allow early alert for reducing window sizes of other associated streams.

To make matters worse, the changes in A or D streams alone can often have multiple explanations. As an example, referring back to Figure 1, we can observe portions of the C segment that could suggest that a lane change maneuver is about to start. The two changes occurring together, however, can indicate a lane change maneuver. This phenomenon suggests that in addition to instance selection that captures salient data points for event detection, we also need to take into account a combination of different streams in detecting a maneuver. This issue connects instance selection with feature selection, a subject addressed on the instance selection section. The underlying idea is to form small subsets of streams that need to be monitored and considered together for maneuver recognition. Since it is costly to monitor all possible streams at a time and different streams may have changes started at different times, we propose to identify a representative lead stream for each subset of streams. Ideally, this lead stream should be the one in which its events often occur earlier than other streams in the subset. Using the simulator as our test bed, we will investigate (1) various transformation methods suitable for streaming data, (2) optimal adaptive windows for different sensors to achieve efficient adaptive sampling, (3) subsets of associated streams that can effectively detect maneuvers, and (4) identifying lead streams and studying their synchronization with associated streams in effective instance selection for efficient pattern detection.

7. AN EMPIRICAL CASE STUDY

We implement the adaptive sliding windows as follows. We first established arbitrarily to maintain a small buffer to contain 8 data points at all times for reasons that soon will be apparent. We then establish a minimum window size by taking in consideration sensor resolution and maneuver duration. Even though many of the sensors are sampled at 60 Hz rate (~ 17mSecs/data point), we limited our highest resolution to 50mSecs/data point. This accommodates most of the sensors and still leaves room for higher resolution if later needed. Since some maneuvers can last as little as 2-to-3 seconds, we then set up our standard sampling sliding window to 1.6 seconds with a resolution of 200mSecs/data point.

This initial set-up was built in order to generate a minimum of at least three different window sizes, with a maximum size of 1.6 seconds, such that in the event of detecting relevant activity in the stream, we could initially increase the resolution for the window by 2X, to a 100mSecs higher resolution and a new window size of 0.8 seconds. If the algorithm determines that the relevant activity is persistent, we could again increase the resolution by 2X, up to 50mSecs/data point (max resolution), and create a new window size of 0.4 seconds. Table 1 summarizes the window sizes and their resolutions.

Window Size (seconds)	Resolution (mSecs/data point)	Data Points
1.6	200	8
0.8	100	8
0.4	50	8

Table 1: Adaptive Windows Resolution

We calculate the slope m of the best line fit using the least squares method using the formula. In the formula on the left below, *x* represents the time component, and *y* represents the stream data value.

$$m = \frac{\sum (x - \bar{x})(y - \bar{y})}{\sum (x - \bar{x})^2} \qquad m = \frac{n(\sum xy) - (\sum x)(\sum y)}{n(\sum x^2) - (\sum x)^2}$$

The formula on the right is a simplification to easily calculate on-line, where *n* represents the number of data points evaluated, which for our case $n = 8$.

Following is a description of the algorithm. We will be making references to Figure 2 as the algorithm progresses. We start the algorithm with the highest resolution, and quickly capture the first 8 data points at a cost of 200mSecs. It then calculates the slope for the window and compares against an initial threshold value. If the result is below some threshold that is particular for each individual lead stream, it immediately switches to the lowest resolution, with a window size of 1.6 seconds (segments M-P of Figure 2). As the stream progresses, it continues capturing data points at the lowest resolution, adding a basic window as a data point, and removing the oldest data point. At each new data point, the slope for the 8 data points is calculated, and compared to the previous window result. If the resulting value exceeds the threshold, is when the algorithm first recognizes that there maybe some relevant activity (point P), and it changes to the mid-resolution. If the activity persists (segment P-Q), as a possible indication of a maneuver, it then switches to the highest resolution. At this point, when the lead stream detects a potential maneuver, all associated streams within the same subset will also adjust the maximum resolution. As soon as the data points reaches the peak of the signal (point Q), or recognized as no longer relevant, the algorithm switches again to the lowest resolution, to minimize the number of instances generated.

Preliminary Results and Discussions. We conduceted three sets of experiments on the streaming data shown in Figure 1. In the first experiment, we set Wmin = 50ms and Wmax = 200ms. The results are shown in Figure 3. The adaptive window starts with its size Wmax. The blue curve is the original stream and the pink curve represents the results after the data processing using the adaptive window algorithm described above. When it detects some upward change, the window size immediately changes to Wmin in order to closely monitor any potential critical maneuvers. It can be seen from the figure that the adaptive window traces the original streaming data quite well in the sense it has some delays for uninteresting segments, but little delays for the big upward curves. Here are the statistical details after the processing using adaptive window: (1) compression rate is 35.7%, (2) reproduction error D(S,R) is 234.7, where D(S, R) is defined in Section 2. These metrics are used to comparison purposes with different Wmax as the next two experiments shown in Figures 4 and 5 for Wmax = 400ms and Wmax = 800ms, respectively. Their statistics are: for Wmax = 400ms, compression rate is 20.2%, and reproduction error D(S,R) = 332.3; and for Wmax = 800ms, compression rate is 9.4%, and reproduction error D(S,R) = 438.8. With the compression rate increases, the reproduction error increases. The latter can also be seen in Figures 4 and 5 that the delays of the pink curves with respect to the blue curves are distinct, which are consistent with our early analysis. This set of experiments shows the process of using the concept of adaptive windows on the streaming data for maneuver detection. In order to timely capture a potential maneuver, we wish to minimize any delays. However, streaming data does not allow for storage of an entire data stream. In other words, we wish to maximize the data reduction. In order to balance the two demands, we need to explore further data reduction for data mining in order to keep selected data with little information loss.

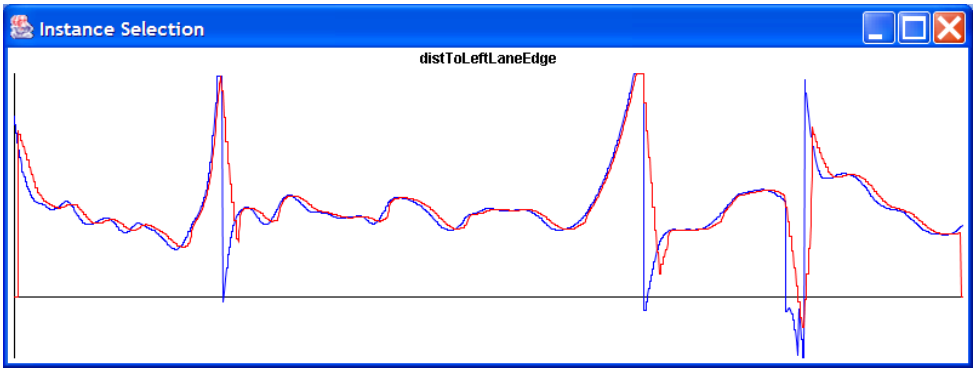

Figure 3 – Wmax = 200ms

Hence, we conduct another experiment using liner approximation to the same streamlining data of Figure 1, as shown in Figure 6. It mimics the problem addressed in Figure 2 of section 5 when implementing an aggressive reduction if just capturing the most relevant points associated with a maneuver. Although the algorithm is very effective at summarizing the stream with very few instance points, it does not resolve the relevant points until well into the maneuver.

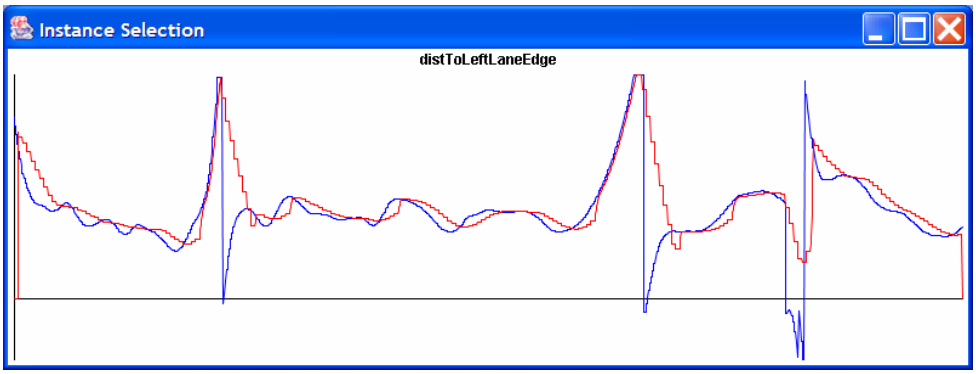

Figure 4 – Wmax = 400ms

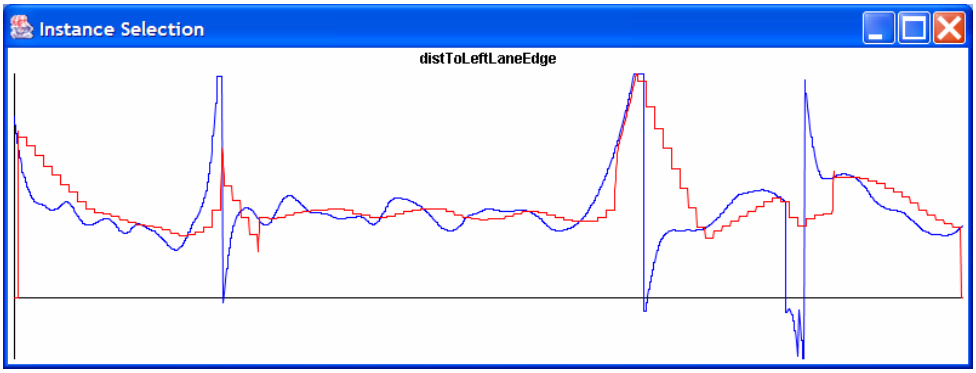

Figure 5 – Wmax = 800ms

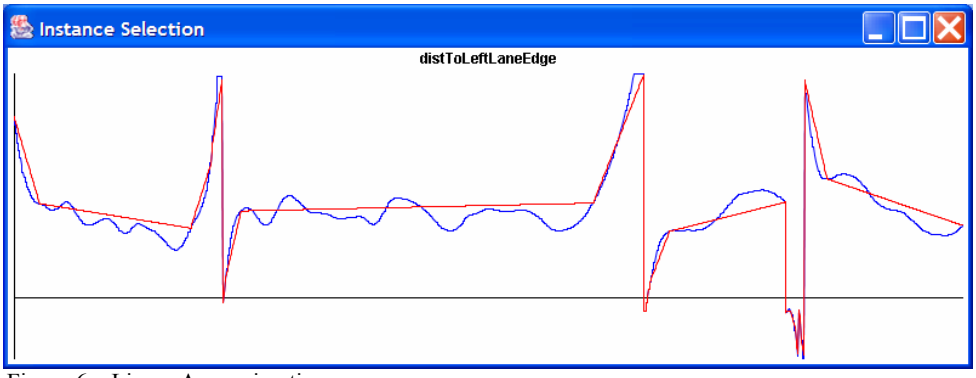

Figure 6 – Linear Approximation

To make matters worse, each sensor can have different behavior in representing certain maneuvers. Therefore, adaptive windows are logically different for different data streams. Although linear approximation is not suitable for timely maneuver detection, it can be employed to summarize data for data mining purposes to mine patterns pertinent to critical maneuvers.

8. CONCLUSIONS

In this work we study the problem of instance selection in the context of streaming data for maneuver detection. We propose a concept of adaptive window in reducing the number of instances to minimize the workload for central processing unit which may need to work with hundreds of sensor that emit streaming data. A closely relevant issue is

how to reduce information loss while performing instance selection. Our initial empirical study shows the feasibility of adaptive window and suggests many possible lines of research. First, we will investigate how to automatically design and implement an optimal adaptive window that can achieve a balance between the minimum compression rate and the minimum reproduction error for each of many in-vehicle sensors. Second, we will explore how to combine transformation techniques such as linear approximation with adaptive window so that we can store sufficient information for mining patterns for critical maneuvers. Third, we will combine instance selection with sensor selection to identify subgroups of sensors, and key sensors that can act as lead sensors for early alert. As we believe they could significantly reduce the instances if relevant instances properly identified.

REFERENCES

1. A. Arasu and G.S. Manku, "Approximate Counts and Quantiles over Sliding Windows", PODS, 2004.
2. A.L. Blum and P. Langley, "Selection of relevant features and examples in machine learning", Artificial Intelligence, 97:245-271, 1997.
3. K. Chakrabarti, E. Keogh, S. Mehrotra and M. Pazzani, "Locally Adaptive Dimensionality Reduction for Indexing Large Time Series Databases", ACM Transactions on Database Systems, 2002.
4. M. Datar, A. Gionis, P. Indyk, and R. Motwani. "Maintaining stream statistics over sliding windows", In ACM-SIAM SODA, 2002.
5. A.C. Gilbert, Y. Kotidis, S. Muthukrishnan and M. J., Strauss, "Surfing Wavelets on Streams: One Pass Summaries for Approximate Aggregate Queries", VLDB Conference, 2001.
6. A.C. Gilbert, Y. Kotidis, S. Muthukrishnan and M. J., Strauss, "Surfing Wavelets on Streams: One Pass Wavelet Decompositions of Data Streams", IEEE, June 2003.
7. S. Guha, C. Kim and K. Shim. "XWAVE: Optimal and Approximate Extended Wavelets for Streaming Data", VLDB Conference, 2004.
8. J. Hershberger, N. Shrivastava, S. Suri and D. Tóth, "Adaptive Spatial Partitioning for Multidimensional Data Streams", In Proceedings of the 15th ISAAC Conference, 2004.
9. D. Kifer, S. Ben-David and J. Gehrke, "Detecting Change in Data Streams", In Proceedings of the 30th VLDB Conference, 2004.
10. E. Keogh, S. Chu, D. Hart and M. Pazzani, "An Online Algorithm for Segmenting Time Series" , ICDM, 2001.
11. E. Keogh, K. Chakrabarti, M. Pazzani and S. Mehrotra "Dimensionality Reduction for Fast Similarity Search in Large Time Series Databases", Journal of Knowledge and Information Systems, 2000.
12. H. Liu and H. Motoda, "Feature Selection for Knowledge Discovery & Data Mining", Boston: Kluwer Academic Publishers, 1998.
13. H. Liu and H. Motoda, "Instance Selection and Construction for Data Mining", Kluwer Academic Publishers, 2001.
14. G.S. Manku and R. Motwani, "Approximate frequency counts over data streams", In Proceedings of the 28th VLDB Conference, 2002.
15. S. Papadimitriou, A. Brockwell, and C. Faloutsos, "Awsom: Adaptive, hands-off stream mining", In Proceedings of 29th International Conference on Very Large Data Bases, pages 560-571. Morgan Kaufmann, 2003.
16. I. Pivanov and R. Miller, "Similarity Search Over Time-Series Data Using Wavelets", ICDE, 2002
17. B.H. Park, G. Ostrouchov, N. F. Samatova and A. Geist, "Reservoir-based Random Sampling with Replacement from Data Streams", In proceedings of the 4th SIAM International Conference on Data Mining, 2004.
18. T. Palpanas, M. Vlachos, E. Keogh, D. Gunopulos and W. Truppel. "Online Amnesic Approximation of Streaming Time Series", In Proceedings of the 20th ICDE Conference, 2004.
19. C.S. Perng, H. Wang, S. Zhang and D.S. Parker, "Landmark: A New Technique for Similarity-Based Pattern Querying in Time Series Databases", In Proceedings of the 16th ICDE Conference, 2000.
20. K. Torkkola, S. Venkatesan and H. Liu, "Sensor Selection for Maneuver Classification", In proceedings of the 7th IEEE International ITSC Conference, 2004.
21. Y. Zhu and D. Shasha, "Statstream: Statistical monitoring of thousands of data streams in real time", In Proceedings of the 28th VLDB Conference, 2002.