VEDAS: A Mobile and Distributed Data Stream Mining System for Real-Time Vehicle Monitoring

Published in the Proceedings of the 2003 International SIAM Data Mining Conference

Hillol Kargupta, Ruchita Bhargava, Kun Liu, Michael Powers,
Patrick Blair, Samuel Bushra, James Dull
Computer Science and Electrical Engineering Department,
University of Maryland at Baltimore County,
Baltimore Maryland 21250 USA
{hillol, ruchita1, kunliu1}@cs.umbc.edu

Kakali Sarkar, Martin Klein, Mitesh Vasa, and David Handy
AGNIK, LLC, Maryland, USA
{kakali, martinklein, davidhandy}@agnik.com

Abstract
This paper presents an overview of an experimental mobile and distributed data stream mining system that allows real time vehicle-health monitoring and driver characterization. It offers the motivation behind this application, explains the system architecture, discusses many challenges that the project faced, and shares some of the adopted solutions. The main contribution of the paper is our experience in building one of the very early distributed data stream mining systems for wireless applications that performs most of the data analysis related tasks using light-weight on-board computing devices. This paper points out that the distributed data mining technology can play a key role in solving real-life problems in a mobile application environment where computing, storage, power, and communication resources are limited. The paper also illustrates how privacy-preserving distributed data mining can play an important role in this type of applications.

1 Introduction

Mining distributed data streams in an ubiquitous environment offers many exciting possibilities. Monitoring patient health on a regular basis, security surveillance, large sensor networks for defensive measures are some examples. Data mining may play a very important role in these applications. Several years of research on distributed data mining [14] and data stream mining have produced a reasonably powerful collection of algorithms and system-architectures that can be used for developing interesting classes of distributed applications for lightweight wireless environments. In fact an increasing number of such systems are being reported in the literature. The MobiMine system for mobile stock monitoring/mining [9] and health monitoring system [8] are some examples.

This paper reports the development of VEhicle DAta Stream mining (VEDAS) system, for monitoring and mining vehicle data streams in real-time. VEDAS is designed to monitor vehicle fleets using on-board PDA-based distributed data stream mining systems and other remote modules connected through wireless networks. Consider a nationwide grocery delivery system which operates a large fleet of trucks. Regular maintenance of the vehicles in such fleets is an important part of the supply chain management. Normally commercial fleet management companies get the responsibility of maintaining the fleet. Fleet maintenance companies usually spend a good deal of time and labor in collecting vehicle performance data, studying the data off-line, and estimating the condition of the vehicle primarily through manual efforts. Fleet management companies are also usually interested in studying the driving characteristics for maintenance related reasons. One may also think of many exciting possibilities such as real-time drunk driving detection through on-board analysis of vehicle data streams. The VEDAS system aims to offer a viable technical solution to these real-life problems. Similar applications also arise for monitoring the health of airplanes and space vehicles. There is a strong need for real-time on-board monitoring and mining of data (e.g. flight systems performance data, weather data, radar data about other planes). The VEDAS system can also be applied to this aviation safety domain where it monitors airplanes and space vehicles instead of automobiles and trucks.

The main unique characteristics of the VEDAS

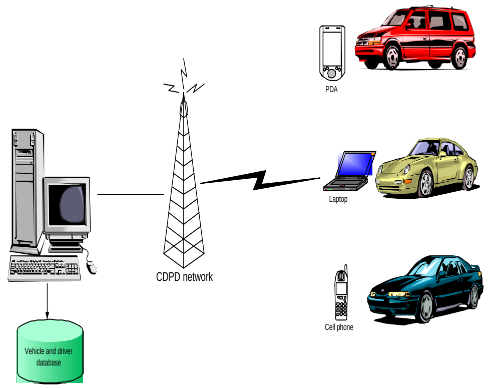


Figure 1: Conceptual overview of the VEDAS system.

system are as follows:

1. On-board data stream management and mining using PDAs or other comparable light-weight mobile computing devices.
2. Distributed mining of the multiple mobile data sources with little centralization of the data.
3. Designed to pay careful attention to the following important resource constraints: (i) Minimize data communication over the wireless network; (ii) minimize consumption of power in the on-board computing device since battery power is a limited resource in certain classes of applications of VEDAS where the PDA cannot be hooked up with the on-board power supply unit all the time; (iii) minimize on-board data storage and the footprint of the data stream mining software; (iv) minimize computing and memory usage for analyzing continuous data streams.
4. Respect privacy constraints of the data.

Section 2 discusses the overall architecture of the VEDAS system. Section 3 offers a closer look at its vehicle health monitoring module. Section 4 describes driver characterization module and consider the problem of detecting unusual driving patterns in real-time. Section 5 presents the capability of VEDAS in performing distributed fleet-level analytics where preserving data privacy is important, since it involves monitoring drivers. Section 6 presents power consumption characteristics of a few of the data mining algorithms used in VEDAS and argues the rationale behind some of the design choices from the power consumption perspective. Section 7 offers the future directions and the conclusions of the paper.

2 An Overview of the VEDAS Architecture

VEDAS is an experimental mobile data stream mining environment where the mobile devices perform various non-trivial data mining tasks on-board a vehicle in real-time. VEDAS analyzes the data produced by the various sensors present in most modern vehicles. It continuously monitors data streams generated by a moving vehicle using an on-board computing device, identifies the emerging patterns, and reports these patterns to a remote control center over low-bandwidth wireless network connection, if necessary. This section presents an overview of the architecture of the system and the functionalities of different modules.

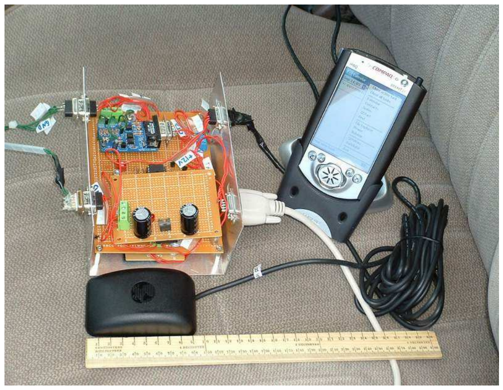


Figure 2: VEDAS system connected to a car. The scantool is connected to the car through the OBD-II connector. A GPS device is connected for location information and the iPAQ downloads the data from the car for analysis.

The current implementation of VEDAS mines and monitors only the data generated by the vehicle's on-board diagnostic system and the Global Positioning System (GPS). It is currently implemented for WinCE based mobile devices like Personal Digital Assistants and handheld computers. The overall conceptual process diagram of the system is shown in Figure 1. It shows multiple vehicles installed with the VEDAS software which can be concurrently monitored by a central site. The vehicles can have different types of computing devices ranging from PDA's to special-purpose tablet PCs monitoring and collecting and analyzing the data generated by the vehicle. Any standard commercial data network can be used for the wireless communication. The VEDAS system is comprised of four important components:

1. Hardware interface for the on-board diagnostic (OBD-II[1]) data bus that couples with our software. This module was developed in-house for interfacing with the on-board OBD-II data bus. It also contains a GPS module.
2. On-board data stream management and mining

[1]OBD-II is a standard for on-board data generated by the vehicle established by Society of Automotive Engineers.

module: The VEDAS system offers a communication system and a run time environment for performing on-board data analysis and management. The on-board PDA-based module monitors, manages the data stream, and triggers actions when unusual activities are observed. The on-board module connects to the desk-top-based remote control station through a wireless network. The system allows the fleet managers to monitor and model vehicle behavior remotely without necessarily downloading all the data to the central monitoring station over the expensive wireless connection.

3. Remote desktop-based control station for fleet managers: The VEDAS control station supports the following main operations (i) Interacting with the on-board module for remote management, monitoring, and mining of vehicle data streams. (ii) Interactive statistical data analysis. (iii) Interactive online PCA-based vehicle health regime monitoring. (iv) Visualization of the driving characteristics.

4. Privacy management module: This module plays an important role in the implementation of the privacy policies. For example, the fleet drivers may have a quite justifiable objection against continuous monitoring of their driving behavior. However, they may be willing to allow the management to analyze the data for detecting drunk drivers as long as the privacy of the sober drivers is not compromised.

The following sections explain the hardware and software design of these components of VEDAS.

2.1 Interfacing with the On-board Data Bus

Most modern vehicles generate a large amount of data continuously using the sensors that monitor the different process parameters in various sub-systems (e.g. transmission system, engine system, and fuel system) in a vehicle.

The VEDAS system accesses these data streams by connecting a PDA (or PDA-like device) to the OBD-II data bus on a vehicle. In 1996, to comply with Environment Pollution Agency's emission norms US government made the interface to OBD-II of vehicles mandatory for all vehicles. OBD-II was developed as a standard by SAE to provide a mechanism for universal inspection and diagnosis. The OBD-II interface can be used to access a rich set of generic and manufacturer specific parameters collected from various sensors in the car. On-board Diagnostic data is collected by connecting a serial cable with the OBD-II data bus with appropriate middleware. The OBD-II interfacing hardware

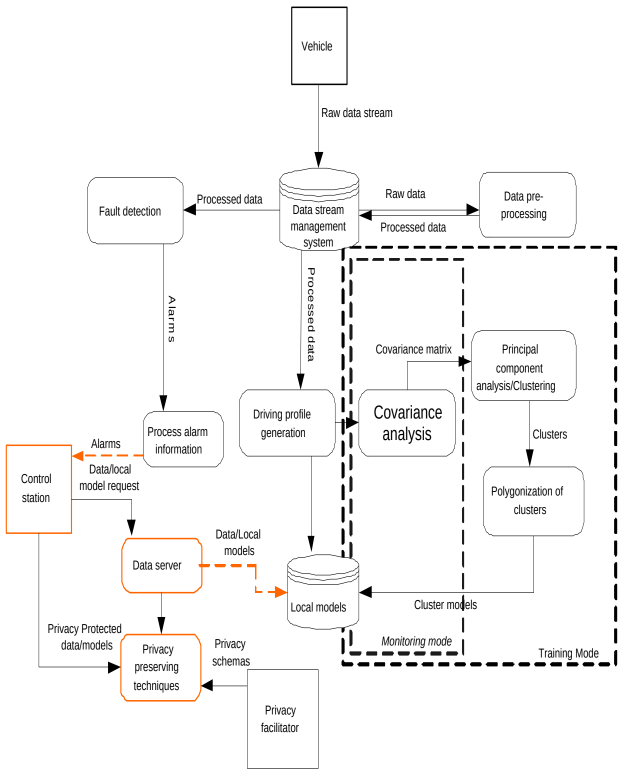


Figure 3: The system architecture for the on-board data management and mining module. It also shows part of the control station module.

module that we have developed (shown in Figure 2) can be used to sample only the desired set of features at regular intervals and it can be controlled by the software on the PDA or remotely through the control station interface. In addition to the OBD-II interface we also have a Garmin GPS device connected to the PDA. The GPS device provides geographical location information for the vehicle, which in conjunction with the vehicle performance data may be used for model building and analysis. The GPS-based functionalities of the system is still under development.

2.2 On-board Data Stream Management and Mining

The data collected through the OBD-II interface is first analyzed by the data mining software running on the on-board PDA. The on-board system architecture is shown in Figure 3. So far we have used two different PDA architectures for testing the VEDAS system: (i) Compaq iPaq Pocket PC is a thin, lightweight portable WinCE 3.0 based personal digital assistant. It has a Intel StrongARM 206MHz RISC based microprocessor with 32MB of RAM and 16 MB of ROM. (ii) HP Jornada 690 is a WinCE 2.1 based handheld PC. It runs on a 133MHz 32-bit Hitachi SH3 processor and has 32MB of RAM.

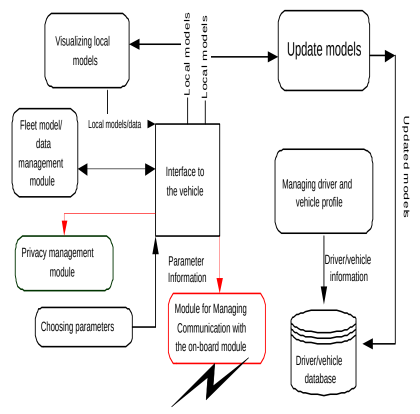
The on-board system is designed to communicate with the central control site using low bandwidth channels such as the Cellular Digital Packet Data (CDPD) wireless network (used in the experiments reported in this paper). CDPD is a data transfer technology which uses cellular phone technology at speeds up to 19.2 Kbps to get mobile data access. A large fraction of the data mining tasks is performed on-board the vehicle to avoid the high communication costs associated with the cellular network and also to maintain the privacy.

The data collected through the OBD-II bus is buffered and managed by the Data Stream Management (DSMS) system. The DSMS provides a mechanism to systematically control the access to the data. It offers a variety of data management primitive operations that are used to support continuous statistical queries over the stream data. The data set consists of both real and categorical data. For the experimental results presented in this paper we used data from two Ford car models: 2001 Ford Focus and a 2003 Ford Taurus. We collected both the generic OBD-II parameters and the manufacturer enhanced parameters. Experimental results reported in this paper used a data set comprised of 64 real valued parameters. In addition, the system also had access to many discrete attributes that often show the status of various sub-systems and fault codes.

The on-board module keeps track of different statistical aggregates like the mean, variance, and covariances. The Data Stream Management System (DSMS) module provides basic primitive operators for computing these statistical aggregates. This allows efficient execution of the continuous queries needed for monitoring the statistical properties of the observed parameters. The computed statistics are used for performing different normalization operations on the data. They are also used for monitoring the changes in the underlying distribution, if any.

After pre-processing, the on-board module reduces the dimensionality of the data and constructs a low dimensional representation of the data. The current implementation supports the following techniques for constructing the new representation: (i) incremental Principal Component Analysis (PCA) [7]; (ii) incremental Fourier transformation; (iii) online linear segmentation. All of the above techniques do not necessarily have to run all the time. They can be turned on and off remotely using the control station interface of the VEDAS system.

The VEDAS system also supports a collection of light-weight unsupervised techniques to analyze the data streams and they are listed in the following: (i) incremental clustering; (ii) generating cluster descriptions using techniques from computational geometry; (iii) a collection of statistical testing and analysis techniques. Currently the VEDAS on-board module does not support any supervised or predictive technique. However, we plan to explore this aspect in future.


Visualizing local models
Local models
Local models
Update models
Updated models
Fleet model/ data management module
Local models/data
Interface to the vehicle
Managing driver and vehicle profile
Privacy management module
Parameter Information
Driver/vehicle information
Choosing parameters
Module for Managing Communication with the on-board module
Driver/vehicle database
Figure 4: The functional architecture of the central control station.

The data stream mining techniques in the VEDAS on-board module are designed and implemented with due attention to the on-board resource constraints. Most of the algorithmic implementations are very different from their traditional counterparts for desktop environments. For example, performing incremental PCA for real-time monitoring with limited computing computing, power, and memory is a very challenging task and the standard algorithmic implementations for desktop applications do not work very well. Rather approximate techniques that demand only occasional eigen analysis do work well in practice. Moreover, energy consumption, privacy issues, and limited communication bandwidth introduce many constraints that made building VEDAS a challenging experience. Rest of this paper shares some of these challenges in details and describes the adopted solutions. However, before that let us discuss the central control station module which provides the interface to the fleet manager for monitoring the vehicles in the fleet.

2.3 Central Control Station Module The Control Station module consists of a server running on a desktop with a database to store fleet and driver information. The system architecture for the control station is shown in Figure 4. The main features of the control station system are as follows:

1. The control station has a visualization module which can be used to view both local models and the fleet level global models. Figure 5 shows the visualization module for an individual vehicle health monitoring data.

2. It provides an interface to control the data mining operations on-board the vehicles. It can also request the vehicles for local models if needed.
3. The control station has an event management service which complements the unusual event detection module that runs on-board the vehicles. In case of any “unusual” event, the on-board module notifies the event management module at the control station which in turn draws the attention of the user.
4. The control station can also be linked with a Geographical Information System which provides mapping tools using the GPS information conveyed by the vehicle. This information can be used in conjunction with the locally built models to gain better information about the system.

The following section discusses some of the VEDAS-modules in details.

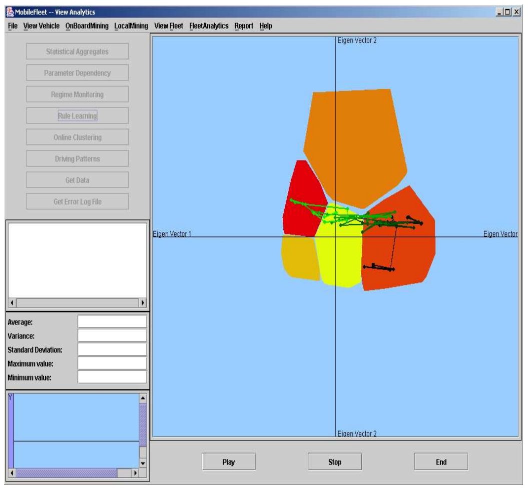


Figure 5: Control station interface: Visual representation of the safe operating regimes.

3 Vehicle Health Monitoring

The vehicle health monitoring module of VEDAS is responsible for tracking the operating characteristics of the vehicle and detecting abnormal patterns from the vehicle health data. This module estimates the distribution of the data using different incremental parametric and non-parametric techniques. In this section, we discuss only one of them that makes use of an incremental operating regime identification technique. It identifies the safe operating regime of the vehicle in the low dimensional eigenspace of the covariance matrix by first clustering the data and then capturing the clusters using techniques from computational geometry. We assume that initially when the vehicle is certified to be in good health condition, we can observe its behavior, gradually generate the clusters, and then use the stable cluster descriptions to define the healthy operating regimes of the vehicle. Later, during the monitoring phase, the module simply notes whether or not the observed data point falls within the safe operating regime in the projected state space of the vehicle. If it does then everything is okay; otherwise the module raises a flag and reports unexpected behavior. The main steps of this process are as follows: (1) Principal Component Analysis(PCA)-based projection of the data; (2) incremental clustering in the projected space; (3) construction of cluster descriptions using techniques from computational geometry. Each of these components is further described in the following.

3.1 Issues in On-board PCA-based Representation Construction PCA is a mathematical procedure which transforms a number of correlated variables into a smaller number of uncorrelated variables called principal components. PCA performs eigen analysis on the covariance matrix of the data. This symmetric matrix is converted into a tridiagonal form using Householder reduction and then eigen-analysis is performed by using the QL algorithm with implicit shifts [5]. The eigenvectors corresponding to the relatively large eigenvalues are used to project the data into a new space. These eigenvectors capture most of the variance in the data.

Changes in the driving characteristics may introduce changes in the eigenvectors of the covariance matrix. Such changes must be quickly detected in order to properly represent the underlying projected state-space. Therefore, continuous monitoring of the underlying eigenspace is important for vehicle health and driver monitoring applications. However, recomputing the eigenvectors on a regular basis is difficult to do in real-time; it can be overwhelming in a resource constrained environment.

VEDAS contains an incremental PCA computation module that makes clever use of matrix perturbation theory for determining when to run the PCA computation module and when not to. It determines the upper bounds on the changes in the eigenvectors and eigenvalues over time. Let us assume that the module already performed the PCA on the data block X_{t-1}. Now it receives another data block X_t. The question is the following: Should we update the covariance matrix and recompute PCA? Matrix perturbation theory may be used for detecting large changes in the spectral distribution without necessarily recomputing the PCAs.

Let us note that the covariance matrix is a symmetric matrix. Let C_{t-1} be the covariance matrix at time $t-1$. Let γ_i and λ_i be the i-th ranking eigenvector and the corresponding eigenvalue such that $\lambda_1 \geq$

$\lambda_2 \geq \cdots \lambda_k$. Therefore γ_1 represents the most dominant eigenvector, γ_2 represents the second-most dominant eigenvector, and so on. Let C_t be the new covariance matrix updated from C_{t-1}. Let $E = C_t - C_{t-1}$. Note that E is symmetric. Let $\lambda_1^{'} \geq \lambda_2^{'} \geq \cdots \lambda_k^{'}$ be the dominant eigenvalues of the perturbed matrix C_t; let $\gamma_1^{'} \geq \gamma_2^{'} \geq \cdots \gamma_k^{'}$ be the corresponding eigenvectors. Using matrix perturbation theory [16] we can write,

$$||\gamma_1 - \gamma_1^{'}|| \leq \frac{4||E||_F}{\delta - \sqrt{2}||E||_F} \tag{3.1}$$

$$|\lambda_1 - \lambda_1^{'}| \leq \sqrt{2}||E||_F \tag{3.2}$$

Where δ is the difference between the λ_1 and λ_2 of the matrix C_{t-1}, sometimes called its eigengap. The Frobenius norm of a matrix E is defined as $||E||_F = (\sum_i \sum_j E_{ij}^2)^{1/2}$. Equation 3.1 assumes that $\delta > \sqrt{2}||\Delta^T\Delta||_F$. Bounds like this can be very useful in designing triggers to initiate re-computation of PCA all over when the underlying distribution has considerably changed. Using this approach the data stream is continuously monitored for changes in spectral distribution.

Any change in distribution results in construction of new models which are then transmitted to the control station. In addition to this, various boundary conditions are checked to find irregular patterns in the vehicle performance. In case of an error an alarm is generated at the control station. The on-board system processes the data so that it can be utilized to perform further advanced data mining operations. Various parameters for controlling the data mining algorithms can be set from the control station and data and generated models can be communicated on demand. The following section presents the incremental clustering module used for identifying the safe operating regimes.

3.2 Issues in On-board Incremental Clustering

The main objective of this module is to identify a collection of polygonal representations in the projected space that correspond to the safe operating regimes. VEDAS tries to address this problem by first clustering the data incrementally. The incremental approach is essential since the data sets cannot be stored on-board for long time. The storage resource constraints simply do not allow storing large quantity of data. Therefore, we need to incrementally construct clusters and generate cluster descriptions for efficient summarization of the clusters.

The current implementation of VEDAS uses a novel incremental k-means clustering algorithm to separate the set into k safe regimes, where k is determined by empirical observations. For each of these safe regimes we compute a polygon to represent it. As new data arrive from a sliding window, we re-cluster after including selected points in the sliding window along with the points making up the vertices of our polygons in presence of information about the estimated distribution within the clusters. This approach has the benefit of maintaining information from the entire data set without requiring us to store all of the past observations.

One of the main challenges in this scheme is to find an appropriate way for constructing polygonal representations of the clusters. VEDAS uses a Delaunay Triangulation based polygonization approach as described in [11] to get an accurate and efficient description of data clusters. Since this method is capable of capturing non-convex clusters, it offers a large improvement over a simple convex hull-based cluster representation. The basic algorithm consists of 3 phases. Let $D = d_1, d_2, d_3, .., d_n$ be our set of n data points where $d_i = (x_i, y_i)$ is an ordered pair on the Cartesian plane, and let $ID(d_i)$ be the cluster identifier for each point d_i where $1 \leq i \leq n$. After clustering the points in our data set, we provide the set D, as well as $ID(d_i)$ for each point d_i in D as input to the polygonization algorithm. In phase 0 we compute the Delaunay Triangulation of D. Delaunay Triangulation gives us a graph where each edge is the side of one or two triangles. Let $T_i = d_i, d_j, d_k$ be one of our Delaunay triangles, where d_i, d_j, and d_k are the vertices of T_i. The triangle T_i possesses the property that its circumcircle contains no point from the set D. In phase 1, each edge $e = d_i, d_j$ from the Delaunay Triangulation is labeled as either an inter-cluster edge or an intra-cluster edge. The edge e is an inter-cluster edge if $ID(d_i) \neq ID(d_j)$and e is an intra-cluster edge if $ID(d_i) = ID(d_j)$.

Next we proceed with phase 2 where the algorithm extracts the boundary edges by analyzing every intra-cluster edge. An inter-cluster edge cannot be a boundary edge since it joins two different clusters. Therefore, the polygonization algorithm disregards all inter-cluster edges in phase 2. Lee et al. describe all of the possible cases that each intra-cluster edge may fall into. After proceeding with their analysis, we have a graph $PG(D)$, which contains precisely enough oriented edges to construct a polygon representation of each C_i. The edges are oriented such that the interior of the polygon is to the left of those edges making up its boundary. Finally, in phase 3 the algorithm finds the circular paths in $PG(D)$. Each path represents the edges of exactly one polygon and each polygon represents one of our clusters C_i. This gives us a polygonal representation of each of the clusters.

One downfall of this algorithm, which may be worth noting, is that the shape of one cluster can be affected by points not contained in that cluster. However, our initial experiments show that this method does

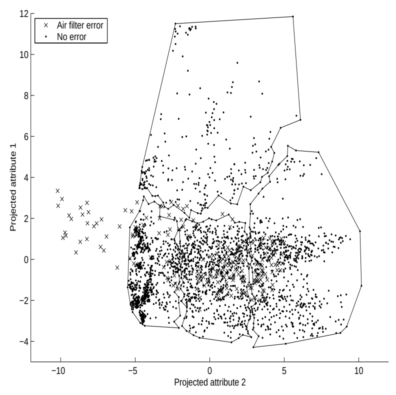


Figure 6: Observations from a vehicle with dirty air filter fell outside the polygons representing the safe operating regimes.

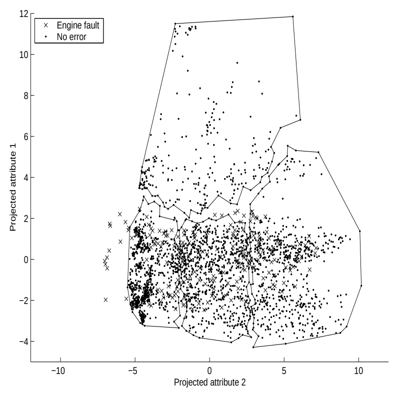


Figure 7: Observations from a vehicle with faulty spark plugs fell outside the polygons representing the safe operating regimes.

indeed provide a better alternative to a convex hull-based approach to define clusters. We are also exploring density based clustering techniques to create an even more efficient and accurate description of the clusters which includes both the shape and density parameters.

We used two data sets collected from vehicles with known health problems in our experiments. The first data set (Figure 6) is from a vehicle with a dirty air filter. The second data set (Figure 7) is generated after leaving one of the engine spark plug caps open. As we see in both figures, the faults generate data points which fall outside the boundaries of the polygons defining the normal operating regimes and thus recognized as outliers by the system. The following section discusses the problem of detecting unusual driving patterns.

4 Detecting Unusual Driving Patterns

One of the main objectives of VEDAS is to characterize the driving characteristics and to detect the unusual patterns in order to aid the driver and the fleet managers quickly detect problems. For example, drowsy driving can be life threatening and the drivers themselves may be highly interested in getting a response from the on-board module in case it detects patterns of drowsy driving. Moreover, managers of hazardous material transportation fleets may be interested in quickly detecting drunk driving. VEDAS has a module to detect anomalous driving and vehicle behavior patterns and this section presents a brief overview of that module.

Deng et al.[4] divide time series recognition approaches into two categories: parametric and non-parametric. Parametric methods assume that there is an underlying model that generates the time series. The detection of the unusual event in the time series is thus equivalent to the classification of the underlying models. Recently, Hidden Markov Model (HMM) [13], Auto Regression Moving Average [4] and neural network based methods have been used for solving this problem. Instead of using models, non-parametric methods extract the patterns from the normal series such as the mean, variance, frequencies, peaks and valleys, and try to find the interesting patterns from the new series. Several other techniques using ideas from negative-selection mechanism of the immune system, one-class Support Vector Machines and Self-Organizing Maps have been proposed elsewhere.

VEDAS offers a combination of both parametric and non-parametric techniques to monitor the data streams. For example, it deploys auto-regressive models and other statistical models. However, in this paper, we mainly discuss the non-parametric techniques used in VEDAS. We present a relatively simple but effective approach used in the current version of VEDAS for finding abnormal patterns from the OBD-II data based on hypothesis testing and Fourier spectral analysis. Since linear segmentation requires linear computation time and fast Fourier transformation requires $O(nlog(n))$ time, they are suitable for implementation on a platform with limited resources.

4.1 Background In order to illustrate the unusual driving detection capabilities of VEDAS, we use simulated drunk driving data generated by a driving simulator - Live For Speed (LFS)[2]. A data replay analyzer F1PerfView [3] is used to display detailed customizable driving data from one or more RAF files in conjunction with a 2D top-down view of the path on the track. The simulated data set was generated by noting some of the following known characteristics of drunk driving: (1) Braking erratically; (2) Weaving from one side of

[2]http://lfs.racesimcentral.com

[3]http://www.xs4all.nl/~rsdi/f1perfview.html

the road to the other; (3) Taking extremely wide turns; (4) Driving at very slow speeds - at least 10 mph below the limit and without headlights on; (5) Stopping inappropriately in places such as green lights and crosswalks with no pedestrians, etc.

The first three signs can be detected from vehicle speed and steering wheel angle data. In the following section we use vehicle speed and steering wheel angle as the primary means for modeling driver behavior for our preliminary analysis. A sample data plot of the steering behavior for a sober and a simulated drunk driver is shown in Figure 10.

4.2 Monitoring Techniques In this section, we discuss some of the non-parametric techniques adopted in VEDAS for unusual driving pattern detection in simulated drunk driving data and offer experimental results.

Speed and acceleration are two important characteristics of driving behavior. Frequent sharp accelerations and decelerations are usually viewed as unsafe driving practices. VEDAS extracts typical local acceleration patterns from the continuous driving data and estimates the underlying distribution. Distribution statistics (e.g. mean, variance, range, median, KL distance) are used to classify the quality of driving. A threshold can be set and if the new sample statistics is significantly higher than the corresponding threshold, an alarm flag is raised.

In order to efficiently extract the typical local acceleration from continuous speed stream, we explored the bottom up Piecewise Linear Segmentation method [10]. Intuitively, Piecewise Linear Segmentation refers to the approximation of a time series T of length n, with k straight lines, as shown in Figure 8. Because $k << n$, this method makes the transmission and computation of the data more efficient. Moreover, Piecewise Linear Segmentation "smoothens" the data streams and usually retains the typical features. The computational complexity of bottom up linear segmentation algorithm [10] is $O(Ln)$ where n is the number of data points in the stream and $L = n/k$ is the average segment length. The acceleration can thus be computed easily from the slope of the segment in constant time.

Furthermore, we can use statistical tests to make quantitative decisions about the difference between the means or variances of two sample distributions. For example, the F-hypothesis test can be used to compare the equality of the variances from two samples. This test assumes that the samples follow the Gaussian distribution. Note that for the same driver on the same road the acceleration pattern is usually stable. According to our experiments this appears to approximately follow the

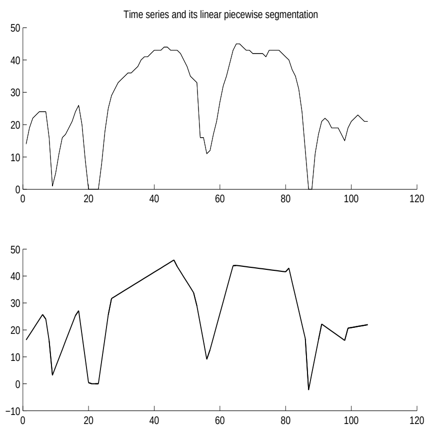


Figure 8: Piecewise linear segmentation. (Top) Original time series of vehicle speed with 108 points. (Bottom) Corresponding linear segmentation with only 18 segments (36 points, 33% of original series).

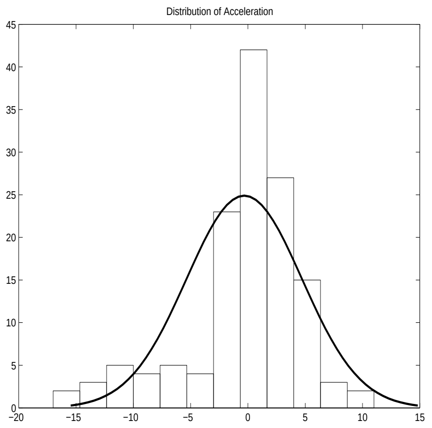


Figure 9: Distribution of acceleration.

Gaussian distribution. Figure 9 illustrates this based on the actual driving data. Let $x_1, x_2, \ldots, x_{n_1}$ be a sample from an unknown driving distribution $N(\mu_1, \sigma_1^2)$ with sample variance s_1^2, and let $y_1, y_2, \ldots, y_{n_2}$ be a sample from a safe driving distribution $N(\mu_2, \sigma_2^2)$ with sample variance s_2^2. $\mu_1, \mu_2, \sigma_1^2, \sigma_2^2$ are unknown.

The F-test is defined as:

$$\begin{aligned} H_0 : \sigma_1^2 &= \sigma_2^2 \\ H_a : \sigma_1^2 &> \sigma_2^2 \end{aligned}$$

The hypothesis that the two variances are equal is rejected if

$$\frac{s_1^2}{s_2^2} \geq F_\alpha(n_1 - 1, n_2 - 1)$$

i.e., the variance of new acceleration behavior is significantly higher than that of the normal behavior. In the real world the acceleration may not always approximate to Gaussian distribution due to different reasons. We can use several other non-parametric test methods such as permutation tests [6] as an alternative. Next, we use Mann-Whitney U-Test to compare the central tendency of the two behaviors. This is a distribution-free test which uses the rank sums of two samples. The null hypothesis is that the populations have the same mean and the alternative hypothesis is that the means are not the same. The test first ranks all $n_1 + n_2$ observations in ascending order and then it calculates the sum of the ranks from each sample, denoted by T_a and T_b. The U statistic is calculated as follows:

$$U_a = n_1 n_2 + 0.5 n_1(n_1 + 1) - T_a$$
$$U_b = n_1 n_2 + 0.5 n_2(n_2 + 1) - T_b$$

U is selected for the smaller sample size (smallest n). If the sample size is the same, U is the larger of U_a and U_b, i.e., $U = max(U_a, U_b)$. If the U value is greater than the corresponding critical value then there is a significant difference and the null hypothesis is rejected. VEDAS makes use of several distributed algorithms for comparing observed data sequences with typical normal driving sequences stored at the control station. Some of these algorithms are proprietary in nature and they are not discussed here.

Another important sign of a drunk driver is weaving from one side of the road to the other and vice versa. Therefore, we expect that the periodic structure in the frequency domain of a drunk driver would be more significant than that of a sober driver. Figure 10 shows the difference between the driving performances of a sober driver and a drunk driver. This module monitors Fourier transformation of the steering wheel angle data. This helps us classifying the data stream based on the periodic components and dominant frequencies. Figure 11 illustrates the power spectrum at different frequencies for both sober and simulated drunk driving. Note that this module of VEDAS is primarily designed for unusual driving behavior detection, not just for detecting the drunk driving behavior.

In order to detect the unusual steering pattern, we first build a sample space for a normal driver's steering wheel behavior. When a new window from the driving data stream arrives, we perform Fourier transformation on the data. Then we compare the frequency range of the two power spectrums and compute the distance of first several dominant Fourier coefficients between the normal and the unknown behavior. The alarm flag is raised when the difference is high. Currently, we are also exploring the problem using system identification idea like ARMAX (Auto Regression Moving Average with Exogenous inputs) and Box-Jenkins structures [3]

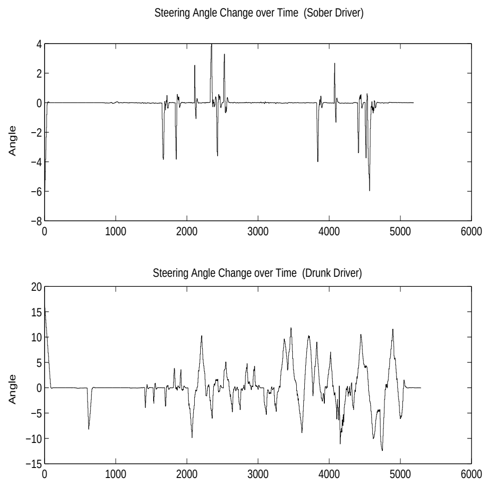


Figure 10: Steering wheel angles for sober and drunk drivers.

which can define driver's behavior through a parametric model. The following section discusses the privacy management module of the VEDAS system.

5 Protecting the Privacy of the Driver

Privacy management played an important role in the design of VEDAS. Like most useful technologies mobile and distributed data mining can be used for achieving good things. On the other hand, it can also be potentially used to intrude the privacy of the driver for undesirable practices. Therefore, it is important that we take measures to protect the privacy of the good drivers. This section considers some of the distributed statistics computation problems in VEDAS that may require comparing on-board data with typical behavior data stored at the control station (because of limited on-board data storage space) and explains some of the solutions that VEDAS adopted to protect driver's privacy. Although the techniques are experimental in nature and the field of privacy preserving data mining [1, 12, 18] still lacks general purpose techniques to guarantee privacy protection against every attack types, we believe that the techniques offer better protection compared to no protection at all. These specific techniques can also be replaced as the technology matures.

Consider the problem of computing the distance between two sequences—one is comprised of the currently observed on-board data and the other defines the usual driving behavior. The distance can be the inner product or the Euclidean distance. Since Euclidean distance computation problem can be easily reduced to the inner product computation problem, in the rest of this section we consider just the inner product computing problem. So our objective is to compute the inner product ma-

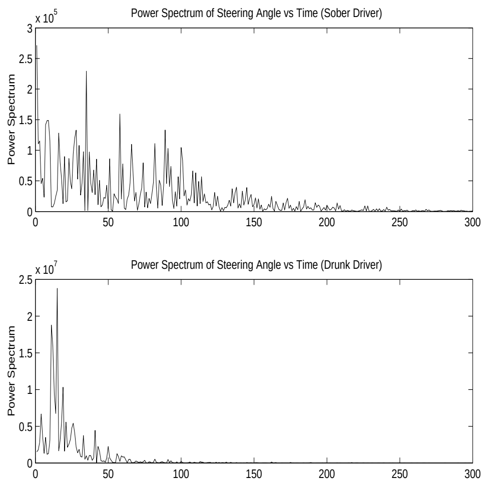


Figure 11: Fast Fourier transform of steering wheel angle data. (Top) Power spectrum of sober driver's steering wheel angle. (Bottom) Power spectrum of a drunk driver's steering wheel angle.

trix (in the general case involving more than two data vectors) from multiple data locations without necessarily divulging the the on-board data to the management operating the control station.

The approach we adopt works using a sequence of randomized linear transformations [12] of the data. Rest of this section offers a brief overview of this approach.

LEMMA 5.1. *Let R be a $p \times q$ dimensional random matrix such that each entry $r_{i,j}$ of R is independently chosen according to some unknown distribution with mean 0 and variance 1. Then,*

$$E[RR^T] = qI$$

Intuitively, this result echoes the observation made elsewhere [15] that in a high-dimensional space vectors with random directions are almost orthogonal. A similar result was proved elsewhere [2].

Lemma 5.1 can be used to prove the following result.

LEMMA 5.2. *Let u, v be row vectors with m real entries, i.e. $u,v \in \Re^m$. Let R be an $m \times k_1$ dimensional random matrix such that each entry $r_{i,j}$ of R is identically, independently chosen according to some unknown distribution with mean 0 and variance 1. Let $u_1 = uR$ and $v_1 = vR$, then*

$$E[u_1 v_1^T] = k_1 uv^T$$

Note that uv^T is nothing but their inner product. The above result shows that if the owners of the u and v project their data to a random space to get u_1 and v_1 respectively and then hand these over to a third

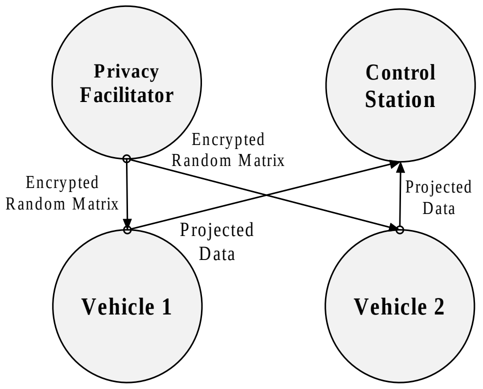


Figure 12: Privacy preserving distributed inner product computation schematic.

party, inner product can still be computed by the third party. However, given u_1 or v_1 one may not be able to accurately determine the original data u or v, which is based on the premise that the possible solutions are infinite when the number of equations is less than the number of unknowns. A more detailed analysis of the privacy protection can be found elsewhere [12].

In our system, a trusted privacy facilitator agent sends out encrypted privacy schema, i.e., the random matrix to local vehicles. On receipt of this random matrix, the local vehicle projects its data onto the random matrix and sends the projected data to the control station, where the inner product matrix can be computed without knowing the true value of the original data. Figure 12 illustrates the schematic.

6 Experimental Results

The last decade has seen a substantial improvement of wireless communication and computing technology. However, battery power and bandwidth are still two of the most scarce resources in most wireless applications and they are likely to remain so in the near future. This section presents some experimental results that document the performance of some of the VEDAS modules on those two grounds.

6.1 Communication Mining distributed data sources connected through wireless networks using a traditional centralized data mining system would require downloading all the data to a central location before any analysis. The main problem with this approach is the lack of sufficient bandwidth and high power consumption rate needed for data transmission. Thus reducing data transmission was an important consideration in the design of our system. Since most of the computation is performed on-board, VEDAS incur very little data transmission overhead, unless

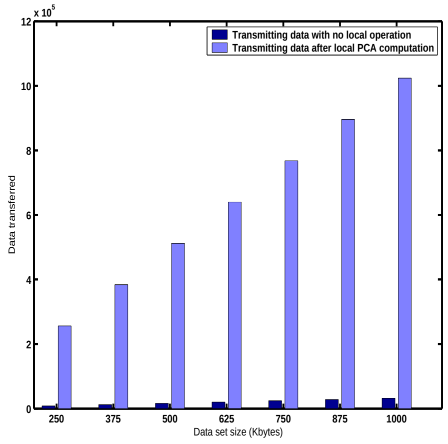


Figure 13: Communication bandwidth usage comparison for transmission of data with no on-board computation and for transmission of data reduced to two dimensions using on-board PCA computation.

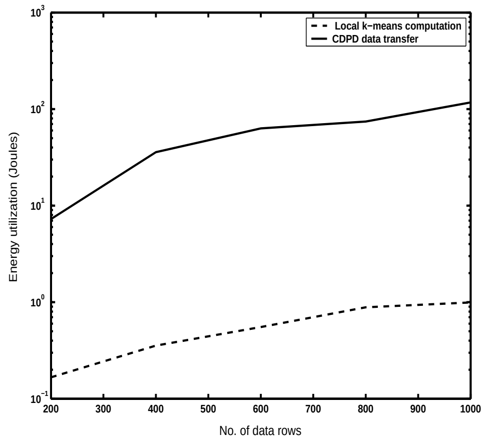


Figure 14: Comparison of average energy usage for on-board k-means clustering (std. deviation = 0.0265) and CDPD transmission of data with no on-board computation.

the control station explicitly seek downloading the data for specific investigative reasons. Even then VEDAS applies various techniques to reduce the data transmission. For example, performing PCA reduces the dimensionality and thus the size of the data set. In all of our experiments only a handful of eigenvectors were sufficient to capture most of the variance in the data. Figure 13 shows the savings in communication because of the PCA-based compression. VEDAS also deploys various other distributed approximation low-communication-algorithms that are not discussed here.

6.2 Power Consumption Most of the wireless devices we have considered in this paper are battery operated and lack a constant source of power. In commercial fleets one may be able to provide alternate power supply. However, in the version of VEDAS designed for personal stand-alone on-board use, we consider the scenario where the driver is not willing to set up additional power supply hardware on-board. In this case, the power consumption characteristics of the data stream mining algorithms become an important issue. Moreover, most other distributed data stream mining applications (e.g. sensor networks for defense applications) heavily rely on battery powered sensors. Therefore, we are better off designing algorithms and systems that minimize the energy consumption [17]. This section presents the experimental results for the energy consumption profile of some of the data mining techniques used by VEDAS.

The energy E needed to execute a sequence of operations is given by the equation $P(t) = V(t) \times I_c$, where $V(t)$ is the supply voltage, I_c is the current and P is the power and energy can be calculated as $E = \int_{t_1}^{t_2} P(t)dt$. The total energy consumption of a system depends on both the hardware and software components and the communication sub-system. For the current application, let us define the total energy consumption to be, $E = E_c + E_t$, where E_c and E_t are the energy consumptions for computation and communication of the data respectively.

We built an experimental setup for characterizing the energy consumption profile of the VEDAS modules. It runs on an HP Jornada 690 with CDPD network connection. Energy consumption was determined by measuring the input voltage and the current across a 1Ω resistor using Agilent 54622A oscilloscope and 12V DC power. Figures 14 and 15 present the energy consumption characteristics of the some of the modules. The reported data were computed after the energy needed for the GUI and WinCE background tasks was subtracted from the observed energy measurements for the VEDAS modules. They show that in general, from the energy consumption perspective on-board computation works a lot better than computing at remote desktop environment after transmitting the data through the wireless network. These results strengthen the case for building on-board data mining applications for mobile devices.

7 Conclusions and Future Work

Although mobile and wireless computing devices are becoming very popular, most of the current applications are still in an early stage from the data analytics perspective. Limited computing resources, restricted bandwidth, and other constraints on the human-computer interactions offer several challenges toward building advanced data analytics-based applications for ubiquitous distributed environments. However, we believe that we

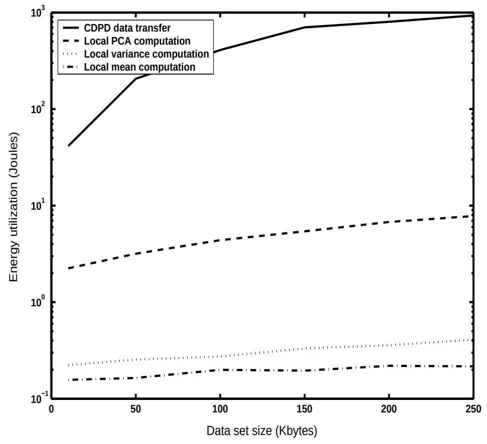


Figure 15: Comparison of energy usage for on-board computation of mean, variance and PCA and for CDPD transmission with no on-board computation. The standard deviations for mean, variance, PCA and CDPD are 0.0877, 0.1281, 0.4128 and 71.67 respectively.

can develop a new generation of distributed data mining algorithms and architectures that meet the challenges posed by such environments. This paper presented VEDAS, a real-time on-board data stream mining system used for vehicle-health-monitoring and driver status-characterization. The overall objective of the VEDAS system is to help the drivers by characterizing their status and help the fleet managers by quickly detecting security threats and problems vehicle-health. The paper considered several specific challenges in designing algorithm and the system—minimizing bandwidth, power consumption, privacy management, and making on-board computation very efficient. We are currently in the process of incorporating many additional on-board data stream mining techniques and fleet level distributed data mining capabilities.

Acknowledgments

The authors would like to acknowledge support from TEDCO, the United States National Science Foundation CAREER Award IIS-0093353, US Air Force Grant F33615-03-M-4120 for performing this research.

References

[1] R. Agrawal and S. Ramakrishnan. Privacy-preserving data mining. In *Proceedings of SIGMOD Conference*, pages 439–450, 2000.

[2] R. Arriaga and S. Vempala. An algorithmic theory of learning: Robust concepts and random projection. In *Proc. of the 40th Foundations of Computer Science*, New York, New York, 1999.

[3] Peter J. Brockwell and Richard A. Davis. *Introduction to Time Series and Forecasting.* Springer texts in statistics. Springer, New York, 2002.

[4] K. Deng, A. Moore, and M. Nechyba. Learning to Recognize Time Series: Combining ARMA models with Memory-based Learning. In *IEEE International Symposium on Computational Intelligence in Robotics and Automation*, volume 1, pages 246–250, 1997.

[5] G.H. Golub and Van Loan C.F. *Matrix Computations.* John Hopkins University Press, Baltimore, 2 edition, 1989.

[6] Phillip Good. *Permutation Tests A Practical Guide to Resampling Methods for Testing Hypotheses.* Springer-Verlag, New York, 1993.

[7] H. Hotelling. Analysis of a Complex of Statistical Variables into Principal Components. *Journal of Educational Psychology*, 24, 1933.

[8] A. Jorge. Adaptive Tools for the Elderly: New Devices to Cope with Age-induced Cognitive Disabilities. In *Proceedings of the 2001 EC/NSF workshop on Universal accessibility of ubiquitous computing: providing for the elderly*, pages 66–70, 2001.

[9] H. Kargupta, B.H. Park, S. Pittie, L. Liu, D. Kushraj, and K. Sarkar. MobiMine: Monitoring the Stock Market from a PDA. *ACM SIGKDD Explorations*, 3(2):37–46, January 2002.

[10] Eamonn J. Keogh, Selina Chu, David Hart, and Michael J. Pazzani. An Online Algorithm for Segmenting Time Series. In *Proceedings of IEEE International Conference on Data Mining*, pages 289–296, 2001.

[11] Ickjai Lee and Vladimir Estivill-Castro. Polygonization of Point Clusters through Cluster Boundary Extraction for Geographical Data Mining. In Springer-Verlag D. Richardson and P. v. Oostrom, editors, *10th International Symposium on Spatial Data Handling (SDH)*, pages 27–40, Ottawa, Canada, July 2002.

[12] K. Liu, H. Kargupta, and J. Ryan. Random projection and privacy preserving correlation computation from distributed data. Technical Report TR-CS-03-24, Computer Science and Electrical Engineering Department, University of Maryland, Baltimore County, 2003. In communication.

[13] N. Oliver and A. P. Pentland. Graphical Models for Driver Behavior Recognition in a SmartCar. In *Proceedings of IEEE International Conference on Intelligent Vehicles*, Detroit. Michigan, October 2000.

[14] B. H. Park and H. Kargupta. Distributed Data Mining: Algorithms, Systems, and Applications. In Nong Ye, editor, *The Handbook of Data Mining.* Lawrence Erlbaum Associates, 2002.

[15] R.Hecht-Nielsen. Context vectors: general purpose approximate meaning representations self-organized from raw data. *Computational Intelligence: Imitating Life*, pages 43–56, 1994.

[16] G.W. Stewart and J.G. Sun. *Matrix Perturbation Theory.* Academic Press, New York, 1990.

[17] V. Tiwari, S. Malik, A. Wolfe, and M.T.-C. Lee. Instruction level power analysis and optimization of

software. *Journal of VLSI Signal Processing Systems*, 13, August 1996.

[18] J. S. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *In The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Edmonton, Canada, July 2002.