REDUCTION

2011-2014

Deliverable 2.3.1

Progress report on advanced predictive analytics models

July 30, 2014

Project acronym: Reduction

Project full title: Reducing Environmental Footprint based on Multi-Modal Fleet management Systems for Eco-Routing and Driver Behavior Adaptation

Work Package: 2

Document title: Progress report on advanced predictive analytics models

Version: 1.0

Official delivery date: 31/08/2013

Actual publication date: 31/08/2013

Type of document: Deliverable Report

Nature: Public

Authors: Josif Grabocka, Umer Khan

Approved by: Lars Schmidt-Thieme

Reviewed by: Bin Yang

Version	Date	Sections Affected
0.1	01/07/2013	Initial version
0.9	30/07/2013	Reviewed by UHI
1.1	31/01/2014	Incorporated Comments of the Technical Review Report

Contents

List of Tables

List of Figures

Executive Summary

The REDUCTION project is a EC-FP7 STREP project which addresses the reduction of fuel consumption through exploring advances in Intelligent Transportation Systems. The approach followed by the consortium consists of three domains, ecological routing, ecological driving and distributed data mining, aided by advances in V2I and V2V communication. A consortium of nine partners from five countries merge their efforts in seven work packages. The first work package (WP1) develops efficient V2I and V2V communication algorithms lead by UTH and state-of-art devices by DDE. Furthermore the second work package (WP2) extends the state-of-art in ecological driving (eco-driving) and distributed data mining, in V2V scenarios, through the contribution of UHI. Ecological routing is included in the third work package (WP3) of the project and is steered by AU and AAU. The integration of the aforementioned contributions is conducted by TRI in the fourth work package (WP4). In the fifth work package (WP5), methodological advances of WP1-3 will be applied on the Trainose, CTL, FlexDanmark and Trinite field-studies.

This deliverable is part of WP2 and includes the progress achieved in the period 01-Sept-2012 to 31-Aug-2013, incorporating novel methods in the domains of eco-driving and distributed data mining. The scope of the reported work lies within the directives of the Task 2.3 of the DOW document.

A novel methodology is proposed in the light of eco-driving and is covered in Chapters 5 and 6. The innovative method re-conceptualizes the assessment of driving behaviors from a data mining perspective. Analysis of velocity plots computed from the GPS recordings of vehicles leads to the discovery of local driving patterns. Those local driving behaviors are represented as symbolic polynomials and are stored in histograms together with the frequency of occurrence. The fuel consumption estimation is used to select the most influential local driving patterns with respect to fuel efficiency. Our experimental analysis over real-life data concluded that sudden accelerations and decelerations represent the most influential local driving behaviors. In addition, this deliverable covers the state-of-art advancements by REDUCTION in the domain of distributed data mining. Advancements of distributed algorithms, suited for V2V topologies, are detailed in Chapter 3. A novel paradigm is followed in sharing intelligence among neighboring vehicles, aiming at broadcasting locally learned statistical models rather than raw data. This can significantly reduce the overall cost of the communication.

Chapter 1

Scope of Work

1.1 Project Description

The reduction of CO2 emissions is a great challenge for the transport sector nowadays. Despite recent progress in vehicle manufacturing and fuel technology, still a significant fraction of CO2 emissions in EU cities is resulting from road transport. Therefore, additional innovative technologies are needed to address the challenge of reducing emissions. The REDUCTION project focuses on advanced ICT solutions for managing multi-modal fleets and reducing their environmental footprint. REDUCTION collects historic and real-time data about driving behaviour, routing information, and emissions measurements, that are processed by advanced predictive analytics to enable fleets enhance their current services as follows:

1. Optimising driving behaviour: supporting effective decision making for the enhancement of drivers education and the formation of effective policies about optimal traffic operations (speeding, braking, etc.), based on the analytical results of the data that associate driving-behaviour patterns with CO2 emissions;
2. Eco-routing: suggesting environmental-friendly routes and allowing multi-modal fleets to reduce their overall mileage automatically; and
3. Support for multi-modality: offering a transparent way to support multiple transport modes and enabling co-modality.

REDUCTION follows an interdisciplinary approach and brings together expertise from several communities. Its innovative, decentralised architecture allows scalability to large fleets by combining both V2V and V2I approaches. Its planned commercial exploitation, based on its proposed cutting edge technology, aims at providing a major breakthrough in the fast growing market of services for ”green” fleets in EU and worldwide, and present substantial impact to the challenging environmental goals of EU.

1.2 Objectives of Work Package 2 (WP2)

REDUCTION's division of technical work is composed of V2I/V2V communication (Work Package 1), ecological routing (Work Package 3), ecological driving (Work Package 2) and distributed data mining (Work Package 2). In addition, REDUCTION includes four field-studies (Work Package 5), where the state-of-art advances are applied to real-world scenarios.

The primary scope of the second work package is to develop methods which involve statistical prediction in the realm of ecological ICT solutions. More specifically, the contribution is targeted at three specific problems. First of all, eco-driving refers to the adaptation of driving behaviors and is addressed by analyzing large amounts of driving behavior data. The detection of inefficient behaviors will be conducted by calibrating the predictive models with the help of fuel consumption estimations. Distributed data mining is the next problem addressed by WP2 and refers to the elaborations of statistical models which operate in a decentralized mode. Such topologies are inherent to the nature of inter-vehicular (V2V) communication, where centralized decision-making is not feasible. REDUCTION aims at developing distributed statistical models per vehicle, which share intelligence within a neighborhood. The final problem addressed in the scope of WP2 is the prediction of trajectories, which refers to the prediction of the motion of vehicles in a close time frame.

WP2 is organized in four tasks, namely:

- **Task 2.1 -** Requirement specification and Software architecture (TRI); Duration: Month 3 to 12; Scope: Collection of Requirements
- **Task 2.2 -** Basic prediction models (UHI); Duration: Months 7 to 12; Scope: Eco-Driving and Distributed Data Mining
- **Task 2.3 -** Advanced prediction models (UHI); Duration: Months 12 to 36; Scope: Eco-Driving and Distributed Data Mining
- **Task 2.4 -** Vehicle Motion Prediction Algorithms (UTH); Duration: Months 18 to 24; Scope: Motion/Trajectory Prediction

1.3 Objectives of Deliverable 2.3.1

This deliverable, D2.3.1 lies within the context of Task 2.3 which includes advancements in the domains of Eco-Driving and Distributed Data Mining. The time frame of the deliverable are months 12 to 24 that corresponds with the duration of project's second year. The objectives targeted by this deliverable can be summarized as follows:

- Distributed Data Mining (covered by Chapter 3)
  - Development of decentralized algorithms for V2V/V2I decision-making
  - Development of efficient local predictive models which learn from intra-vehicular data
  - Development of broadcasting algorithms for sharing local predictive models with neighboring peers/vehicles

  - Experiments to validate the gained benefits in terms of improved prediction accuracy and reduced communication time

- Eco-Driving (covered by Chapters 5 and 6)
  - Processing of GPS trajectory data into driving behaviors
  - Analysis of driving behavior data and detection of driving patterns
  - Identification of fuel-inefficient driving patterns
  - Empirical evidences over real-life GPS data

The objectives of this deliverable (D2.3.1) and the summary of the progress achieved along each objective are summarized in Table 1.1.

Table 1.1: Summary of the Objectives of Deliverable 2.3.1

No	Description	Progress Summary
1	Local Predictive Models for Intra-Vehicular Networks	Developed decentralized V2V/V2I statistical models for efficient local prediction
2	Eco-Driving from GPS data	Processing of GPS trajectories in Form of Velocity Plots; Estimating Fuel Consumption of each Velocity Plot using the VT-Micro Model
3	Detection of Local Driving Patterns	Computing Frequencies of Local Polynomials from Velocity Plots; Extracting Fuel-Inefficient Behaviors
4	Empirical Evaluation of Methods	Conducted Experiments on The Infati Dataset for mining Eco-Friendly Driving Patterns

Chapter 2

Introduction

Modern vehicles are embedded with a variety of sensors monitoring different functional components of the vehicle and driver's behavior. Large number of vehicles connected to wide-area wireless networks create a ubiquitous environment of distributed sources of voluminous data and mobile dynamic nodes. Such compute nodes, leveraged by wireless network devices, can provide access to vehicle diagnostic data along with location and accelerometer information. These data offer a rich source of information about the vehicle and driver performance [44]. Combined with other contextual data such as road environment, location and driver behavior, interesting fuel-friendly applications can be conceived. Among the most substantial applications are cooperative lane changing via distributed learning and classification of driver behavior in terms of Eco-Driving.

Optimizing traffic flow using cooperative driving requires a distributed learning and information exchange method. The distributed models (lane change, traffic density, congestion) are learned locally on each node and then are exchanged with neighboring vehicles. On the other hand, Eco-Driving is a way of maneuvering a vehicle with a human driver that is intended to minimize fuel consumption, while coping with varying and uncertain road traffic by employing the most efficient driving strategies whenever necessary. The series of driving related actions/parameters history/records can be accessed through the GPS measurements of a driver's historical trips, and be represented as velocity plots. Therefore, analyzing driving patterns can be achieved by time-series classification methods.

2.1 Our Methodological Contribution

In the first part of this deliverable, we present our methods and corresponding experimental evaluations related to distributed data mining models in ad-hoc networks. In addition, their applications to cooperative learning scenarios are further elaborated. Learning patterns from large-scale distributed networks, such as Vehicular Ad-Hoc Networks (VANET), is a challenging task because the centralization of data is not feasible. The centralized approach of data mining works by regularly uploading mission critical data into a warehouse for subsequent centralized analysis. Such a centralized

approach does not perform optimally in many of the emerging distributed, ubiquitous, possibly privacy-sensitive data mining applications. The communication cost, long response time, lack of proper use of distributed resources and router-less ad-hoc environments with a highly dynamic topology of inter-connecting vehicles make centralization of data impractical. Being motivated by this problem, we have developed a distributed data mining approach which produces *light weight* predictive models locally on each ad-hoc computing node, and then exchanges these models with the neighboring vehicles in a communication efficient way. For collaborative knowledge sharing, our method performs transitive propagation of local classification models and helps scarcely connected peers to receive knowledge that improves the accuracy of their local classifiers. This propagation follows a transitive reduction process, which keeps reducing the size of model, by only forwarding significant knowledge. To evaluate our approach, we developed a simulation and our experimental results show that our approach achieves a performance comparable to in-feasible centralized approaches, while keeping the communication cost low. Chapter 3 describes our proposed distributed learning approach, while Chapter 4 explains cooperative lane changing scenarios in VANETS, as an application of distributed data mining for fuel consumption.

Second part of deliverable, describes our time-series classification methods for analyzing drivers' behaviors. As mentioned earlier, using GPS measurements, parameters and historical records can be extracted as a set of time-series. For instance, parameters can be: velocity of a car measured at a particular frequency, acceleration, throttle or fuel consumption. In this deliverable the focus lies on mining velocity plots. Behavior messages exhibit a time series continuity, therefore we can employ time-series methodologies in order to classify the time series, detect anomalies or regularities in the records. This can ultimately help in classifying a driving pattern as eco-friendly or non eco-friendly. In Chapter 5 we have proposed and evaluated a novel method to classify long time series composed of local patterns occurring in an unordered fashion and by varying frequencies. Our principle relies on detecting local polynomial patterns which are extracted in a sliding window approach, hence fitting one polynomial to each sliding window segment. We refer to this approach as *Bag-of-Patterns Representation of Time-Series*. In Chapter 6, we describe the application our time-series methods to determine fuel efficient driving patterns, where we extract driving behavior patterns from velocity plots (driving behavior time series) of the GPS recordings. The mining of local patterns in the velocity plot is conducted using the bag of patterns algorithm. In order to detect which of the extracted local patterns is more influential with respect to fuel consumption, the VT-Micro estimation model is utilized for fuel estimation based on instantaneous velocity and acceleration. We performed our experiments on Infati dataset and selected velocity plots from the most frequently traveled map segment. Our empirical analysis showed that top influential local patterns with respect to describing fuel consumption were the sudden accelerations and decelerations.

Chapter 3

Advanced Models on Distributed Data Mining

3.1 Introduction

In recent years, there is an increasing interest for analytical methods that perform data mining over large-scale data distributed over networks, for real world problems like parallel and distributed learning in wide spread computer networks and ad-hoc networks like P2P and WANET (MANET, VANETS etc). Distributed learning is a challenging task, because centralization of data is not feasible for reasons like limited and unreliable communication resources, resources of data and computations are sparsely distributed and data collections are evolving dynamically across the network. Therefore, the goal is to develop distributed mining algorithms that are communication efficient, scalable, asynchronous, and robust to peer dynamism, which achieve accuracy as close as possible to centralized but unfeasible ones.

Existing research in distributed learning in networks focuses on developing local classification or clustering algorithms which use primitive operations, such as distributed averaging, majority voting and other aggregate functions to combine results of local classifiers in order to establish a consensus of classification decisions among nodes in the network [58]. [6] proposed a distributed decision tree induction algorithm in a P2P network. Most of these locally synchronized algorithms are reactive in a sense that they tend to monitor every change in data and keep track of data statistics. Another important work is distributed learning through model propagation. [3] and [63] proposed Support Vector Machines based communication-efficient classifier by propagating models among network peers.

An important limitation of the aforementioned approaches is that they do not consider P2P networks with *'scale free'* topology where the number of connections among peers and the amount of data stored in peers are distributed in a skewed (i.e., non uniform) fashion. As a result, there are usually few peers with many connections and large amount of data, whereas there are many peers with few connections and small amounts of data. For example, two real world P2P networks, Gnutella and BitTorrent, express skewed topology based on power laws [70], [19]. By ignoring the scale free topology

and the resulting skewness, existing methods are susceptible to waste communication cost without resulting in effective classification models, especially for the large number of peers with few connections and small amounts of data.

In this work, we have developed a distributed classification approach which takes into account the scale free nature of ad-hoc networks, especially P2P networks. This approach is based on the idea that motivates the propagation of classification model beyond the direct neighbors, with continuous reduction in model size at each transitive step. A local model is learned using Reduced-SVM [50], at each peer. Then this model is exchanged with the peers in the network using the idea of transitive reduction explained in Section 3.3. This way, our approach helps not well connected peers to receive local models of other, better connected peers, in order to enhance the local models of the not well connected peers, and thus to substantially improve classification accuracy over the entire P2P network. Since uncontrolled transitive propagation of classification models can increase the communication cost (leading eventually to flooding the network), this approach works in a controlled way by first using 'light weight' models generated by Reduced-SVM, and then peforming transitive reduction, that keeps the communication cost low and improves the classification accuracy at a weak peer, which represents the vast majority of peers in real world P2P networks. We would like to emphasize out that the vehicles share statistical models, not merely information. In addition, the algorithms described in this chapter applies to both the urban and inter-urban transportation.

3.2 Related Work

Distributed data mining deals with learning in resource constrained environments with distributed data, computing nodes and users. A very good introduction to this area is provided by [45] and [64].

Parallel and distributed classification based on ensemble classifiers, such as Bagging[7] and Boosting [25], is examined by [68]. Ensemble approaches can be further categorized in to meta-learning and voting, based on the method used to combine models. In voting, final decision is made by a plurality of votes of base-classifiers. For naturally distributed data, voting techniques proposed by [78] and [49] are important to mention. On the other hand, meta-learning is used to learn a combining method based on the meta-level attributes obtained from predictions of base classifiers [13].

Recently, in [51] authors proposed a method to learn an SVM classifier in a distributed scenario. This method considers vertical partitioning of data. Optimization is performed independently on local features of each partition using multiple local kernels and approximate feature mappings. A central coordination is required to adjust the weights of local kernels for improved classification. Most of these algorithms assume stable network and data, and require centralized control. That is why, the quality of such algorithms degrades as the scale of the network grows to millions of nodes.

Current *state-of-the-art* methods for distributed learning in P2P networks focus on developing local classification or clustering algorithms which in-turn make use of primitive operations such as distributed averaging, majority voting and other aggregate functions. Most representative work in this regard is distributed association rule mining [82], distributed decision tree induction[6] , distributed K-Means clustering [20] and

distributed classification[58]. A common aspect of these approaches is that they make use of *local algorithms*, which compute their result using information from just a handful of nearby neighbors, and hence are independent of the network size. A more well known local algorithm is Majority Voting by [82] and [81]. In this algorithm, each peer maintains its local belief of an estimate of global sum based on the number of nodes in that network, to confirm whether a majority threshold is met. If not then this peer needs to send messages to its neighbors to re-estimate the global sum. All the above mentioned representative works in P2P data mining, are based on such consensus by majority voting. Such approaches are reactive in the sense that they tend to monitor every single change in data in their neighborhood, which also require extra coordination communication.

Another approach to learn in a distributed manner is by model propagation. A classifier model is learned locally at each peer, and is exchanged with the neighboring peers. In [3] and [63], proposed a communication-efficient SVM cascade approach. This approach builds an SVM for each peer's local data, then iteratively propagate and merge the models to create an improved SVM until all subsets have been combined. In terms of prediction accuracy, their approach performed comparable to a centralized classifier and yet provide an upper bound to communication cost.

A major limitation of these methods is that they consider uniform and controlled placement of peers in a structured P2P network. However, in most real world P2P networks, such as Gnutella and BitTorrent, the network topology is highly skewed and represented by a power law, i.e. $d_p \sim x^{-a}$, where d_p is the degree of peer p [70], [19]. In such cases, most of the peers (henceforth characterized as *'weak'*) have small degrees, whereas few peers (henceforth characterized as *'hubs'*) have large degrees. Additionally, the degree of a peer tends to correlate with the amount of data that this peer contains[72],[43]. As a consequence, weak peers usually do not have a sufficiently large local training data set, and resulting local models perform badly in terms of prediction accuracy. Still, exchange of such meaningless models incur a huge communication cost (since such peers make a large majority in the network). An example scenario in Figure 3.1 illustrates this problem. The more representative knowledge from *hubs* like 1, could not arrive at *weak peers* like 4. And the communication of m_4 from 4 to 3 is useless since m_4 is a weak model, and can not help 3 to improve its local model.

3.3 Transitive Classification in P2P Networks

The methodology proposed in this section, is submitted to INFORMS Journal of Computing, and currently in the review process.

Transitive classification is based on the idea to allow peers to propagate their local models not only to their immediate neighbors, but additionally to other nodes that are reachable through transitive connections among peers. In this way, week peers can receive knowledge from hubs, which have adequately accurate local models. On the other hand, straightforward transitive propagation may result in flooding the P2P network and thus in an in-feasible communication cost. To overcome this problem, the size of the local models should be kept controllable by reducing it progressively at each transit, and simultaneously not harming the classification accuracy.

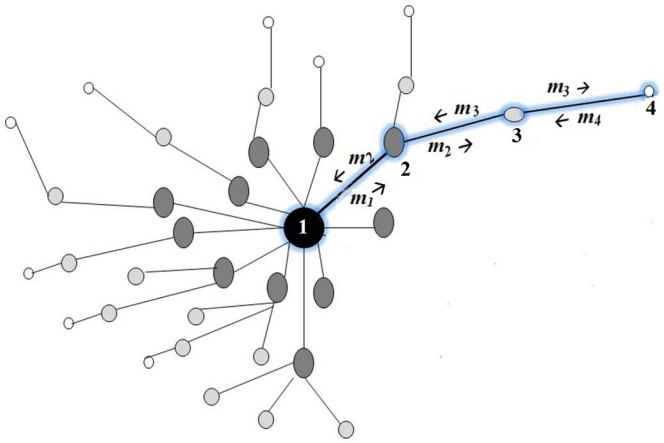

Figure 3.1: Conceptual Example of a P2P Network with Power-law Topology and Model Exchange

3.3.1 Building Local Classifiers

To build *light weight* local model at each peer, we used Reduced-SVM [50]. RSVM first randomly selects a subset r of dataset X_p at each peer and generates a thin rectangular kernel matrix $K(|X_p| \times r)$, where $|X_p|$ is the number of instances in X_p and $r \ll |X_p|$. Then, RSVM uses this K to replace the full kernel matrix $K(|X_p| \times |X_p|)$ that a conventional SVM would normally require. Due to much smaller kernel K used by it, compared to the full kernel of SVM, RSVM significantly reduces the number of support vectors, the set of which comprises the local model m_p of each peer p. Additionally, this substantially reduces the computational time and memory usage required by RSVM compared to SVM, whereas the classification accuracy of RSVM is comparable to that of SVM (especially in the case of non linear kernels) [50], [52], [53], [57].

RSVM Formulation: Given a training set of instance-label pairs $(x_i y_i),\ \forall i = \{1, ., l\}$ where $x_i \in \mathbb{R}^n$ and $y_i \in \{1, -1\}$, the support vector technique solves the following optimization problem:

$$\text{minimize} \qquad \frac{1}{2}||w||^2 + C\sum_{i=1}^{n}\xi_i \tag{3.1}$$

subject to:

$$y_i(w^T\Phi(x_i) + w_0) \geq 1 - \xi_i, \text{and } \xi_i \geq 0, \quad i = 1, ..., l$$

Since $\Phi(x)$ maps x into a higher (maybe infinite) dimensional space, practically we solve its dual, a quadratic programming problem with the number of variables equal to l:

$$\begin{aligned} &\text{minimize} && \frac{1}{2}\alpha^T(Q+\frac{I}{2C})\alpha - e^T\alpha \\ &\text{subject to:} \\ & && y^T\alpha = 0, \text{and } \alpha_i \geq 0, \quad i = 1, ..., l \end{aligned} \tag{3.2}$$

Where e is the vector of all ones, Q is an l by l positive semi-definite matrix, $Q_{ij} = y_i y_j K(x_i, x_j)$, and $K(x_i, x_j) = \Phi(x_i)^T \Phi(x_j)$ is the kernel function.

The idea of RSVM is to reduce the number of support vectors. It is achieved by randomly selecting a subset of m samples for constructing w:

$$w = \sum_{i \in R} y_i \alpha_i \Phi(x_i) \tag{3.3}$$

where R contains indices of this subset. So we solve the following general form.

$$\begin{aligned} &\text{minimize} && \frac{1}{2}(\overline{\alpha}^T\overline{\alpha} + b^2) + C\sum_{i=1}^{l}\xi_i^2 \\ &\text{subject to:} \\ & && Q_{:,R}\overline{\alpha} + by \geq e - \xi \end{aligned} \tag{3.4}$$

Where $\overline{\alpha}$ is the collection of all $\overline{\alpha}_i, i \in R$. Note that now m is the size of R. There are still l constraints.

3.3.2 Transitive Propagation and Re-Learning

After building local models at each peer m_p at each peer p in the first phase, using a predefined reduction parameter η (subset ratio to estimate m), each peer p using its neighbor list, N_p, propagates the local model m_p to all directly connected neighbors. Moreover for receiving models, each peer p waits for time t until m_q from all the $q \in N_p$ have been received. Once all the neighboring models have been received, in the second phase, each p relearns an RSVM on each of the received m_q, using η_p percent reduction. η_p is determined by *hubness level* of peer p. Each peer p is assigned a hubness level according to its degree. Hubness levels result by organizing the degrees of all peers in a small number of groups (in our experiments we used 5 hubness levels). A peer with higher hubness level will also have larger η_p, and thus performs smaller reduction. This means, that hub peers propagate relatively larger portion of knowledge received from neighbors. On the other hand, a weak peer with low hubness level has smaller η_p, performs larger reduction and propagates much smaller portion of knowledge. Figure 3.2 illustrates the concept of transitive reduction based on hubness levels. This reduction based relearning allows to save a lot of communication cost, since in most real-world P2P topologies, the majority of peers are weak. However, this approach can still attain that each peer receives a portion of local models created by hubs.

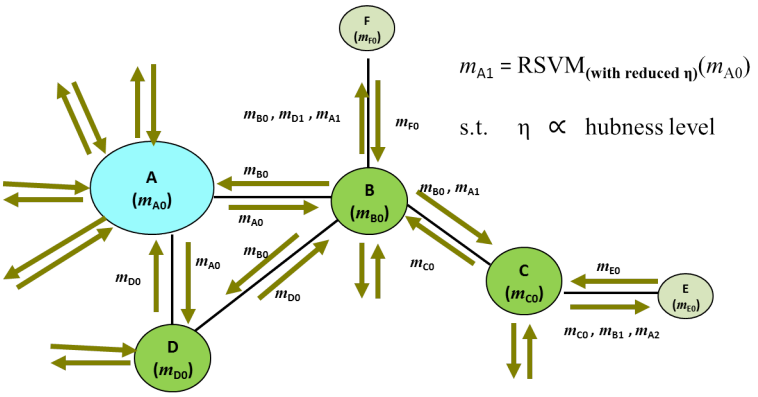

Figure 3.2: Transitive propagation with relearning with reduced η based on corresponding *hubness level*

After relearning, each of the received models with η_p, for each received model, a peer queries all of its directly connected peers with id of *sender* of this model, if they have already received any model from the same *sender*. If a particular neighbor does not have any model from this *sender*, or have one with a smaller size (defined by a threshold), then it acknowledges the querying peer to send its re-learned model.

This process of reduction based relearning and controlled propagation in the second phase continues at each peer iteratively until it has received, from all the neighbors, subsequent more stronger models which it should have obtained. Once the propagation to and receipt from all the neighbors is finished, each peer p merge the instances in all the received models to its local training data and relearn the RSVM with the network wide common value of η to generate the final model m_p. The process of updating the local models using TREDSVM can be repeated periodically by tracking the significant data changes in the network as proposed by some research efforts like [73], and [5].

3.3.3 Architectural Requirements

For transitive classification, some architectural characteristics of real world P2P networks, should be considered. P2P systems like Gnutella, Napster, KaZaA and BitTorrent, exhibit a sort of inherent hierarchy in their overlay network. This hierarchy is either reflected in the form of bootstrapping peers (Gnutella), *super peers* (KaZaA) or *trackers* (BitTorrent), which volunteer to accept additional responsibilities of coordinating and assisting a subset of clients (weak or normal peers) in network joining, neighbor assignment and query processing [83], [85]. In transitive classification algorithm, we consider such peers as hubs. Another assumption regarding P2P network is that the communication among peers is reliable, i.e. messages do not get lost on the way. It is also assumed that the P2P networks provides a reliable copy of set of neighbors N_p, for each peer at any given time. This would help a peer in recognizing if any of its neighbors has left the network.

3.4 Experiments and Evaluations

3.4.1 Network Topology, Data and Simulation

Our evaluation needs to determine the network topology with edge delays and local computations at each peer with message exchange. To generate a *power law* distribution of peers, we used *Barabasi-Albert* model ([4]) with parameters like *preferential connectivity* and *incremental growth*, where a joining peer p connects to an existing peer q with a probability

$$P(p,q) = \frac{d_q}{\sum_{k \in V} d_k} \tag{3.5}$$

where d_q is degree of existing node q, V is the set of nodes that have already joined the network, and $\sum_{k \in V} d_k$ is the sum of degrees of all nodes that previously joined the network. For this purpose we used the BRITE software for generating topologies of P2P networks [2]. The result is a topology represented by a weighted graph with edge weights representing communication delays in milliseconds. We evaluated our experiments with varied number of peers ranging from 100 to 500. We did not experiment with more peers, since it would result in unrealistically small sizes of local data at peers and could adversely affect the performance of P2P classification systems.

For local computation at each peer and monitoring of message exchange, we have built our own simulator (using Java) that simulates distributed dynamic P2P overlay trees. We have used communication delays on BRITE network graph as measurement of time. Local computations and message exchange are regulated with change of network state, with respect to a common global clock of a simulator.

Data: To compare the classification methods, we used two large benchmark data sets from widely used repositories.

- *covertype* (581012×54, 7 classes) data set from UCI repository [24], and
- *cod-rna*(488565×8, 2 classes) data set from LIBSVM repository [14].

Computing Environment Experiments were conducted on a cluster of 41 machines, each with 10 Intel-Xeon 2.4GHz processors, 20GB of Ram, and connected by gigabit Ethernet. For learning local classifiers, we used the LIBSVM ([14]) implementation of RSVM. Moreover, to compare the number of support vectors generated by RSVM with regular SVM, the C-SVC implementation of regular SVM is used.

Data Distribution We split the original data into train and test sets. As the peer distribution in power-law based topology is highly skewed in P2P systems, TRedSVM considers a probabilistic distribution of training data such that size $|X_p|$ of data on a peer p is directly proportional to its degree d_p (and ultimately hubness level),such that $|X_p| = |\mathcal{X}|.\frac{d_p}{\sum d}$, where $|\mathcal{X}|$ is the size of original (centralized) data set. Figure-3.3 depicts the distribution of the sizes of local training sets among all peers. The resulting distribution reflects that most of the data is randomly distributed among very few hub peers, and a vast majority of peers are weak and are allocated the remaining very small portion of data.

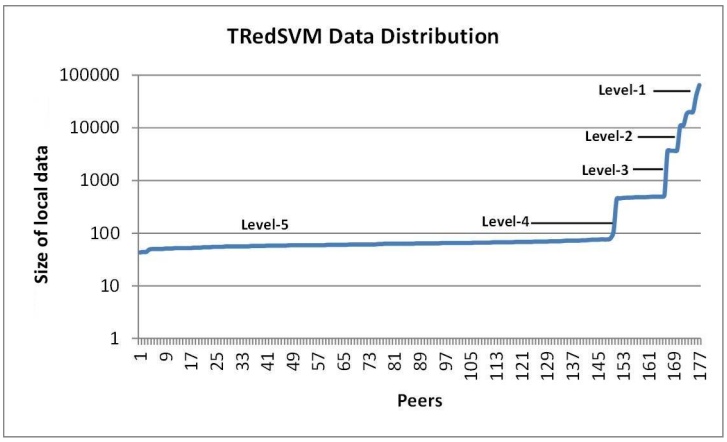

Figure 3.3: Data Distribution Among Peers in Transitive Classification

3.4.2 Experimental Results

In our experiments, we compared transitive classification approach with *three* baselines: with local model without any exchange, a scheme which only exchanges with direct neighbors and finally with flooding (uncontrolled propagation - every peers sends to every other peer). These comparisons are made on the basis of two criteria, i.e. classification accuracy and communication cost, both as in average and per peer. Figure 3.4 depicts the resulting average accuracy for the four examined methods. As expected, the simple baseline that involves no exchange of models (denoted as 'Local Model') is clearly outperformed by all other methods. Transitive classification compares favorably to the method that involves model exchange only among direct neighbors. It is worth to notice that the accuracy of transitive classification is close to that of the method which involves flooding (i.e., uncontrolled exchange of models without reducing their size). In this context, the accuracy attained by the flooding method can be considered as the upper bound, and thus the accuracy of our approach is close to it.

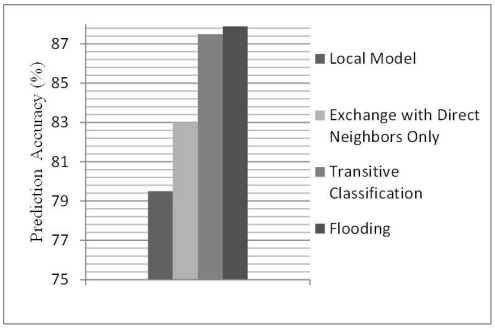

Figure 3.4: Average Prediction Accuracy of Transitive Classification as Compared with Local Models, Direct Exchange Method and Flooding

To understand the efficiency of transitive classification in terms of communication cost, Figure 3.5 illustrates the results for three examined methods, since the simple baseline that involves no exchange of models has by definition no communication cost. As

mentioned before, the accuracy attained by flooding acts as the upper bound. However, from the results in Figure-3.5 it becomes clear that flooding requires a prohibitive communication cost. Thus, it cannot comprise a feasible method in real P2P networks. Transitive classification requires communication cost that is substantially less than that of flooding and only marginally larger compared to the method that involves exchange of models only between direct neighbors. This result indicates that our approach offers a controlled increase in the communication cost that results in an important increase in classification accuracy.

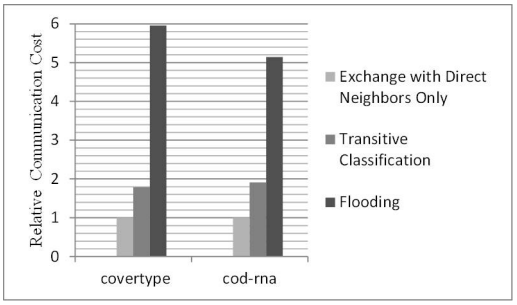

Figure 3.5: Relative communication cost of Transitive Classification compared with the methods of direct exchange and flooding

To provide further insight into the performance of transitive classification, we compared the average classification accuracy of all examined approaches separately for the peers of each hubness level. Figure-3.6 shows that for hub peers (Level-1), all methods attain comparable accuracy. This is natural to expect, since hub peers having adequately large training sets and thus gain nothing from model exchange. This explains also the good performance of the simple baseline (Local Model) that involves no exchange of models. However, for peers at lower levels, and especially for week peers (Level-5), transitive classification is able to provide important improvements in accuracy.

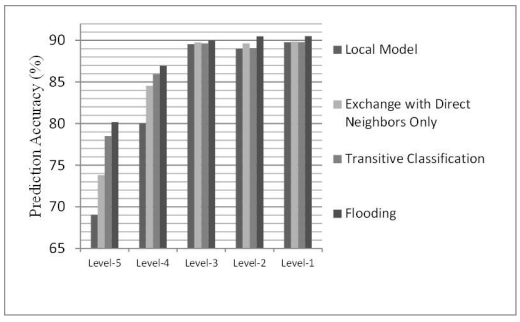

Figure 3.6: Average classification accuracy of peers at each hubness level and for each method

Finally, we examine the impact of parameter η at each transitive step in transitive classification. Figure-3.7 depicts in a combined way prediction accuracy and communication cost with respect to the percentage of support vectors used in RSVM. As expected,

communication cost increases steadily with increasing η values. Prediction accuracy also increases with increasing η values but only up to a threshold ($\eta = 10$), after which accuracy becomes steady. This clearly shows the intuition behind the principle we followed here, i.e., the need to keep the size of exchanged models under control, because otherwise communication cost increases rapidly without any gain in accuracy.

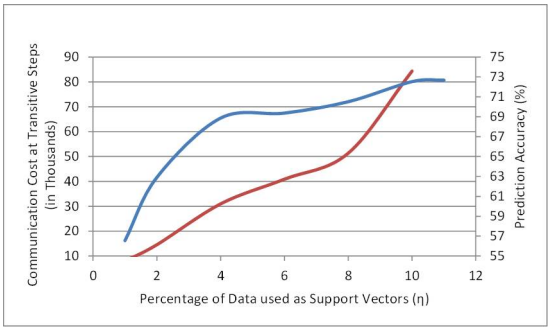

Figure 3.7: How prediction accuracy and communication cost varies with respect to η

3.5 Conclusion

We examined the problem of collaborative, distributed classification in large-scale P2P networks. We have proposed a transitive classification scheme that focuses on 'scale free' unstructured P2P networks whose topology exhibits power laws. Our method performs transitive propagation of classification models and helps scarcely connected peers to receive knowledge that improves the accuracy of their local classifiers. The transitive propagation of models beyond direct neighbors is performed in a controlled way that improves the classification accuracy and, at the same, keeps low the communication cost throughout the network. Our experimental evaluation showed that transitive classification improves the classification performance compared to several existing baseline methods and how it considers the trade-off between communication cost and prediction accuracy.

In our current work, we are focusing to simulate a P2P scenario in VANETS (as V2V), where vehicles can create a P2P network locally in a road segment. Cooperative information exchange and distributed learning models are put together in a V2V scenario to evaluate applications like intelligent lane selection on motorways, as described in Chapter 4.

Chapter 4

Distributed Learning in VANETS - Application Scenario

4.1 Introduction

Advanced traffic management systems are designed to reduce congestion and increase the overall traffic throughput. Many of such systems tend to manage traffic flows by optimizing the scheduling of traffic signals and highway ramp meters [40, 66]. A limitation of such system is that they consider traffic as a homogeneous entity and do not take into account the behavior of individual cars [62], in particular the coordination of cars themselves. Driving behaviors related to lane changes and speed control could be coordinated and optimized to create a smoother traffic flow, avoid congestion and reduce the overall fuel consumption.

Recently, motorway lane-changing has received increased attention [18, 37, 48]. Since lane-changing maneuvers often act as initial perturbations, it is crucial to understand their impact on the capacity, stability and breakdown of traffic flows. And in many cases, lane-changing is often a significant ingredient in triggering traffic breakdowns [77]. Moreover, it has a direct influence on traffic safety since a traffic accident also incurs major road congestion with effects in CO2 emissions. Despite of this significance, the ICT community has not devoted extensive research efforts to lane-changing. Data related to lane-changing behavior is scarcely available because neither cross-sectional data from detectors nor floating car data, are sufficient to provide lane-change parameters. Important lane-change properties include the rate of lane changes, the gaps and the velocities of the cars. Recent progress in video tracking methods, however, allows for a collection of high-quality trajectory data from aerial observations [27], and we expect to see more and more data becoming available in future.

The idea of vehicles coordinating with the neighboring ones on the highway, to optimize desired speed while minimizing rate of lane-changes, can be conceived as a machine learning problem. Each car receives local input from the surrounding traffic patterns and the desired speed of the driver and outputs the lane in which to drive. A vehicle's lane selection strategy should consider not only maintaining its own desired speed, but also how it would affect the speed of neighboring vehicles. In this way,

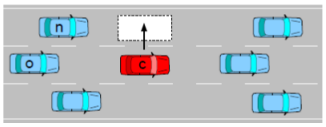

Figure 4.1: Vehicle c considering a lane-change to the left. The new and old successors are denoted by n and o, respectively. (From the book,*Traffic Flow Dynamics: Data, Models and Simulation*)

vehicles could organize themselves into a cooperative system which lets all the drivers acquire their desired speed and reduce deceleration, in addition to enabling all the vehicles in the road network to synchronize breaking. Hence, improving the fuel economy of the whole system.

4.1.1 Lane-change Model Formulation

Acceleration of a single vehicle in a car-following model is defined shown in Equation 4.1 as defined by [77]:

$$a_\alpha := \frac{dv_\alpha}{dt} = a\left(s_\alpha, v_\alpha, \Delta v_\alpha\right) \tag{4.1}$$

The acceleration depends on vehicle's own velocity v_α, the gap s_α to the front vehicle $(\alpha - 1)$ and the relative velocity $\Delta v_\alpha = v_\alpha - v_{\alpha-1}$.

Given these parameters, we can formulate the lane change model as follows. Consider a lane-change from the center lane to the outer lane as show in Figure 4.1. This change generally depends on the leader and the follower of the present and the target lane respectively. To define a lane-change criteria, note the lane-change intention of vehicle c, its successors on the target and present lane are represented by n and o, respectively. The current acceleration of vehicle c is denoted by α_c and its expected acceleration on the target lane is $\tilde{\alpha_c}$ for the same position and velocity. If c changes lane, $\tilde{\alpha_o}$ and $\tilde{\alpha_n}$, denote the acceleration of the old and new followers respectively. Moreover, the leader on the target lane is the nearest vehicle on this lane for which the position is $x > x_c$. Same is true for followers such that $x < x_c$.

We aim to follow an incentive criterion for lane-changing, proposed by [77]. It determines a driver's own advantage in terms of acceleration and his local traffic situation. At the same time, it considers the extent to which the immediate the neighbors are affected. For a driver of a vehicle c, the affected extent is defined as:

$$\tilde{\alpha_c} - \alpha c + p\left(\tilde{\alpha_n} - \alpha n + \tilde{\alpha_o} - \alpha o\right) > \Delta\alpha_{\text{threshold}} \tag{4.2}$$

The first subtraction $(\tilde{\alpha_c} - \alpha c)$ denotes the own lane-change advantage of the driver himself. This advantage would be attractive if the driver could drive faster in the target lane. A politeness factor p determines to which degree the driver in consideration, should show politeness to immediate affected neighbors. It is used to weight the total

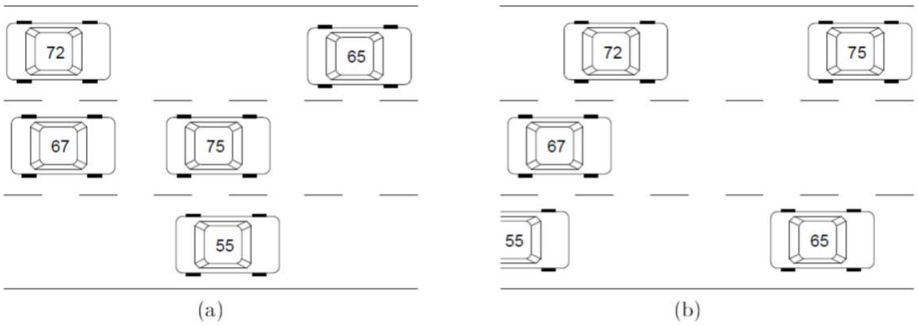

Figure 4.2: Intelligent Cooperative Lane-Change (Image extracted from [62])

advantage (or disadvantage in case of breaking/deceleration) of the immediate affected neighbors as represented by second term. Finally, this criterion is compared with a globally fixed threshold $\Delta\alpha_{\text{threshold}}$ which prevents lane changes if the overall advantage is only marginal compared to a 'keep lane' directive. Note that $\Delta\alpha_{\text{threshold}}$ affects lane changing decisions globally, while politeness parameter p affects it locally.

Since an individual driver can adjust its *politeness factor* p, to balance its own advantage with disadvantages of neighboring vehicles, the lane-changing model contains typical strategic features of classical game theory or reinforcement learning in machine learning. Politeness can vary from $p = 0$ (selfish lane-hoppers) to $p > 1$ to cooperative drivers who care for traffic situation of the followers. To explain this concept, let us consider the cooperative lane-changing situation illustrated in Figure 4.2.

This is an example traffic situation in which the traffic flows from left to right and the number on each car shows the car's speed. Left figure (a), illustrates a situation where lane coordination is beneficial. Vehicle 72 (numbers are speeds) is quickly approaching vehicle 65 and will be unable to pass because of the position of car 67. If there is no coordination, vehicle 65 forces vehicle 72 to reduce its speed and wait for vehicle 67 to let vehicle 65 pass. This will reduce the traffic throughput and also results in dissatisfaction of driver of vehicle 72. With cooperative and intelligent lane-changing, an efficient solution would be for vehicle 75 and vehicle 65 to immediately swap lanes, followed by vehicle 65 moving into the bottom lane, as depicted in right figure (b).

Hence, intelligently coordinated cooperative lane changes tend to increase the sum of acceleration of all involved vehicles or minimize the overall braking induced by lane-changes. This objective can be represented as an optimization problem by maximizing the objective function, in Equation 4.3, for the set of vehicles $V = (v_1, v_2, \ldots, v_N)$ moving in the current road segment.

$$\text{Flow}(V) = \frac{\sum_{v_i/time} \left(Speed_{(actual)} - Speed_{(desired)}\right)^2}{N \times time} - \frac{\sum_{v_i} (NumLaneChanges)}{time} \tag{4.3}$$

To maximize objective function in 4.3, each vehicle in the system, needs its current speed, the desired speed in the target lane, speeds of immediate neighbors and whether a neighbor is equipped with cooperative lane selection system.

4.1.2 Distributed Learning for Cooperative Lane Changes

Distributed learning algorithms for ad-hoc P2P networks discussed in Section 3 can be used to maximize traffic flow as described in Equation (4.3). Using beacon messages, in a V2V scenario, vehicles can exchange their current speeds, latitude and longitude coordinates to the neighboring vehicles. A local learning algorithm at each vehicle takes as input current and desired speeds of its own vehicle, relative speeds of the neighbors, location coordinates of neighbors and outputs the decision, i.e. move left, move right or stay in the current lane. Such a classification model is evaluated in a traffic simulator and makes lane change decisions in a certain percentage of cars. We measure the fitness of a model by evaluating equation 4.3. A model is reinforced if it results in an increased flow, otherwise it is penalized by the learning algorithm. Using this scheme, each model is tested and updated in several simulation trials and ultimately only those models are retained which maximize the traffic flow to the larger extents.

Since different vehicles would have been driven through different traffic situations, exchanging their learned models would help individual vehicles to perform intelligent and cooperative lane changes and reduce the overall braking by selfish or sudden lane changes.

4.1.3 Simulation

Currently, we are developing a simulation scenario for cooperative lane-change scenario, in coordination with University of Thessaly (UTH). Our distributed learning model can be combined with a clustering technique developed by UTH.

We have considered VEINS (*www.veins.car2x.org*), an open source framework for running vehicular network simulations. Veins is based on two simulators, a road traffic simulator and a Network simulator. Road traffic simulation is performed by SUMO, which is well-established in the domain of traffic engineering. Network simulation is performed by OMNeT++ along with the physical layer modelling toolkit MiXiM, which makes it possible to employ accurate models for radio interference, as well as shadowing by static and moving obstacles. In Veins the simulators are bidirectionally coupled and simulations are performed online.

Chapter 5

Bag-of-Patterns Representation of Time-Series

The current Chapter describes in details a method which is used to identify local patterns from time-series data. The elaborated methodology will be utilized in the subsequent Eco-Driving Chapter 6, where driving behavior patterns will be extracted from velocity plots (driving behavior time series) of the GPS recordings.

5.1 Introduction

Classification of time series is an important domain of machine learning due to the widespread occurrence of time-series data in real-life applications. Measurements conducted in time are frequently encountered in diverse domains ranging from medicine and econometrics up to astronomy. Therefore, time series has attracted considerable research interest in the last decade and a myriad of classification methods have been introduced.

Most of the existing literature on time-series classification focuses on classifying short time series, that is series which mainly incorporate a single long pattern. The research problem within this family of time-series data is the detection of pattern distortions and other types of intra-class pattern variations. Among other successful techniques in this category, the nearest neighbor classifier equipped with a similarity metric called Dynamic Time Warping (DTW) has been shown to perform well in a large number of datasets[21].

Nevertheless, few studies [9, 56, 55] have been dedicated towards the classification of time-series data which is long and composed of many local patterns. Avoiding the debate on the categorization of the words *short* and *long*, we would like to emphasize the differences between those two types of data with a clarification analogy. For instance, assume we would like to measure the similarity between two strings (i.e. words or time-series of characters). In such a case, string similarity metrics will scan the characters of those two strings, in a sequence, and detect the degree of matchings/similarities. In contrast, if we are searching for similarities between two books, then a character

by character similarity is meant to fail because the arrangement of words in different books is never exactly the same. Text mining approaches which compute bags of words occurring in a text and then utilize histograms of the occurrences of each word, have been recently applied to the time-series domain by the metaphor of computing "bags of patterns" [55, 9].

This paper presents a novel method to classify long time series composed of local patterns occurring in an unordered fashion and by varying frequencies. Our principle relies on detecting local polynomial patterns which are extracted in a sliding window approach, hence fitting one polynomial to each sliding window segment. As will be detailed in Section 5.3.3 we propose a quick technique to fit sliding window content which has a linear run-time complexity.

Once the polynomial coefficients of each sliding window content are computed, then we convert those coefficients into symbolic forms, (i.e. alphabet words). The motivation for calling the method Symbolic Polynomial arises from that procedure. Such discretization of polynomial coefficients, in the form of words, allows the detection of similar patterns by converting close coefficient values into the same literal word. In addition, the words computed from the time series allow the construction of a dictionary and a histogram of word frequencies, which enables an efficient representation of local patterns.

We utilize an equivolume discretization of the distributions of the polynomial coefficients to compute the symbolic words, as will be explained in Section 5.3.4. Threshold values are computed to separate the distribution into equal volumes and each volume is assigned one alphabet letter. Consequently, each polynomial coefficient is assigned to the region its value belongs to, and is replaced by the region's character. Ultimately, the word of a polynomial is the concatenation of the characters of each polynomial coefficient merged together. The words of each time series are then stored in a separate 'bag'. A dictionary is constructed with each word appearing at least once in the dataset and a histogram is initialized with each row representing a time series and each column one of the words in the dictionary. Finally, the respective frequencies of words are updated for each time series and the rows of the histogram are the new representation of the original time series. Such a representation offers a powerful mean to reflect which patterns (i.e. symbolic polynomial words) and how often they occur in a series (i.e. the frequency value in each histogram cell).

The novelty of our method, compared to state-of-art approaches [56, 55] which utilize constant functions to express local patterns, relies on offering an expressive technique to represent patterns as polynomial of arbitrary degrees. Furthermore, we present a fitting algorithm which can compute the polynomial coefficients for a sliding window segment in linear time, therefore our method offers superior expressiveness without compromising run-time complexity.

Empirical results conducted on datasets from the health care domain, demonstrate qualitative improvements compared to the state-of-art techniques in terms of classification accuracy, as is detailed in Section 5.4.4. We experimented with the datasets from [55], and in order to widen the variety of data we introduce three new datasets. Our method wins in all four datasets with a clear statistically significance in three of them, proving the validity of our method. We add experiments regarding the running time of the method and we show that our method is practically fast and feasible.

5.2 Related Work

5.2.1 Time-Series Representations

In order to understand the regularities embedded inside time-series, a large number of researchers have invested efforts into deriving and discovering time series representations. The ultimate target of representation methods is to encapsulate the regularities of time-series patterns by omitting the intrinsic noise. Discrete Fourier transforms have attempted to represent repeating series structures as a sum of sinusoidal signals [22]. Similarly, wavelet transformations approximate a time-series via orthonormal representations in the form of wavelets [12]. However, such representations perform best under the assumption that series contain frequently repeating regularities and little noise which is not strictly the case in real-life applications. Singular Value Decomposition is a dimensionality reduction technique which has also been applied to extract latent dimensionality information of a series [10], while supervised decomposition techniques have aimed at incorporating class information into the low-rank data learning [31].

In addition to those approaches, researchers have been also focused on preserving the original form of the time series without transforming them to different representations. Nevertheless, the large number of measurement points negatively influence the run-time of algorithms. Attempts to shorten time series by preserving their structure started by linearly averaging chunks of series points. Those chunks are converted to a single mean value and the concatenation of means create a short form known as a Piecewise Constant Approximation [46]. A more sophisticated technique operates by converting the mean values into symbolic form into a method called Symbolic Aggregate Approximation, denoted shortly as SAX [54, 80]. SAX enables the conversion of time-series values into a sequence of symbols and offers the possibility to semantically interpret series segments. Further sophistication of lower bounding techniques have advanced the representation method towards efficient indexing and searching [74], enabling large scale mining of time series [11]. Nonlinear approximations of the series segments have also been proposed. For instance least squares approximation of time series via orthogonal polynomials have been proposed for segmentation purposes in a hybrid sliding/growing window scenario [26]. Throughout this paper we will propose a novel representation technique based on the utilization of polynomial functions of an arbitrary degree to approximate sliding windows of a time series. Our method brings novelty in converting the coefficients into literal representations, while the ultimate form is the frequency of the literal words constructed per each sliding window.

5.2.2 Time-Series Similarity Metrics

The time-series community has invested considerable efforts in understanding the notion of similarity among series. Time series patterns exhibit high degrees of intra and inter class variation, which is found in forms of noisy distortions, phase delays, frequency differences and signal scalings. Therefore, accurate metrics to evaluate the distance among two series play a crucial role in terms of clustering and classification accuracy. Euclidean distance, commonly known as the L_2 norm between vectors, is a fast metric which compares the offset of every pair of points from two series, belonging to the same time stamp index. Despite being a fast metric of linear run-time complexity, the Euclidean distance is not directly designed to detect pattern variations.

A popular metric called Dynamic Time Warping (DTW) overcomes the deficiencies of the Euclidean distance by allowing the detection of relative time indexes belonging to similar series regions. DTW achieves highly competitive classification accuracies and is regarded as a strong baseline [21]. Even though DTW is slow in the original formulation having a quadratic run-time complexity, still recent techniques involving early pruning and lower bounding have utilized DTW for fast large scale search [69].

Other techniques have put emphasis on the need to apply edit distance penalties for assessing the similarity between time series [15, 16]. Such methods are inspired by the edit distance principle of strings which counts the number of atomic operations needed to convert a string to the other. In the context of time series the analogy is extended to the sum of necessary value changes needed for an alignment. Other approaches have put emphasis on detecting the longest common subsequence of series, believing in the assumption that time series have a fingerprint segment which is the most determinant with respect to classification [79]. Detection of similarities in a streaming time-series scenarios motivated attempts to handle scaling and shifting in the temporal and amplitude aspects[17].

5.2.3 Time-Series Classification

During the past decade, most time series classification attempts have targeted the classification of short time series. Nevertheless, the definition of shortness might lead to ambiguous understandings, therefore we would dedicate some room for further clarifying our definition. For instance, assume a time-series dataset representing outer shapes of two different tree leaves, hence a binary classification. Such series include one single pattern each and do not exceed hundreds of data points, therefore we define them simply short. On the other hand assume we have an imaginary large dataset containing concatenated outer shapes of all the leaves of a forest. Then our imaginary task is to compare different forests for classifying which continent the forest is located in. In the second case, the time series are a combination of many local patterns occurring at varying frequencies and positions. This category of datasets is hereafter defined as long time series.

5.2.3.1 Classifying Short Time Series

Classification of short time series has gathered considerable attraction in the literature. Among the initial pioneer methods and still one of the best performing ones is the nearest neighbor classifier accompanied by the DTW distance metrics, which constitute a hard-to-beat baseline [21]. Other powerful nonlinear classifiers like the Support Vector Machines have been tweaked to operate over time series, partially because originally the kernel functions are not designed for invariant pattern detection and partially because DTW is not a positive semi-definite kernel [33]. Therefore the creation of positive semi-definite kernels like the Gaussian elastic metric kernel arose [84]. Another approach proposed to handle variations by inflating the training set and creating new distorted instances from the original ones [32].

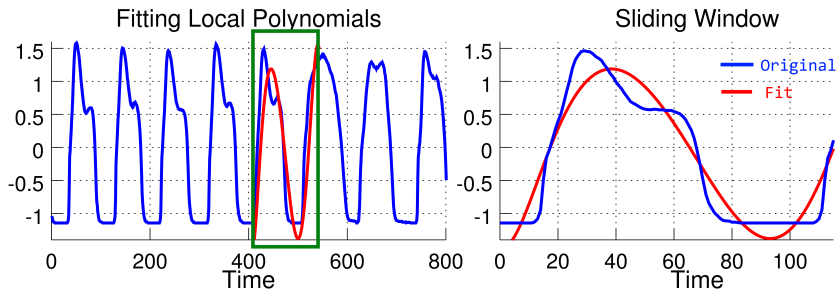

Figure 5.1: Fitting a local polynomial of degree 8 to the sliding window region indicated by a green box. The plot on the right shows a scaled up version of the polynomial fit with coefficients $\beta = [8.4 \times 10^{-15}, -5.7 \times 10^{-12}, 1.6 \times 10^{-9}, -2.4 \times 10^{-7}, 2.15 \times 10^{-5}, -0.0011, 0.034, -0.36, -2.8]$ sorted from the highest monomial degree to the lowest. The time series on the left plot is a segment from the GAITPD dataset.

5.2.3.2 Classifying Long Time Series

The classification of long time series focuses on long signals which are composed of one or more types of patterns appearing in unpredicted order and frequencies. Principally, the classification of those series has been mainly conducted by detecting the inner patterns and computing statistics over them. For instance, underlying series patterns have been expressed as the motifs and the difference between the motif frequencies has been utilized [9]. Other approaches have explored the conversion of each sliding window segment into a literal word constructed by piecewise constant approximations and the SAX method [56, 55]. The words belonging to each time series are gathered in a 'bag' and a histogram of the words is constructed. The histogram vectors are the new representations of the time series. Such a technique has been shown to be rotation-invariant, because the occurrence of a pattern is not related to its position [55]. In contrast to the existing work, our novel study introduces an expressive histogram formulation based on literal words build from local pattern detection via polynomial approximations. Our model ensures scalability by computing in linear run-time.

5.3 Proposed Method

5.3.1 Preamble Definitions

5.3.1.1 Alphabet

An alphabet is an ordered set of distinct symbols and is denoted by Σ. The number of symbols in an alphabet is called the size of the alphabet and denoted by $\alpha = |\Sigma|$. For illutration purposes we will utilize the Latin variant for the English language composed of the set of character symbols $\Sigma = (A, B, C, \ldots, Y, Z)$.

5.3.1.2 Word

A word $w \in \Sigma^*$ from an alphabet is defined as a sequence of symbols, therefore one sequence out of the set of possible sequences of arbitrary length l, defined as the Kleene star $\Sigma^* := \cup_{l=0}^{\infty} \Sigma^l$. For instance CACB is a word from the English alphabet having length four.

5.3.1.3 Polynomial

A polynomial of degree d having coefficients $\beta \in \mathbb{R}^{d+1}$, is defined as a sum of terms known as monomials. Each monomial is a multiplication of a coefficient times the a power of the predictor value $X \in \mathbb{R}^N$, as shown in Equation 5.1. The polynomial can also be written as a linear dot product in case we introduce a new predictor variable $Z \in \mathbb{R}^{N \times (d+1)}$ which is composed of all the powers of the original predictor variable X.

$$\begin{aligned} \hat{Y} = \sum_{j=0}^{d} \beta_j X^j &= Z\beta, \\ \text{where } Z &:= [X^0, X^1, X^2, \ldots, X^d] \end{aligned} \tag{5.1}$$

5.3.1.4 Time Series

A time series of length N is an ordered sequence of numerical values and denoted by $S \in \mathbb{R}^N$. The special characteristics of time-series data compared to plain vector instances is the high degree of correlation that close-by values have in the sequence. A time-series dataset containing M instances is denoted as $T \in \mathbb{R}^{M \times N}$, assuming time series of a dataset have the same length.

5.3.1.5 Sliding Window Segment

A sliding window segment is an uninterrupted subsequence of S having length n denoted by $S_{t,n} \in \mathbb{R}^n$. The time index t represents the starting point of the series, while the index n the length of the sliding window, i.e. $S_{t,n} = [S_t, S_{t+1}, S_{t+2}, \ldots, S_{t+n-1}]$. The total number of sliding window segments of size n for a series of length N is $N - n$, in case we slide the window by incrementing the start index t by one index at a time.

5.3.2 Proposed Principle

The principle proposed in this study is to detect local patterns in a time series via computing local polynomials. The polynomials offer a superior mean to detect local patterns compared to constant or linear models, because they can perceive information like the curvature of a sub-series. Furthermore, in case of reasonably sized sliding

windows the polynomials can approximate the underlying series segment without overfitting. In this paper, we demonstrate that the polynomial fitting for the sliding window scenario can be computed in linear run-time. Once the local polynomials are computed, we propose a novel way to utilize the polynomial coefficients for computing the frequencies of the patterns. The polynomial coefficients are converted to alphabet words via an equivolume discretization approach. Such a conversion from real valued coefficients to short symbolic words allows for the translation of similar polynomials to the same word, therefore similar patterns can be detected. We call such words symbolic polynomials. The words belonging to the time series are collected in a large 'bag' of words, (implemented as a list), then a histogram is created by summing up the frequency of occurrence for each words. Each row of a histogram encapsulates the word frequencies of time series, (i.e. frequencies of local patterns). A histogram row is the new representation of the time series and is used as a vector instance for classification.

5.3.3 Local Polynomial Fitting

Our method operates by sliding a window throughout a time-series and computing the polynomial coefficients in that sliding window segment. The segment of time series inside the sliding window is normalized before being approximated to a mean 0 and deviation of 1. The incremental step for sliding a window is one, so that every subsequence is scanned. Computing the coefficients of a polynomial regression is conducted by minimizing the least squares error between the polynomial estimate and the true values of the sub-series. The objective function is denoted by L and is shown in Equation 5.2. The task is to fit a polynomial to approximate the real values Y of the time series window of length n, previously denoted as $S_{t,n}$.

$$\begin{aligned} L(Y,\hat{Y}) &= ||Y - Z\beta||^2 \\ Y &:= [S_t, S_{t+1}, S_{t+2}, \dots S_{t+n-1}] \end{aligned} \tag{5.2}$$

Initially, the predictors are the time indexes $X = [0, 1, \dots, n-1]$ and they are converted to the linear regression form by introducing a variable $Z \in \mathbb{R}^{n\times(d+1)}$ as shown below in Equation 5.3.

$$Z = \begin{pmatrix} 0^0 & 0^1 & \dots & 0^d \\ 1^0 & 1^1 & \dots & 1^d \\ \vdots & \vdots & \dots & \vdots \\ (n-2)^0 & (n-2)^1 & \dots & (n-2)^d \\ (n-1)^0 & (n-1)^1 & \dots & (n-1)^d \end{pmatrix} \tag{5.3}$$

The solution of the least square system is conducted by solving the first derivative with respect the polynomial coefficients β as presented in Equation 5.4.

$$\frac{\partial L(Y,\hat{Y})}{\partial \beta} = 0 \quad \text{leads to} \quad \beta = \left(Z^T Z\right)^{-1} Z^T Y \tag{5.4}$$

A typical solution of a a polynomial fitting is provided in Figure 5.1. On the left plot

we see an instantiation of a sliding window fitting. The sliding window of size 120 is shown in the left plot, while the fitting of the segment inside the sliding window segment is scaled up on the right plot. Please note that inside the sliding window the time is set relative to the sliding window frame from 0 to 119. The series of Figure 5.1 is a segment from the GAITPD dataset.

Since the relative time inside each sliding window is between 0 and $n-1$, then the predictors Z are the same for all the sliding windows of all time series. Consequently, we can pre-compute the term $P = \left(Z^T Z\right)^{-1} Z^T$ in the beginning of the program and use the projection matrix P to compute the polynomial coefficients β of the local segment Y as $\beta = PY$. Algorithm 1 describes the steps needed to compute all the polynomial coefficients of the sliding windows (starting at t) of every time series (indexed by i) in the dataset. For every time series we collect all the polynomial coefficients in a bag, denoted as $\Phi^{(i)}$. The outcome of the fitting process are the bags of all time series Φ. Please note that the complexity of fitting a polynomial to a sliding window is linear and the overall algorithm has a complexity of $O(M \cdot d \cdot n \cdot N)$, which considering $d << N, d << M$ and $n << N$, means linear run-time complexity in terms of N and M, that is $O(M \cdot N)$.

Algorithm 1 Polynomial Fitting of a Time-Series Dataset

Require: Dataset $T \in \mathbb{R}^{M \times N}$, Sliding window size n, Polynomial degree d
Ensure: $\Phi \in \mathbb{R}^{M \times (N-n) \times (d+1)}$
1: $P \leftarrow \left(Z^T Z\right)^{-1} Z^T$
2: **for** $i \in \{1 \dots M\}$ **do**
3: $\quad \Phi^{(i)} \leftarrow \emptyset$
4: $\quad$ **for** $t \in \{1 \dots N-n\}$ **do**
5: $\quad\quad Y \leftarrow [S_t^{(i)}, S_{t+1}^{(i)}, \dots S_{t+n-1}^{(i)}]$
6: $\quad\quad \beta \leftarrow PY$
7: $\quad\quad \Phi^{(i)} \leftarrow \Phi^{(i)} \cup \{\beta\}$
8: $\quad$ **end for**
9: **end for**
10: **return** $(\Phi^{(i)})_{i=1,\dots,M}$

5.3.4 Converting Coefficients To Symbolic Words

The next step of our study is to convert the computed polynomial coefficients Φ from Algorithm 1 into words. The principle of conversion is to transform each of the $d+1$ coefficient of every β of Φ to one symbol. Therefore, the extracted words have lengths of $d+1$ symbols. For each of the β values of the polynomial coefficients we construct the histogram distribution and divide it into regions of equal volume as shown in Figure 5.2. In the image we have divided the histogram into as many regions as the alphabet size ($\alpha = 4$) we would like to utilize. Such a process is called an equivolume discretization. The thresholds between the regions are named quantile points and are defined in the figure as yellow lines. Dividing the histogram into α many regions is equivalent to sorting the coefficient values and choosing the threshold values corresponding to indexes multiple of $\frac{1}{\alpha}$. For instance, dividing the histogram into 4 regions for an alphabet of size 4 requires thresholds values corresponding to indexes at $\frac{1}{4}, \frac{2}{4}, \frac{3}{4}$ of the total number of values, which means that the each region has

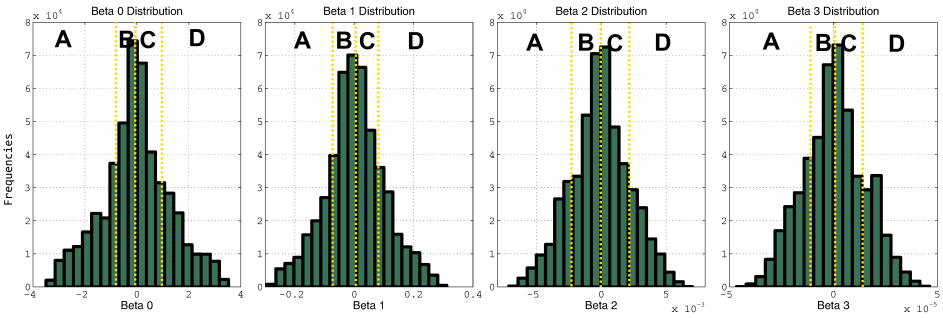

Figure 5.2: Equivolume Discretization of Polynomial Coefficients. The illustration depicts the histogram distributions of a third degree polynomial fit over the sliding windows of the RATBP dataset. Each plot shows the values of a polynomial coefficient versus the frequencies. The alphabet size is four corresponding to the set $\{A, B, C, D\}$. The quantile threshold points are shown by dashed yellow lines.

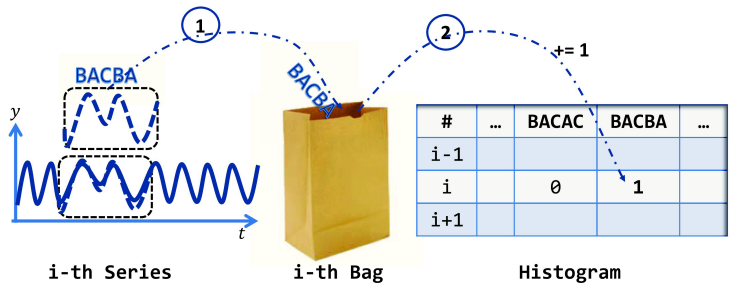

Figure 5.3: Polynomial Words Collection and Histogram Population. In the first step all the words of a series are stored in one 'bag' per each time series. Then a histogram is initialized with a column of zeros for each word that occurs in any bag at least once. During the second step the word frequencies in the histogram are incremented upon each word found in a bag.

25% of the values. Formally, let us define a sorted list of the j-th coefficient values regarding all window segments as $B^j \leftarrow \text{sort}\left(\left\{\beta_j \mid \beta \in \Phi^{(i)}, i = 1, \ldots, M\right\}\right)$ and let the size of this sorted list be $s^j \leftarrow |B^j|$. Then the $(\alpha - 1)$ many threshold values are defined as $\mu_k^j \leftarrow B^j_{\left\lfloor s^j \frac{k}{\alpha} \right\rfloor}, \forall k \in \{1, \ldots \alpha - 1\}$ and $\mu_\alpha^j \leftarrow \infty$.

Algorithm 2 describes the conversion of polynomial coefficients to symbolic form, i.e. words. The first phase computes the threshold values μ_k^j to discretize the distribution of each coefficient in an equivolume fashion. The second phase processes all the coefficients β of time series sliding windows and converts each individual coefficient to a character c, depending on the position of the β values with respect to the threshold values. The concatenation operator is denoted by the symbol $\circ$. The characters are concatenated into words w and stored in bags of words W. The complexity of this algorithm is also linear in terms of N and M. In case a linear search is used for finding the symbol index k, then the complexity is $O(M \cdot N \cdot \alpha)$, while a binary search reduces the complexity to $O(M \cdot N \cdot \log(\alpha))$. Yet, please note that in practice $\alpha << N$ and $\alpha << M$, therefore the complexity is translated to $O(M \cdot N)$.

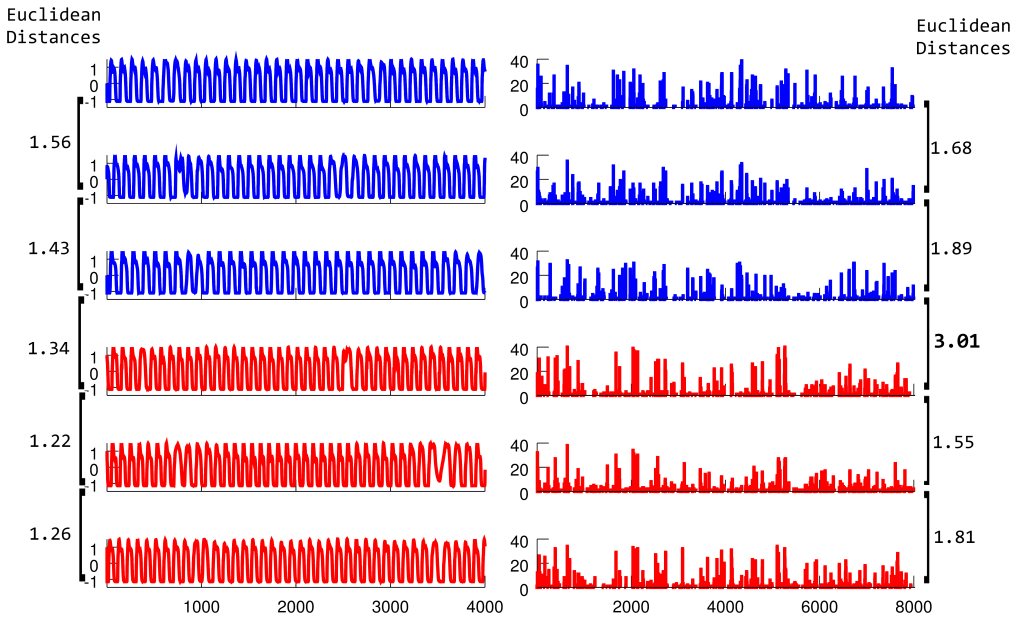

Figure 5.4: GAITPD Dataset: Time series describing the gait in a few, randomly selected, control patients (blue) and Parkinson's disease patients (red). The original time series (x-axis is time and y-axis the normalized value) are displayed on the left column, while the respective histogram of each time series is shown on the right. The x-axis of the right plot represent 8026 words of the dictionary while the y-axis is the frequency of each word. The distances on the right show that histograms among a class have lower Euclidean distances, while the distance between the two histograms belonging to different classes is much higher, at a value of 3.01.

Algorithm 2 Convert Polynomial Coefficients to Words

Require: Polynomial Coefficients Φ, Alphabet Size α
Ensure: $W \in \mathbb{R}^{M \times (N-n) \times (d+1)}$

{Compute the thresholds}
for $j \in \{0 \dots d\}$ **do**
  $B^j \leftarrow \text{sort}\left(\left\{\beta_j \mid \beta \in \Phi^{(i)}, i = 1, \dots, M\right\}\right)$
  $s^j \leftarrow |B^j|$
  $\mu_\alpha^j \leftarrow \infty$
  for $k \in \{1 \dots \alpha - 1\}$ **do**
    $\mu_k^j \leftarrow B^j_{\left\lfloor s^j \frac{k}{\alpha} \right\rfloor}$
  end for
end for
{Convert the coefficients to words}
$\Sigma \leftarrow \{A, B, \dots, Y, Z\}$
for $i \in \{1 \dots M\}$ **do**
  $W^{(i)} \leftarrow \emptyset$
  for $\beta \in \Phi^{(i)}$ **do**
    $w \leftarrow \emptyset$
    for $j \in \{0 \dots d\}$ **do**
      $k \leftarrow \text{argmax}_{k \in \{1, \dots, \alpha\}} \beta_j < \mu_k^j$
      $w \leftarrow w \circ \Sigma_k$
    end for
    $W^{(i)} \leftarrow W^{(i)} \cup \{w\}$
  end for
end for
return $(W^{(i)})_{i=1,\dots,M}$

5.3.5 Populating the Histogram

Once we have converted our polynomial coefficients and converted them to words, the next step is to convert the words into a histogram of word frequencies, as depicted in Figure 5.3. The steps of the histogram population are clarified by Algorithm 4. The first step is to build a dictionary D, which is a set of each word that appears in any time series at least once. Then we create a histogram H with as many rows as time-series and as many columns as there are words in the dictionary. The initial values of the histogram cells are 0. Each cell indicate a positive integer which semantically represent how many times does a word (column index) appear in a time series (row index). The algorithm iterates over all the words of a series and increases the frequency of occurrence of that word in the histogram.

Once the histogram is populated, then each row of the histogram denotes a vector containing the frequencies of the dictionary words for the respective time series. Practically the row represent what local patterns (i.e. words) exist in a series and how often they appear. For instance Figure 5.4 presents instances from the GAITPD dataset belonging to two types of patients in a binary classification task, healthy patients (blue) and Parkinson's Disease patients (red). In the left plot we show the original time series while on the right plot the histogram rows containing the polynomial words versus their frequencies. The parameters leading to the histogram for the GAITPD dataset are

Algorithm 3 Populate the Histogram

Require: Word bags W
Ensure: Histogram H
1: {Build the dictionary}
2: Ordered set dictionary $D \leftarrow \emptyset$
3: **for** $w \in W^{(i)}, \forall i \in \{1 \dots M\}$ **do**
4: **if** $w \notin D$ **then**
5: $D \leftarrow D \cup \{w\}$
6: **end if**
7: **end for**
8: {Build the histogram}
9: $H \leftarrow \{0\}_{i=1,\dots,M\ j=1,\dots,|D|}$
10: **for** $w \in W^{(i)}, \forall i \in \{1 \dots M\}$ **do**
11: Find j with $D_j = w$
12: $H_{i,j} \leftarrow H_{i,j} + 1$
13: **end for**
14: **return** H

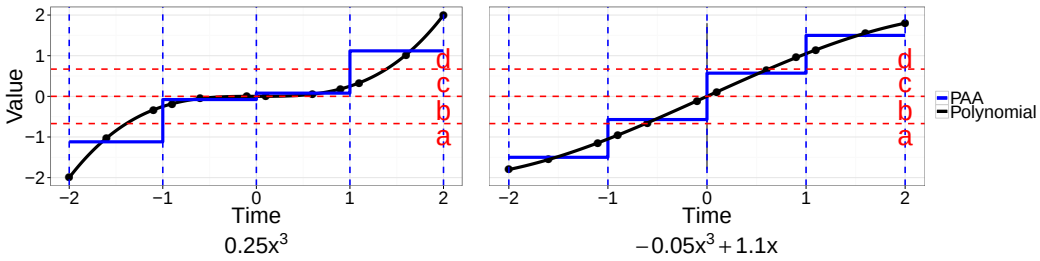

Figure 5.5: Comparison of Symbolic Polynomial method to SAX. As can be seen a constant model named Piecewise Aggregate Approximation (PAA) fails to detect the curvature of the patterns and results in the same SAX word 'ABCD' for both series.

$n = 100, \alpha = 4, d = 7$. As can be inspected the original time-series offer little direct opportunity to distinguish one class from the other and the series look alike. Moreover the Euclidean distance of adjacent series in the figure show that the Euclidean classifier would mistakenly classify the third instance of the blue class. In contrast the histograms are much more informative and it is possible to observe frequencies of local patterns which allow the discrimination of one class from the other. A complete distance matrix between blue B and red R instances is shown in Table 5.1. As can be seen our histogram representations result in perfect accuracy in terms of nearest neighbor classification (**bold**), while the original series result in 2 errors.

5.3.6 Comparison To Other Methods

The closest method comparable in nature to ours is the approach which builds histograms from SAX words [56, 55]. However the SAX words are build from locally constant approximations which in general are less expressive than the polynomials of our approach. Figure 5.5 demonstrate the deficiencies of the locally constant approxi-

Table 5.1: Distances Matrix between Time-series (left) and Histograms (right)

	B_1	B_2	B_3	R_1	R_2	R_3
B_1	-	1.6	**1.28**	1.4	1.29	1.4
B_2	1.6	-	1.4	1.5	1.4	**1.31**
B_3	1.28	1.4	-	1.4	**1.27**	1.4
R_1	1.4	1.5	1.4	-	**1.22**	1.4
R_2	1.29	1.4	1.27	**1.22**	-	1.26
R_3	1.4	1.31	1.4	1.4	**1.26**	-

	B_1	B_2	B_3	R_1	R_2	R_3
B_1	-	**1.7**	2.3	2.9	2.6	2.5
B_2	**1.7**	-	1.9	2.6	2.3	2.3
B_3	2.3	**1.9**	-	3.0	2.6	2.8
R_1	2.9	2.6	3.0	-	1.6	**1.4**
R_2	2.6	2.3	2.6	**1.6**	-	1.8
R_3	2.5	2.3	2.8	**1.4**	1.8	-

mation in detecting the curvatures of a sliding window sub-series. In the experiment of Figure 5.5, we used an alphabet of size four and utilized the classical quantile threshold for SAX, being values $\{-0.67, 0, 0.67\}$. Please note that the series are shown by black dots and represent normalized window segments. We have fitted both a constant model and our polynomial model to the series data. Assume we want to have a four character SAX word for each of the sliding windows segment. As can be easily seen the SAX words for both segments are 'ABCD'. On the other hand, referring to the coefficient values of Figure 5.2, we can see that the symbolic polynomial word belonging to polynomial $y(x) = 0.25x^3$ is 'CCBC', while the polynomial word belonging to $y(x) = -0.05x^3 + 1.1x$ is 'BCDC'. As we can see our method can accurately distinguish the difference between those patterns, while the SAX method averages the content and looses information about their curvatures. We would like to point out clearly that our method is more expressive by needing the same complexity, in this case both methods use four characters. In terms of run-time, SAX needs only one pass through the data, so from the algorithmic complexity point of view both methods have same algorithmic complexity of $O(M \cdot N)$, i.e. number of series by their length. The complexity of our method has an multiplicative constant, which is the degree of the polynomial as shown in Algorithm 2, however $d << M, d << N$. Finally, as will be shown in Section 5.4.4, the classification results of our method are much better than SAX histograms.

5.3.7 Classifier

The classifier that we are going to use is the nearest neighbor method, which is a strong classifier in time-series classification [21] and is used by state-of-art methods [55]. After converting the original time series into pattern frequency representations, in the form of rows of a histogram matrix, then each row will be treated as a vector instance. The nearest neighbor will utilize the Euclidean distance to compute the difference between histogram rows.

5.4 Experimental Setup

5.4.1 Descriptions of Datasets

All the experiments are based on datasets retrieved from Physionet, a repository of complex physiological signals primarily from the health care domain [30]. Our first

Table 5.2: Statistics of Datasets

Dataset	Number of Instances	Series Length	Number of Labels
ECG2	250	2048	5
GAITPD	1552	4000	2
RATBP	180	2000	2
NESFDB	840	1800	2

dataset, ECG2, represents time series from the domain of Electrocardiography belonging to five different sources [55]. The other datasets shown below were not used before in the realm of time-series classification, so we processed and adopted them for a classification task. Being the first paper to introduce them for time-series classification, we are dedicating some lines to explain the preparation of the datasets.

- GAITPD: The dataset contains measures of walking patterns (aka gait) from 93 patients with idiopathic Parkinson's Disease and 73 healthy (control) patients [35]. Measurements of the force underneath each foot was recorded for each patient via 8 sensors. We selected the total force of either feet as two independent series. The recorded gait time series were divided into equilength segments of 4000 measurement points. Finally we created a binary classification dataset by separating healthy and Parkinson's patients.
- RATBP: Hypertension caused by salt consumption was tested over two types of rats, the Dahl salt sensitive (SS) rats and the consomic SS.13BN rats [8]. Both high-salt and low-salt diets were provided to a total of 15 rats throughout a certain period of time. The amount of salt in the diet was alternated for each rat and the blood pressure series of each rat were measured. We segmented the blood pressure time series into equilength chunks of 2000 point measurements and utilized a binary categorization scheme by separating low-salt series from high-salt series.
- NESFDB: Postural sway refers to the oscillations of the body while standing. A group of 27 individuals of young and old age groups were voluntarily tested for postural sway behaviors [67]. Vibrating elements were implanted in the shoes of each volunteer beneath the fore foot and heel. Simulations on the patients were conducted by emitting low-pass filtered vibrational noise. A reflective marker was placed on the shoulders of the individuals and the sway was converted from a video recording into a time series describing the displacement of the marker. We utilized the series derived from both front and lateral video recordings as two instances. The time series were divided into two groups of sways, sway measurements with vibrational stimulus and measurements without stimulus.

The statistics of each dataset in terms of the number of instances, the length of each time series and the number of classes is summarized in Table 5.2. Please note that all the instances within one dataset have the same length.

5.4.2 Baselines

Let us name our method as SymPol, meaning **Sym**bolic **Pol**ynomials and refer to our method with the abbreviation form in the remaining sections. In order to evaluate the

performance of SymPol, we compare against the following three baselines.

5.4.2.1 BSAX

refers to the method of constructing bags of SAX words from time series through a sliding window approach. The words occurring in the bags are used to populate a histogram of frequencies [55]. A nearest neighbor method is applied to classify the histogram instances by treating the histogram rows as the new time-series representation. Comparing against this classifier will give chance to understand the benefit of polynomial approximation compared to constant models and will provide evidences on the state-of-art quality of the results.

5.4.2.2 ENN

is the classical nearest neighbor classifier with the Euclidean L_2 loss as the distance metric. It operates over the whole time series, without segmenting the series for local patterns. The comparison against the plain nearest neighbor will show whether the detection of local patterns has more advantage than comparing the whole long series.

5.4.2.3 DTWNN

differs from the Euclidean nearest neighbor classifier in defining a new distance metric for the comparison of two time series and performs well in time-series classification [21]. Dynamic Time Warping (DTW) operates by creating a matrix with all the possible warping paths, (i.e. alignment of pairs of indexes from two series), and selects the warping alignment with the smallest overall possible distance. DTW compares a full series without segmentations similarly to the Euclidean version of the nearest neighbor. Such comparison will both identify the benefits of the segmentation and also the benefits of local polynomials against global warping alignments.

Table 5.3: Error Rate Results

Dataset	**SymPol**		**BSAX**		**ENN**		**DTWNN**	
	mean	st.dev.	mean	st.dev.	mean	st.dev.	mean	st.dev.
ECG2	**0.0000**	0.0000	0.0080	0.0098	0.5480	0.0240	0.2120	0.0160
GAITPD	**0.0238**	0.0083	0.0548	0.0120	0.3924	0.0211	0.2468	0.0206
RATBP	**0.1333**	0.0272	0.1889	0.0111	0.4389	0.0272	0.3333	0.0994
NESFDB	**0.4310**	0.0212	0.4405	0.0395	0.4929	0.0208	0.5440	0.0356

5.4.3 Reproducibility

Two different type of experiments were conducted in our study. The first empirical evidence focuses on the accuracy of our method with respect to classification of time series. The second experiment will analyze the computational run time of the methods. All the experiments were computed in a **five folds cross-validation** experimental setup. The time-series instances of each dataset were divided into 5 sets. In a circular fashion (repeated five times) each different set was once selected as the testing set, while the remaining four were used for training. Among the four sets used for training, one of them was selected as a validation set and the remaining three left as training. As a summary, all the combination of parameters were evaluated on the validation set and learned on the three training set, while the parameter values giving the smallest errors on the validation were selected. Those parameter values were finally evaluated over the testing set (learning from the three training sets) to report the final error rate.

A grid search mechanism was selected for searching the hyperparameter values. Our method SymPol requires the tuning of three parameters, the size of the sliding window n, the size of the alphabet α and the degree of the polynomials d. The size of the sliding window was selected among the range of $n \in \{100, 200, 300, 400\}$, while the size of the alphabet was picked from $\alpha \in \{4, 6, 8\}$. Lastly the degree of the polynomial was picked to be one of $d \in \{1, 2, 3, 4, 5, 6, 7, 8\}$.

Similarly, the baseline named BSAX also requires the fitting of three hyperparameters. The length of a SAX word, denoted $|w|$, was selected from the range of $\{2, 3, 4, 5, 6, 7, 8, 9\}$, while the size of the alphabet was selected among the values $\{4, 6, 8\}$. The size of the sliding window is selected from a range of $\{100, 200, 300, 400\}$, however those values were rounded to fit the length of a sax word. For instance if the length of a Sax word is 3, then the size of the sliding window was rounded from 100 to 102 in order for the sliding window to be equally divisible into three chunks. The hyperparameter values found in our experiments are shown in Table 5.4, with ranges of multiple values due to different parameter searches per each different validation set.

Table 5.4: Hyper-parameter Search Results

Dataset	**SymPol**			**BSAX**		
	n	α	d	n	$\lvert w\rvert$	α
ECG2	100	4	3,4	100	4,6	4,6
GAITPD	100	4,6	7,8	100	6,7	6,8
RATBP	100	4,6	4,5,6,7	100	4,6,7	4,6,8
NESFDB	100,200	4,6,8	3,4,5	100,200,300	3,4	4,6

Table 5.5: Run Time Results (seconds)

Dataset	**SymPol**		**BSAX**		**ENN**		**DTWNN**	
	mean	st.dev.	mean	st.dev.	mean	st.dev.	mean	st.dev.
ECG2	4.1	0.1	3.3	3.8	2.0	0.2	1651.5	52.9
GAITPD	1124.0	807.4	535.0	198.2	91.7	0.9	337555.1	20015.2
RATBP	27.2	36.7	1.8	0.8	0.7	0.0	1124.1	17.2
NESFDB	18.3	2.9	10.2	2.8	12.0	1.6	19535.2	304.7

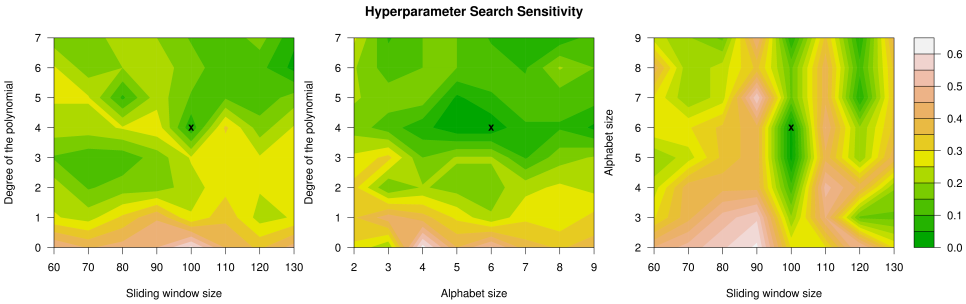

Figure 5.6: Hyperparameter Search Sensitivity. Parameter search for one fold of the RATBP dataset resulting in the optimal values $n = 100, \alpha = 6, d = 4$. In the two dimensional illustration the invisible parameter is fixed to the optimal.

5.4.4 Results

The classification accuracy results of our experiments are presented in Table 5.3. For our method SymPol and all the baselines we show the mean and the standard deviation of the five fold cross-validation experiments as described in the setup section. The smallest error rate is highlighted in bold.

As can be clearly seen our method demonstrates an unrivaled superiority. SymPol wins in all the four datasets and has a clear statistical significance in three of them ECG2, GAITPD, RATBP. Our method performs perfectly in the ECG2 dataset by having 100% classification accuracy. In addition, SymPol reduces the error on the GAITPD dataset by 57% with respect the closest baseline, while on the RATBP dataset the error is reduced by 29%.

The second type of results represent the running times of the algorithms and is shown in Table 5.5. As can be clearly seen the Euclidean distance on the original dataset is the fastest method, which is a natural behavior because no processing is done over series to extract histograms. The BSAX method is the next in terms of speed due to the computational advantage of the constant model which requires only one pass over the data. SymPol is positively positioned in terms of run time. As already analyzed before, the algorithmic complexity is comparable to the BSAX except for an additional constant, which is the polynomial degree. In datasets like ECG2, GAITPD and NESFDB the execution times are bigger by a small constant factor of two. The runtime constant in the RATBP dataset is higher because the hyperparameter search resulted to require a degree of 7. As a summary, we can clearly see that the method is practically very feasible and fast in terms of run time and is close even to techniques that require only a single scan over the time-series values.

5.4.5 Hyper-parameter Search Sensitivity

As presented in Section 5.4.3 our hyperparameter search technique is the grid search, where we scan for all the possible combination of one parameter's values to all the possible values of other parameters. As Figure 5.6 shows, the error rate is nonlinear with respect to the parameter values of the method. Therefore, a grid search mechanism is

practically suitable, because gradient based methods would have resulted in local optima while nonlinear optimization techniques would require much more computations than the grid. As can be seen in the plot, the grid search could successfully detect the global optimum in the region denoted by a mark.

5.5 Conclusion

In this study we presented a novel method to classify long time series, which are composed of local patterns. Local polynomial approximations are computed in a sliding window approach for each normalized segment under the sliding window. The computed polynomial coefficients are converted to symbolic forms (i.e. literal words) via an equivolume discretization procedure. Thresholds for the distribution of the values of each coefficient are determined to split the coefficient's histogram into equal regions and each region is assigned an alphabet symbol. In a second step all the polynomial coefficients are transformed into characters by locating them within the threshold values of the histogram and assigning the region symbol. The final literal representation of a polynomial is a word composed of the concatenation of each coefficient's character, in the order of the coefficient's monomial degrees. Once the bags of words are computed then a histogram is populated with the frequencies of each word in a time series. We presented a linear time technique to compute the polynomial approximation of a sliding window segment, while the overall method has a run time complexity which is linear in terms of the series points.

The classification accuracy of the nearest neighbor method utilizing the histogram rows that our method computed was compared against the performance of three baselines. Our method won all the experiments, most of them with a statistically significant margin. Furthermore, empirical results demonstrate that our method has a practically feasible running time performance, comparable even to the fastest methods which require a single scan over the time series.

Chapter 6

Eco-Driving Data Analysis By Mining Local Driving Behaviors

6.1 Introduction

The rising cost of fuel have increased the attempts to reduce transportation costs by adopting driving behaviors that minimize fuel consumption. Eco-Driving is a recent paradigm of transportation aiming at adopting driving behaviors in order to decrease fuel consumption. Even though a series of methods were developed to tackle Eco-driving, they are mostly based on heuristics triggered by instantaneous behavioral data. A review of state-of-art Eco-driving methodologies is described in Section ”6.2”.

In contrast to existing approaches, this deliverable proposes a new approach towards discovering influential driving patterns with respect to fuel consumption. The new paradigm focuses on analyzing driving data collected in forms of velocity plots from vehicles, where local behavioral patterns are extracted using the bag of patterns method described in Section ”6.4”. The data used to describe driving behaviors are the GPS records measured in the vehicles as detailed in Section ”Data Preparation”. The latitude and longitude information and the time stamps are converted to velocity plots which are time-series representing instantaneous velocity values.

In order to analyze the driving behavior the fuel consumption of each velocity plot is computed using the VT-Micro model [34]. The problem is therefore converted to a time-series regression, where the target is fuel, a continuous variable. The reader should not confuse the time-series regression with time-series forecasting, where the aim is to forecast the successive point given the past observations. The formulation of the emission model is presented in Section ”6.4.1”.

Since our fuel estimation model can be applied to the test instances as well, our problem is more specific than a plain regression. Instead the goal is to understand why a particular velocity plot results in high or low fuel consumption. In order to allow a fair comparison we selected driving behaviors from the same road segment. Such a restriction eliminates external factors such as inclination, road conditions and different speed limits, among others. Section ”6.5” is dedicated to the methodological aspect of

extracting most influential patterns, which accurately approximate the fuel consumption.

In the end of the study, we conduct and present the experimental results that demonstrate the validity of our method. We utilize the Infati dataset which is composed of high-frequency GPS recordings [38]. After applying our method on data filtered from driving behaviors of the most populous segment, we observe interesting results. The most influential patterns over fuel consumption are patterns characterizing instant accelerations and decelerations as will be shown in Section "6.6". Therefore, our study offers an empirical and data-driven evidence to the intuition which attributes non-eco friendly driving to sudden accelerations and decelerations.

6.2 Related Work

Modern eco-driving systems support the driving experience using messages shown to the driver [23, 1]. On-board awareness panels are equipped with indicators which display the actual degree of eco-friendliness based on the recent driving performance [23, 1]. For instance, green lights are typically displayed as a consequence of minimalistic fuel consumption. Eco-friendly behaviors are highly associated with constant cruising velocity at particular levels, which results in optimal fuel economy.

In contrast to real-time systems, Nissan offered off-line evaluation of driving data, which were processed by a data center upon driver request [39]. The feedback contained an analysis of fuel inefficient trends, aiming at improving the fuel efficiency of the driver's overall behavior. Collected large-scale intelligence lead, nevertheless, to an on-board assistance system which motivates ecological driving by providing comparative fuel efficiency rankings among a pool of users [39].

Other methodologies which provide ecological assistance are based on control theory, where the optimality of a model composed of parameters like weight, gear and velocity is computed [71]. More realistic systems take into account contextual information like traffic jams, road signals, road inclination and inter-vehicular distance. The outcome of such an assistance relies on providing advisory messages to the driver, in forms of actions pointing to either keep driving or reducing the pressure on the pedal [76].

Nevertheless, few state-of-art methodologies provide information regarding long-term eco-driving guidance. Among those, the ecological driver-assistance system, denoted as EDAS, supports the driver along urban roads [41]. EDAS incorporates a control theory model that forecasts future vehicle states via a dynamic modeling of the engine. The optimum control strategy of the vehicle, which results in long-term fuel efficiency, is computed.

Fuel consumption is sensitive to the inclination of the road [59]. In order to extend the functionality on diverse terrains, containing up-down slopes, initial attempts utilized dynamic programming in order to solve a drive cycle [36, 47]. Improved fuel economy was achieved by using digital road-map data on hilly roads, in a combination of terrain and vehicle dynamics models [42].

In comparison to the state-of-art, this study proposes a novel off-line data driven analysis of Eco-driving. Our method relates the historical driver's behavior data with the behaviors of other drivers on similar circumstances. Such an analysis can correctly

identify fuel inefficient behaviors within a range of drivers. In addition, our eco-driving study operates over GPS data only, therefore its range of applicability is potentially wider.

6.3 Data Preparation

The raw data used to mine the driving behaviors are the GPS position recordings of vehicles while conducting a trip. The recordings of a whole trip encapsulate a trajectory and is denoted by $T = (p_1, p_2, \ldots, p_n)$. Each recording p_i is measured at a particular frequency and is composed of at least three fields $p_i = (\phi_i, \lambda_i, t_i)$. The position of the vehicle at the moment of measurement is described by the latitude ϕ_i and longitude λ_i, while time by t_i.

6.3.1 Computing Instantaneous Velocity

The instantaneous velocity of the vehicle is computed by dividing the distance between two successive positions with the time difference. In order to compute the distance between two GPS positions on earth, we utilize the Great-circle distance, defined as the shortest distance between two points of a sphere [28]. The central angle between two successive positions p_{i-1}, p_i is given by the Spherical Law of Cosines [29] as shown in Equation 6.1. Such an angle is formed between the center of earth and the two locations on the surface.

$$\begin{aligned} \sigma_i &= \arccos\left(\sin\phi_{i-1}\sin\phi_i + \cos\phi_{i-1}\cos\phi_i\cos\left(\lambda_i - \lambda_{i-1}\right)\right), \\ &\quad \forall i \in \{2, \ldots, n\} \end{aligned} \tag{6.1}$$

However the angle of Equation 6.1 results in numerical rounding errors for small distances. The Vincenti formula [28], presented in Equation 6.2 is more accurate on computing angles on ellipsoids.

$$\begin{aligned} \triangle\lambda &= \lambda_i - \lambda_{i-1} \\ \sigma_i &= \arctan\left(\frac{\left(\cos\phi_i \sin\ \triangle\lambda\right)^2 + \left(\sin\phi_{i-1}\cos\phi_i - \sin\phi_{i-1}\sin\phi_i\cos\ \triangle\lambda\right)^2}{\sin\phi_{i-1}\sin\phi_i + \cos\phi_{i-1}\cos\phi_i\cos\ \triangle\lambda}\right) \\ &\quad \forall i \in \{2, \ldots, n\} \end{aligned} \tag{6.2}$$

Once the angle between two GPS positions is identified, then the distance is computed as a product of the angle with the earth radius, as is depicted in Equation 6.3.

$$D_i = R\,\sigma_i, \quad \forall i \in \{2 \ldots n\} \tag{6.3}$$

A good choice for the radius of the earth is the Mean Earth Radius [61], which is defined in Equation 6.4. Please note that $a = 6378.137$ km is the equatorial radius of

a flattened sphere approximation of the earth, while $b = 6356.752$ km is the distance from the center of the sphere to each pole.

$$R = \frac{1}{3}(2a + b) \approx 6371 \text{ km} \tag{6.4}$$

Finally we can compute the instantaneous velocity of a vehicle from two successive GPS position recordings as shown in Equation 6.5.

$$V_i = \frac{D_i}{t_i - t_{i-1}}, \quad \forall i \in \{2, \ldots, n\} \tag{6.5}$$

6.3.2 Matching Positions To Map Segments

Matching a GPS position to a map segment is often referred as Map-Matching and constitutes an important slice of geo-spatial research. Nevertheless, the extensive review of related map-matching works is not the aim of this study and the focus will lie only on describing the used algorithm.

Map-matching can be categorized as either on-line and off-line matching. On-line matching requires instant matching, while off-line matching can utilize the full trajectory after the trip has ended [65]. Our scenario refers to off-line matching of GPS trajectories. The Marchal algorithm is an off-line matching that operates in a step-wise fashion [60]. In the first step, the segments are mapped to trajectory points nearby and each segment represents a candidate. The score of a candidate segment is based on the sum of Euclidean distance least-squares between a trajectory and the segment. The optimal candidates are characterized by minimal scores.

An improvement of the Marchal algorithm aims at correcting errors due to non-smooth matching and outliers [65]. The improvement relies on computing a better matching sequence through a non-linear optimization approach called GEMMA [65]. A genetic algorithm is designed, where a gene is one trajectory point and an individual is a matching sequence. The alleles represent close-by links and the fitness function is composed by criteria such as geometric continuity and the transition between matched and unmatched parts. Finally, a hybrid approach combining the Marchal algorithm and GEMMA is denoted M-GEMMA [65].

6.4 Mining Driving Patterns

The method used to extract local driving behaviors is based on the technique of Chapter 5. which extracts local patterns from time series. A sliding window is traversed throughout the velocity plot and local polynomials are fit to the content of the segment inside each sliding window. The polynomial coefficients are discretized into symbolic alphabet characters, via an equi-volume discretization of each coefficient's distribution. The granularity of discretization is based on a so-called alphabet size. Finally a histogram is constructed using the frequencies of the symbolic words occurring in each velocity plot. The algorithm is elaborated and formalized in detail in Chapter 5

and will not be revisited thoroughly. Nevertheless, an illustration of extracting local patterns as polynomials is provided in Figure 6.1. The velocity plot belongs to the Infati dataset [38]. Please let us emphasize that the output of the pattern mining method is the histogram of local patterns and such a histogram will be later used to detect the most influential driving patterns. In the histogram structure, each column represent a local pattern, while each cell contains the frequency of the occurrence of that pattern in a velocity plot (row).

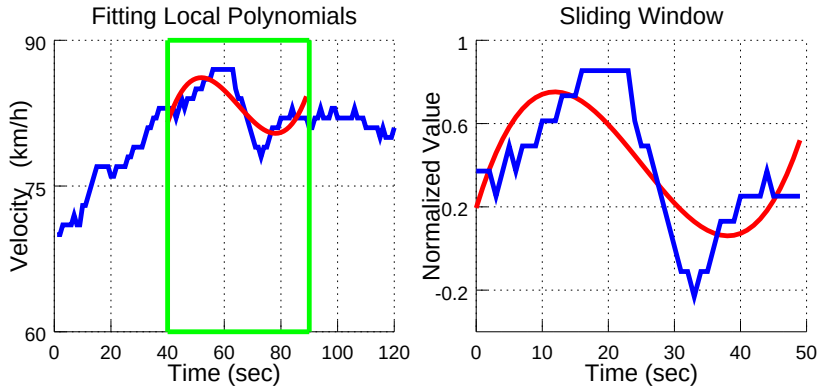

Figure 6.1: A velocity plot from the Infati dataset [38]: Local Driving Patterns as Polynomials

6.4.1 Fuel Estimation Model

The estimation of fuel consumption is instrumental in providing an evidence on how fuel efficient a particular velocity plot is. In absence of real fuel consumption evidences, since we only posses the GPS recordings, then the VT-Micro estimation model is used to approximate the fuel consumption based on instantaneous velocity and acceleration. In order to proceed with a definition of the fuel consumption estimation model, we define the instantaneous acceleration in Equation 6.6.

$$A_i = \frac{V_i - V_{i-1}}{t_i - t_{i-1}}, \qquad \forall i \in \{2, \ldots, n\} \tag{6.6}$$

The estimation of the fuel consumption is carried through a model called VT-Micro which is based on the instantaneous acceleration and velocity [34]. The model is described in Equation 6.7, while it depends on the powers of the instantaneous acceleration and velocity in addition to a set of constants K, L. The constants are calibrated using experiments with real vehicles and the overall model has a proportional approximation to the real fuel consumption of a vehicle [34]. However, care should be paid to distinguish that the model has no real units, since it is an approximation of proportions and not exact fuel units. Please note that $\exp(x) = e^x$.

$$F_i(V_i, A_i) = \exp\left(\sum_{j=0}^{3}\sum_{k=0}^{3} C_{j,k}(A_i)\, V_i^j\, A_i^k\right), \quad \forall i \in \{2,\ldots,n\} \tag{6.7}$$

$$C_{j,k}(A_i) = \begin{cases} K_{j,k} & A_i \geq 0 \\ L_{j,k} & A_i < 0 \end{cases}, \quad j = 0,\ldots,3; \;\; k = 0,\ldots,3$$

The coefficients of the VT-Micro for both acceleration (K) and deceleration (L) are empirically calibrated by comparing to the consumption of light vehicles [34]. The values of the coefficients are shown in Table 6.1.

Table 6.1: Calibrated VT-Micro Model Coefficients

$K =$

	0	1	2	3
0	-8.7605e-001	8.1221e-002	3.7039e-002	-2.5500e-003
1	3.6270e-002	9.2460e-003	-6.1800e-003	4.6800e-004
2	-4.5000e-004	-4.6000e-004	2.9600e-004	-1.7900e-005
3	2.5500e-006	4.0000e-006	-1.8600e-006	3.8600e-008

$L =$

	0	1	2	3
0	-7.5584e-001	-9.2100e-003	3.6223e-002	3.9680e-003
1	2.1283e-002	1.1364e-002	2.2600e-004	-9.0000e-005
2	-1.3000e-004	-2.0000e-004	4.0300e-008	2.4200e-006
3	7.3900e-007	8.4500e-007	-3.5000e-008	-1.6000e-008

6.5 Selection Of Influential Patterns

Once the histogram containing the local patterns is built, denoted by H, then we progress towards identifying the local patterns which are influential in estimating fuel consumption. The columns of the histogram matrix serve as predictors for estimating the fuel consumption. The estimation model used is the Least Square Support Vector Machines [75]. The most influential patterns are selected using a forward search, in forms of variables' subsets. Those patterns are selected in a greedy fashion, where the next most-influential pattern is added to the list of previous influential predictors.

6.5.1 Regression model

The estimation of the fuel consumption is described as a regression problem. The predictor variables are the patterns of the histogram $H \in \mathbb{R}^{(N+N')\times M}$, composed of N training velocity plots and N' test velocity plots. The number of patterns (columns) of the histogram is denoted by M. The target variable is the fuel consumption denoted as $F^{Train} \in \mathbb{R}^N, F^{Test} \in \mathbb{R}^{N'}$, while the task is to correctly estimate the fuel consumption of the test instances F^{Test}, via learning over the training instances F^{Train}. The statistical regression model used to estimate the fuel consumption is the Least Squares Support Vector Machines (LSSVM) [75]. The objective function of the LSSVM is focused on minimizing the least squares error together with the maximum

margin regularization. The solution of such an objective function is provided in Equation 6.8. Please note that the solution is expressed via the dual weights $\alpha \in \mathbb{R}^N$ and bias $W_0 \in \mathbb{R}$. The hyper-parameter λ controls the over-fitting of the model.

$$\begin{bmatrix} W_0 \\ \alpha \end{bmatrix} = \begin{bmatrix} 0 & \vec{1}^T \\ \vec{1} & K + \lambda I \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ F^{Train} \end{bmatrix} \tag{6.8}$$

The kernel $K \in \mathbb{R}^{N \times N}$ represents similarities between histogram rows, i.e. velocity plots. An example kernel function is the polynomial kernel defined in Equation 6.9. The degree of the polynomial, denoted d, is a hyper-parameter of the LSSVM model.

$$K(H_{i,:}, H_{l,:}) = (\langle H_{i,:}, H_{l,:} \rangle + 1)^d, \quad d \in \mathbb{N}; \;\; i = 1, \ldots, N; \;\; l = 1, \ldots, N \tag{6.9}$$

Once the weights are learned, a prediction of the fuel consumption for any velocity plot l, can be conducted using the prediction model of Equation 6.10.

$$\hat{F}_l(H) = W_0 + \sum_{i=1}^{N} \alpha_i \, K(H_{i,:}, H_{l,:}), \quad l = 1, \ldots, N + N' \tag{6.10}$$

The quality measurement of our method is conducted using the Mean Square Error metric defined in Equation 6.11. Please note that the quality metric is applied only over the testing instances.

$$\text{MSE}(F^{Test}, \hat{F}) = \frac{1}{N'} \sum_{l=1}^{N'} \left(F_l^{Test} - \hat{F}_{l+N} \right)^2 \tag{6.11}$$

6.5.2 Forward Variable Selection

The projection operator, defined by π, is inherited from relational calculus. When applied to a matrix, given the column ids as parameters the projection selects a smaller matrix. The selection contains only the columns indicated by the list of numbers fed as arguments to the projection operator. The formalization is shown in Equation 6.12.

$$\pi_{\{j,k,\ldots,l\}}(H) = [H_{:,j} H_{:,k} \ldots H_{:,l}], \quad \forall j, k, \ldots, l \in \{1 \ldots M\} \tag{6.12}$$

The selection of the influential patterns is conducted using a greedy search called forward selection. The idea is described in Algorithm 4. The input of the algorithm are the histogram H and the desired number of patterns K. The output of the algorithm is the list of column indexes in the histogram representing the most influential patterns.

The algorithm operates by starting with an empty list of influential pattern indexes. The first iteration selects the column of the histogram which can best approximate the real fuel consumption. The selected first pattern is the most influential in predicting the

Algorithm 4 Most Influential Patterns Selection

Require: Histogram H, Number of top patterns K
Ensure: List of top pattern indexes L
1: $L \leftarrow \emptyset$
2: **for** $k = 1 \dots K$ **do**
3: $j^* \leftarrow \text{argmin}_j \left\{ \left(F - \hat{F}\left(\pi_{L\cup\{j\}}(H)\right)\right)^2 \;\middle|\; j \notin L \;\wedge\; j = 1, \dots, \text{length}(H) \right\}$
4: $L \leftarrow L \cup \{j^*\}$
5: **end for**
6: **return** L

consumption and is added to the list L. In each successive iteration the next pattern is applied jointly with the previously selected indexes. Such a forward search follows the greedy paradigm since the best list is incremented by preserving the results of previous iterations.

6.6 Experimental Results

6.6.1 Experimental Setup

In order to validate the methodology presented in this Chapter we will utilize the Infati dataset [38]. The dataset is composed by the GPS recordings of trips from 24 test vehicles in a period spanning over four months. The frequency of the GPS records is one second. In order to enable a fair comparison between driving behaviors we selected the road segment having the most trips. Such a decision helps minimizing the influence from external factors like inclination and road conditions. The selected road segment's id is 5490. Since the driving behaviors, i.e. velocity plots, have different lengths we have to crop equilength-ed chunks of the driving behaviors. The threshold was selected to be the first two minutes of driving behavior, while velocity plots less than 120 seconds were omitted. Further statistics on the utilized dataset are described in Table 6.2.

Table 6.2: Statistics of Infati Dataset - Segment 5490

Number of Velocity Plot	Length of Plot	Mean Fuel Consumption (VT-Micro)
89	120	116.70

The VT-Micro model described in Section "6.4.1" is utilized to estimate the fuel consumption of the Infati dataset. The distribution of fuel consumption is provided in the histogram of Figure 6.2. As can be observed, most behaviors consume little fuel consumption in the range of a hundreds, while a few driving behaviors are extremely inefficient and consume excessive amounts of fuel.

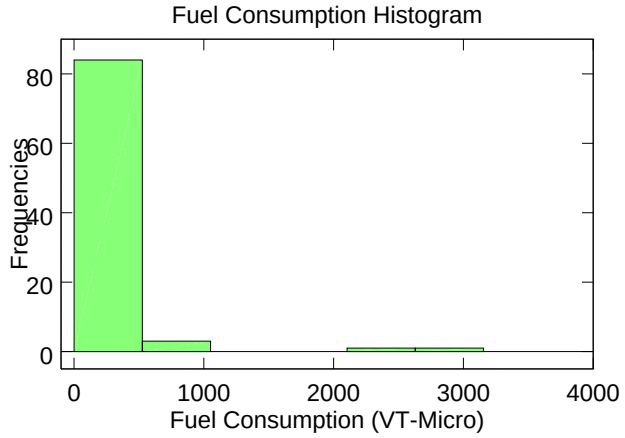

Figure 6.2: The histogram distribution of the fuel consumptions of velocity plots

6.6.2 Results of Influential Patterns

The algorithm described in Section ”6.4” for mining local patterns requires the fitting of three parameters: the size of the sliding window, the degree of the polynomial and the alphabet size. In addition, the Least Square SVM needs the setting of the complexity parameter and the degree of the polynomial kernel. All the parameters are set in a cross-validation search over validation data splits. The mining of influential patterns is conducted using the aforementioned greedy forward search. The results of the approximative errors combined with the incremental patterns search is shown in Figure 6.3. As we can observe, few patterns already help decrease the mean square error, while the error is minimized with as few as six top patterns. The increase in MSE after the addition of the last patterns is due to the noisy nature of those local patterns in estimating the target.

The top patterns of the Infati dataset which approximate the fuel consumption are illustrated in Figure 6.4. The order of the pattern indexes are sorted by the order of forward search, i.e. the most influential pattern is listed first. The influence is a measure of how good a particular pattern is at estimating the fuel consumption. As we can observe from the top patterns, the most influential driving behaviors are related to sudden acceleration and decelerations.

The reason why sudden accelerations and decelerations are selected as top patterns lies within the criteria of the greedy forward search, because such patterns are present in driving behaviors which have a high fuel consumption. Since the mean square error is mostly influenced by high fuel consumption then the model is steered towards detecting patterns present in inefficient driving behaviors. A demonstration of the sudden acceleration and decelerations present in the most fuel inefficient velocity plots of the Infati dataset is found in Figure 6.5.

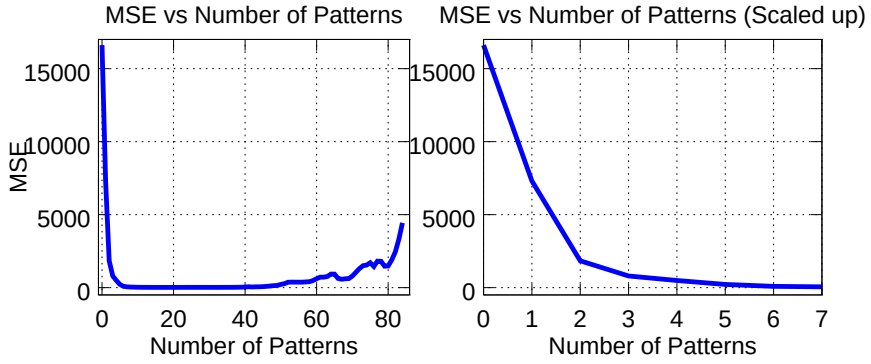

Figure 6.3: Impact of selecting the most influential patterns on fuel consumption (Sliding window 40, Alphabet size 6, Polynomial degree 3, LSSVM λ 1.0, LSSVM polynomial degree 3)

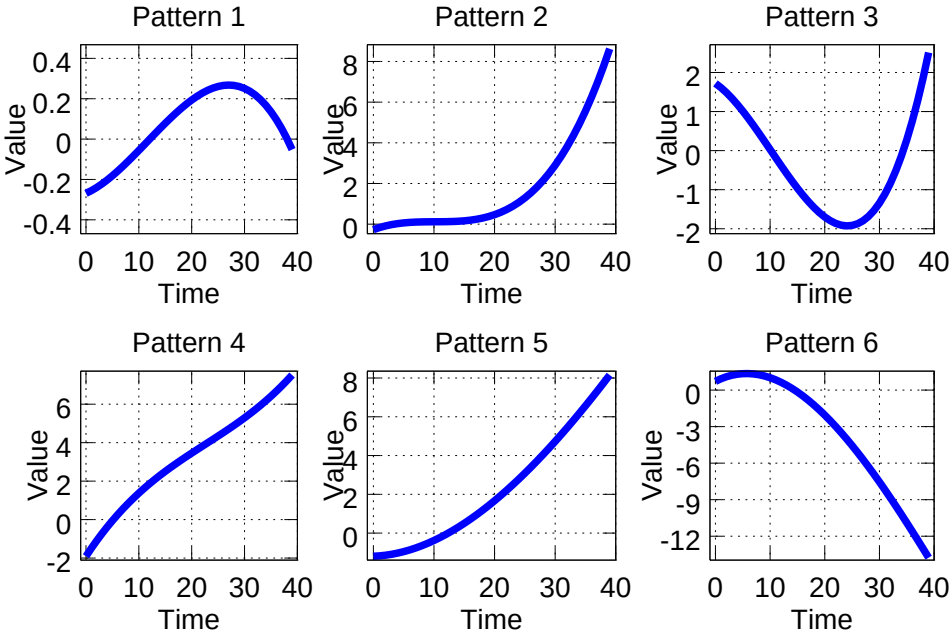

Figure 6.4: The top six most influential driving patterns (Sliding window 40, Alphabet size 6, Polynomial degree 3)

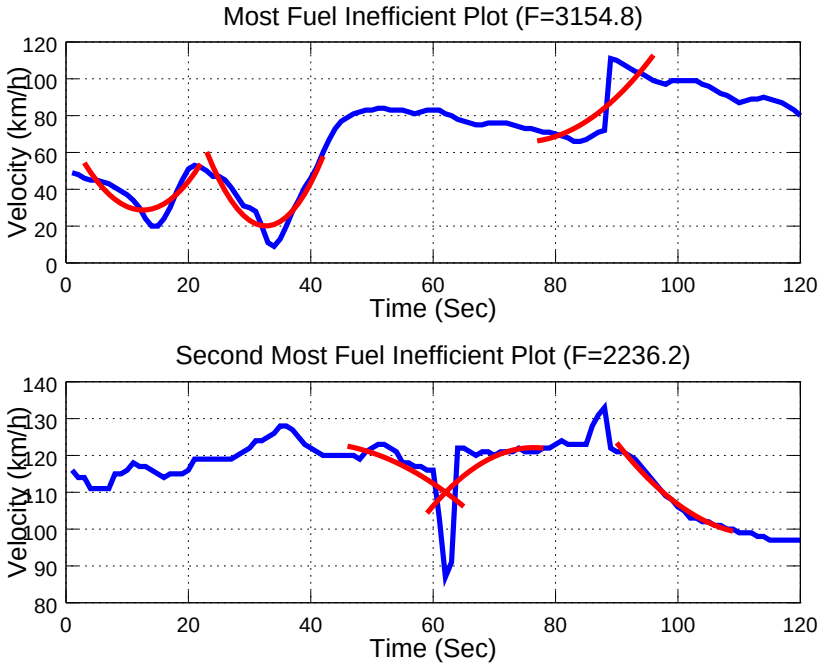

Figure 6.5: Analysis of cases occurring in fuel-inneficient velocity plots (Sliding window 20, Polynomial degree 2)

6.7 Conclusion

The method presented in this deliverable is focused on detecting fuel inefficient patterns from driving behavior data. High frequency GPS recordings were converted to velocity plots by computing the distance between two successive latitude and longitude measurements. The mining of local patterns in the velocity plot is conducted using the bag of patterns algorithm of Section 6.4, where the local patterns are stored as columns in a frequency histogram. In order to detect which of the extracted local patterns is more influential with respect to fuel consumption, the consumption of velocity plots is estimated. The VT-Micro estimation model is utilized for fuel estimation based on instantaneous velocity and acceleration.

The search for most influential patterns is carried via greedy forward search over the space of histogram columns, i.e. over local driving patterns. Experiments on driving behavior plots of the Infati dataset were conducted by selecting velocity plots from the most frequently traveled map segment. The top influential local patterns with respect to describing fuel consumption were the sudden accelerations and decelerations. The presented study completed the purpose of experimentally analyzing driving behaviors from driving data in the context of Eco-driving. Our analysis empirically validated the existing intuition that sudden accelerations and decelerations are the main reason for inefficient fuel consumption.

Chapter 7

Risk Assessment and Conclusion

7.1 Risk Assessment

Both sets of methodologies, regarding distributed data mining (Chapter 3) and Eco-Driving (Chapter 6) are already implemented and functional. There are no current (or expected) deviations from the objectives of DOW and the time-frames. The synthetic data for the distributed data mining is already generated, while data related to eco-driving is possessed [38] and is expected to be enriched by CTL use-case. Therefore, no deviations are expected in terms of data availability as well.

7.2 Conclusion

This deliverable focused on reporting the progress accomplished by the REDUCTION project in the contexts of distributed data mining and ecological driving. The scope of the deliverable lies within the Task 2.3 of the second work package, which is responsible for developing state-of-art methodologies. In the light of the objectives, the presented deliverable appropriately answered all the obligations with respect to both the problems of distributed data mining and ecological driving.

Regarding distributed data mining, Chapter 3 reported the development and application of decentralized statistical prediction model on a P2P communication topology. Local models were developed using state-of-art nonlinear classification model and the results of the statistical learning process were broad-casted to the neighborhood. The share of intelligence contributes significantly to the improvement of local classification accuracy. In addition the communication costs were kept to feasible levels due to the utilization of compact learning models for broadcasting. Empirical results demonstrated the validity of the results, by demonstrating the efficiency of the decentralized model and its superiority in terms of communication time compared to selected baselines.

Ecological driving was detailed in the Chapters 5 and 6 of the deliverable. The detection of inefficient patterns from the fuel consumption perspective is carried by mining the velocity plots of the GPS recordings of vehicles. Local patterns are mined through state-of-art techniques and represented as histograms of symbolic polynomials. Identification of local driving patterns which mostly influence fuel consumption is conducted by utilizing fuel estimation models. The patterns which best approximate the estimated fuel consumption are found via a greedy forward search technique. Finally experiments over the real-life GPS data of the Infati dataset were conducted. The top-most influential driving patterns were found to be sudden accelerations and deceleration, therefore our study empirically proves the hypothesis via the mining of behavioral data.

REDUCTION project accomplished all the objectives set in the scope of Work Package 2 within the time frame of Months 13-24. The progress in detecting ecological driving patterns is achieved by mining trajectory data. Similarly, distributed algorithms are proposed to tackle the de-centralization of local statistical inference.

Furthermore, the project achieved advances beyond the state of the art in the fields of eco-driving and distributed data mining. While current eco-driving methods operate on instantaneous data frames of real-time CANBUS signals, REDUCTION developed a more general approach. The proposed method operates off-line using only GPS data trajectories, which are cheaper and more widely present. Moreover, the state-of-the-art data mining methods for V2V networks were further improved with additional reductions in the sizes of the statistical models and the communication cost, by preserving the prediction accuracy.

Glossary

C-SVC Regularized Support Vector Classification. 18

DTW Dynamic Time Warping. 27

EDAS Ecological Driver-assistance System. 46

GPS Global Positioning System. 9

ICT Information and Communication Technologies. 7

LSSVM Least-square Support Vector Machines. 50

MANET Mobile Ad-hoc Networks. 12

P2P Peer to Peer. 12

REDUCTION Reducing Environmental Footprint based on Multi-Modal Fleet management Systems for Eco-Routing and Driver Behaviour Adaptation. 7

RSVM Reduced Support Vector Machines. 15

SAX Symbolic Aggregate Approximation. 29

SVM Support Vector Machines. 13

TREDSVM Transitive Reduced Support Vector Machines. 17

V2I Vehicles To Infrastructure. 7

V2V Vehicles To Vehicles. 6, 7

VANET Vehicular Ad-hoc Networks. 10

WANET Widespread Ad-hoc Networks. 12

Bibliography

[1] Team Minus 6%. 10 items of eco-driving performance (in japanese). [online]. Available `http://www.team-6.jp/`.

[2] Ibrahim Matta John Byers Alberto Medina, Anukool Lakhina. Brite: Boston university representative internet topology generator, 2002.

[3] Hock Hee Ang, Vivekanand Gopalkrishnan, Steven C. Hoi, and Wee Keong Ng. Cascade rsvm in peer-to-peer networks. In *Proceedings of the 2008 European Conference on Machine Learning and Knowledge Discovery in Databases - Part I*, ECML PKDD '08, pages 55–70, Berlin, Heidelberg, 2008. Springer-Verlag.

[4] Albert-Lszl Barabsi and Rka Albert. Emergence of scaling in random networks. *Science*, 286(5439):509–512, 1999.

[5] Kanishka Bhaduri and Hillol Kargupta. A scalable local algorithm for distributed multivariate regression. *Statistical Analysis and Data Mining*, 1(3):177–194, 2008.

[6] Kanishka Bhaduri, Ran Wolff, Chris Giannella, and Hillol Kargupta. Distributed decision-tree induction in peer-to-peer systems. *Stat. Anal. Data Min.*, 1(2):85–103, June 2008.

[7] Leo Breiman. Bagging predictors. *Mach. Learn.*, 24(2):123–140, August 1996.

[8] Scott M Bugenhagen, Allen W Cowley, and Daniel A Beard. Identifying physiological origins of baroreflex dysfunction in salt-sensitive hypertension in the dahl ss rat. *Physiol Genomics*, 42(1):23–41, 2010.

[9] Krisztian Buza and Lars Schmidt-Thieme. Motif-based classification of time series with bayesian networks and svms. In Andreas Fink, Berthold Lausen, Wilfried Seidel, and Alfred Ultsch, editors, *GfKl*, Studies in Classification, Data Analysis, and Knowledge Organization, pages 105–114. Springer, 2008.

[10] James A. Cadzow, Behshad Baseghi, and Tony Hsu. Singular-value decomposition approach to time series modelling. *Communications, Radar and Signal Processing, IEE Proceedings F*, 130(3):202–210, 1983.

[11] Alessandro Camerra, Themis Palpanas, Jin Shieh, and Eamonn Keogh. isax 2.0: Indexing and mining one billion time series. In *Proceedings of the 2010 IEEE International Conference on Data Mining*, ICDM '10, pages 58–67, Washington, DC, USA, 2010. IEEE Computer Society.

[12] Kin-Pong Chan and A.W.-C. Fu. Efficient time series matching by wavelets. In *15th International Conference on Data Engineering, Proceedings 1999*, pages 126–133, 1999.

[13] Philip Chan and Salvatore J. Stolfo. A comparative evaluation of voting and meta-learning on partitioned data. In *In Proceedings of the Twelfth International Conference on Machine Learning*, pages 90–98. Morgan Kaufmann, 1995.

[14] Chih-Chung Chang and Lin. Libsvm: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011.

[15] Lei Chen and Raymond Ng. On the marriage of lp-norms and edit distance. In *Proceedings of the Thirtieth international conference on Very large data bases - Volume 30*, VLDB '04, pages 792–803. VLDB Endowment, 2004.

[16] Lei Chen, M. Tamer Özsu, and Vincent Oria. Robust and fast similarity search for moving object trajectories. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, SIGMOD '05, pages 491–502, New York, NY, USA, 2005. ACM.

[17] Yueguo Chen, M.A. Nascimento, Beng-Chin Ooi, and A. Tung. Spade: On shape-based pattern detection in streaming time series. In *Data Engineering, 2007. ICDE 2007. IEEE 23rd International Conference on*, pages 786–795, 2007.

[18] B. Coifman, S. Krishnamurthy, and X. Wang. Lane-change maneuvers consuming freeway capacity. In *Traffic and Granular Flow 03*, pages 3–14. Springer Berlin Heidelberg, 2005.

[19] Dan. Power-law revisited: A large scale measurement study of p2p content popularity. In *Proc. of International Workshop on Peer-to-peer Systems (IPTPS)*, pages 430–441, apr 2010.

[20] Souptik Datta, Chris Giannella, and Hillol Kargupta. Approximate distributed k-means clustering over a peer-to-peer network. *IEEE Trans. on Knowl. and Data Eng.*, 21(10):1372–1388, October 2009.

[21] Hui Ding, Goce Trajcevski, Peter Scheuermann, Xiaoyue Wang, and Eamonn Keogh. Querying and mining of time series data: experimental comparison of representations and distance measures. *Proc. VLDB Endow.*, 1(2):1542–1552, August 2008.

[22] Christos Faloutsos, M. Ranganathan, and Yannis Manolopoulos. Fast subsequence matching in time-series databases. In *Proceedings of the 1994 ACM SIGMOD international conference on Management of data*, SIGMOD '94, pages 419–429, New York, NY, USA, 1994. ACM.

[23] FORD-WERKE. Schneller schalten, weiter kommen, cologne ford eco-driving, 2003.

[24] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[25] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Additive logistic regression: a statistical view of boosting. *Annals of Statistics*, 28:2000, 1998.

[26] Erich Fuchs, Thiemo Gruber, Jiri Nitschke, and Bernhard Sick. Online segmentation of time series based on polynomial least-squares approximations. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(12):2232–2245, December 2010.

[27] Deutsches Zentrum fur Luft-und Raumfahrt (DLR). Clearing house for transport data and transport models, 2007.

[28] Kenneth Gade. A non-singular horizontal position representation. *Journal of Navigation*, 63:395–417, 7 2010.

[29] Hellwich M. Kastner H. Gellert W., Gottwald S. and Kunstner H. *VNR Concise Encyclopedia of Mathematics*. New York: Van Nostrand Reinhold, 1989.

[30] A. L. Goldberger, L. A. N. Amaral, L. Glass, J. M. Hausdorff, P. Ch. Ivanov, R. G. Mark, J. E. Mietus, G. B. Moody, C.-K. Peng, and H. E. Stanley. PhysioBank, PhysioToolkit, and PhysioNet: Components of a new research resource for complex physiologic signals. *Circulation*, 101(23):e215–e220, 2000 (June 13). Circulation Electronic Pages: http://circ.ahajournals.org/cgi/content/full/101/23/e215 PMID:1085218; doi: 10.1161/01.CIR.101.23.e215.

[31] Josif Grabocka, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Classification of sparse time series via supervised matrix factorization. In Jörg Hoffmann and Bart Selman, editors, *AAAI*. AAAI Press, 2012.

[32] Josif Grabocka, Alexandros Nanopoulos, and Lars Schmidt-Thieme. Invariant time-series classification. In Peter A. Flach, Tijl De Bie, and Nello Cristianini, editors, *ECML/PKDD (2)*, volume 7524 of *Lecture Notes in Computer Science*, pages 725–740. Springer, 2012.

[33] Steinn Gudmundsson, Thomas Philip Runarsson, and Sven Sigurdsson. Support vector machines and dynamic time warping for time series. In *IJCNN*, pages 2772–2776. IEEE, 2008.

[34] Chenjuan Guo, Yu Ma, Bin Yang, Christian S. Jensen, and Manohar Kaul. Ecomark: evaluating models of vehicular environmental impact. In *Proceedings of the 20th International Conference on Advances in Geographic Information Systems*, SIGSPATIAL '12, pages 269–278, New York, NY, USA, 2012. ACM.

[35] J.M. Hausdorff, J. Lowenthal, T. Herman, L. Gruendlinger, C. Peretz, and N. Giladi. Rhythmic auditory stimulation modulates gait variability in parkinson's disease. *Eur J Neurosci*, 26(8):2369–75, 2007.

[36] Erik Hellström, Jan Åslund, and Lars Nielsen. Design of a well-behaved algorithm for on-board look-ahead control. In *IFAC World Congress*, Seoul, Korea, 2008.

[37] P. Hidas. Modelling vehicle interactions in microscopic simulation of merging and weaving. *Transportation Research Part C: Emerging Technologies*, 13(1):37–62, February 2005.

[38] Christian S. Jensen, H. Lahrmann, Stardas Pakalnis, and J. Runge. The Infati Data. *Computing Research Repository*, cs.DB/0410, 2004.

[39] M. Sugimoto K. Satou, R. Shitamatsu and E. Kamata. Development of an on-board eco-driving support system (in japanese). *Nissan Tech. Rev., no. 65(2009-9)*, 12:68–71, 2009.

[40] K. Kagolanu, R. Fink, H. Smartt, R. Powell, and E. Larsen. An intelligent traffic controller. 1995.

[41] M.A.S. Kamal, M. Mukai, J. Murata, and T. Kawabe. Ecological driver assistance system using model-based anticipation of vehicle–road–traffic information. *IET Intelligent Transport Systems*, 4(4):244–251, 2010.

[42] Murata Kamal, Mukai and Kawabe. Ecological vehicle control on roads with up-down slopes. *IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS*, 12:783–794, 2011.

[43] Murat Karakaya, İbrahim Körpeoğlu, and Özgür Ulusoy. Counteracting free riding in peer-to-peer networks. *Comput. Netw.*, 52(3):675–694, February 2008.

[44] Hillol Kargupta. Connected cars: How distributed data mining is changing the next generation of vehicle telematics products. In *Sensor Systems and Software*, volume 102 of *Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, pages 73–74. Springer Berlin Heidelberg, 2012.

[45] Hillol Kargupta and Philip Chan, editors. *Advances in Distributed and Parallel Knowledge Discovery*. MIT Press, Cambridge, MA, USA, 2000.

[46] Eamonn J. Keogh, Kaushik Chakrabarti, Michael J. Pazzani, and Sharad Mehrotra. Dimensionality reduction for fast similarity search in large time series databases. *Knowl. Inf. Syst.*, 3(3):263–286, 2001.

[47] F. Kirschbaum, M. Back, and M. Hart. Determination of the fuel-optimal trajectory along a known route. In *Proc. IFAC 15th Triennial World Congress*, Barcelona, Spain, 2002.

[48] Jorge A. Laval and Carlos F. Daganzo. Lane-changing in traffic streams. *Transportation Research Part B: Methodological*, 40(3):251 – 264, 2006.

[49] Aleksandar Lazarevic and Zoran Obradovic. Boosting algorithms for parallel and distributed learning. *Distrib. Parallel Databases*, 11(2):203–229, March 2002.

[50] Lee and Olvi L. Mangasarian. Rsvm: Reduced support vector machines. In *Data Mining Institute, Computer Sciences Department, University of Wisconsin*, pages 00–07, 2001.

[51] Sangkyun Lee, Marco Stolpe, and Katharina Morik. Separable approximate optimization of support vector machines for distributed sensing. In *ECML/PKDD (2)*, pages 387–402, 2012.

[52] Yuh-Jye Lee, Wen-Feng Hsieh, and Chien-Ming Huang. epsilon-ssvr: A smooth support vector machine for epsilon-insensitive regression. *IEEE Transactions on Knowledge and Data Engineering*, 17:678–685, 2005.

[53] Yuh-Jye Lee and Su-Yun Huang. Reduced support vector machines: A statistical theory. *IEEE Transactions on Neural Networks*, 18(1):1–13, 2007.

[54] Jessica Lin, Eamonn Keogh, Li Wei, and Stefano Lonardi. Experiencing sax: a novel symbolic representation of time series. *Data Min. Knowl. Discov.*, 15(2):107–144, October 2007.

[55] Jessica Lin, Rohan Khade, and Yuan Li. Rotation-invariant similarity in time series using bag-of-patterns representation. *J. Intell. Inf. Syst.*, 39(2):287–315, 2012.

[56] Jessica Lin and Yuan Li. Finding structural similarity in time series data using bag-of-patterns representation. In *Proceedings of the 21st International Conference on Scientific and Statistical Database Management*, SSDBM 2009, pages 461–477, Berlin, Heidelberg, 2009. Springer-Verlag.

[57] Kuanming Lin and Chihjen Lin. A study on reduced support vector machines. *IEEE TRANSACTIONS ON NEURAL NETWORKS*, 14:1449–1459, 2003.

[58] Ping Luo, Hui Xiong, Kevin Lü, and Zhongzhi Shi. Distributed classification in peer-to-peer networks. In *Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '07, pages 968–976, New York, NY, USA, 2007. ACM.

[59] J. Murata M. A. S. Kamal, M. Mukai and T. Kawabe. Ecological vehicle control based on road shapes prediction. *Proceedings of SICE 10th Conference on Control Systems, Paper 164-3-3*, 2010.

[60] F. Marchal, J. Hackney, and K. W. Axhausen. Efficient map-matching of large gps data sets - tests on a speed monitoring experiment in zurich. Technical report, Arbeitsbericht Verkehrs und Raumplanung, Volume 244, 2004.

[61] G. T. McCaw. Long lines on the earth. *Empire Survey Review 1 (6)*, pages 259–263, 1932.

[62] David Moriarty and Pat Langley. Distributed learning of lane-selection strategies for traffic management. Technical report, Technical Report 98-2). Daimler-Benz Research and Technology, 1998.

[63] Odysseas Papapetrou, Wolf Siberski, and Stefan Siersdorfer. Collaborative classification over p2p networks. In Sadagopan Srinivasan, Krithi Ramamritham, Arun Kumar, M. P. Ravindra, Elisa Bertino, and Ravi Kumar, editors, *Proceedings of the 20th International Conference on World Wide Web, WWW 2011, Hyderabad, India, March 28 - April 1, 2011 (Companion Volume)*, pages 97–98. ACM, 2011.

[64] Byung-Hoon Park and Hillol Kargupta. Distributed data mining: Algorithms, systems, and applications. In *Distributed Data Mining: Algorithms, Systems, and Applications*, pages 341–358, 2002.

[65] Francisco Camara Pereira, Hugo Costa, and NunoMartinho Pereira. An off-line map-matching algorithm for incomplete map databases. *European Transport Research Review*, 1(3):107–124, 2009.

[66] Farhard J Pooran, Philip J Tarnoff, and Ravi Kalaputapu. Rt-tracs: Development of the real-time control logic. In *Intelligent Transportation: Realizing the Benefits. Proceedings of the 1996 Annual Meeting of ITS America.*, 1996.

[67] A.A. Priplata, J.B. Niemi, J.D. Harry, L.A. Lipsitz, and J.J. Collins. Vibrating insoles and balance control in elderly people. *Lancet*, 362(9390):1123–4, 2003.

[68] Andreas L. Prodromidis, Philip K. Chan, and Salvatore J. Stolfo. Meta-learning in distributed data mining systems: Issues and approaches. In *Advances of Distributed Data Mining*, volume 114, pages 74–81. AAAI Press, 2000.

[69] Thanawin Rakthanmanon, Bilson Campana, Abdullah Mueen, Gustavo Batista, Brandon Westover, Qiang Zhu, Jesin Zakaria, and Eamonn Keogh. Searching and mining trillions of time series subsequences under dynamic time warping. In

Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining, KDD 2012, pages 262–270, New York, NY, USA, 2012. ACM.

[70] Matei Ripeanu, Ian Foster, and Adriana Iamnitchi. Mapping the gnutella network: Properties of large-scale peer-to-peer systems and implications for system design. *IEEE Internet Computing Journal*, 6:2002, 2002.

[71] Y. Saboohi and H. Farzaneh. Model for developing an eco-driving strategy of a passenger vehicle based on the least fuel consumption. *Applied Energy*, 86(10):1925–1932, 2009.

[72] Stefan Saroiu, Krishna P. Gummadi, and Steven D. Gribble. Measuring and analyzing the characteristics of napster and gnutella hosts. *Multimedia Syst.*, 9(2):170–184, August 2003.

[73] Izchak Sharfman, Assaf Schuster, and Daniel Keren. A geometric approach to monitoring threshold functions over distributed data streams. In Michael May and Lorenza Saitta, editors, *Ubiquitous Knowledge Discovery*, volume 6202 of *Lecture Notes in Computer Science*, pages 163–186. Springer Berlin Heidelberg, 2010.

[74] Jin Shieh and Eamonn Keogh. isax: indexing and mining terabyte sized time series. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '08, pages 623–631, New York, NY, USA, 2008. ACM.

[75] Johan A. K. Suykens and Joos Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.

[76] Daisuke Yamaguchi Yoichi Sato Takashi Ichihara, Shiro Kumano and Yoshihiro Suda. Driver assistance system for eco-driving. *ITS World Congress 2009*, 2009.

[77] M. Treiber, A. Kesting, and C. Thiemann. *Traffic Flow Dynamics: Data, Models and Simulation*. Springer-Verlag GmbH, 2012.

[78] Grigorios Tsoumakas, Ioannis Katakis, and Ioannis Vlahavas. Effective voting of heterogeneous classifiers. In *In Proceedings of the 15th European Conference on Machine Learning*, pages 465–476, 2004.

[79] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684, 2002.

[80] Li Wei, Eamonn Keogh, and Xiaopeng Xi. Saxually explicit images: Finding unusual shapes. In *Proceedings of the Sixth International Conference on Data Mining*, ICDM '06, pages 711–720, Washington, DC, USA, 2006. IEEE Computer Society.

[81] Ran Wolff, Kanishka Bhaduri, and Hillol Kargupta. Local l2 thresholding based data mining in peer-to-peer systems. In *SIAM Conf. Data Mining*, pages 430–441, 2006.

[82] Ran Wolff and Assaf Schuster. Association rule mining in peer-to-peer systems. In *Proceedings of the Third IEEE International Conference on Data Mining*, ICDM '03, pages 363–, Washington, DC, USA, 2003. IEEE Computer Society.

[83] Beverly Yang and Hector Garcia-m. Designing a super-peer network. 2003.

[84] Dongyu Zhang, Wangmeng Zuo, D. Zhang, and Hongzhi Zhang. Time series classification using support vector machine with gaussian elastic metric kernel. In *Pattern Recognition (ICPR), 2010 20th International Conference on*, pages 29–32, 2010.

[85] Yue-nan; Niu Xia-mu Zhu, Ce; Li. *Streaming Media Architectures, Techniques and Applications*. IGI Global, 2011.