UNIVERSITY OF THESSALY

Knowledge extraction from traffic flows in wireless networks, based on machine learning algorithms

by

Angeliki Kapodistria

A thesis submitted in partial fulfillment for the
degree of Master of Science
Science and Technology of Electrical and Computer Engineering

in the
School of Engineering
Department of Electrical and Computer Engineering

Supervisors:
Dr. Antonios Argyriou
Dr. Athanasios Korakis
Dr. Gerasimos Potamianos

July 2017

UNIVERSITY OF THESSALY

Abstract

School of Engineering
Department of Electrical and Computer Engineering

Master of Science

by Angeliki Kapodistria

In the last few years, wireless networks are becoming more and more popular. Along with them rises the support of real time and streaming applications over wireless networks. The need to understand these applications and to optimize their performance over wireless networks, motivated us to study the parameters that can cause problems over the implementation and usage. Jitter is among these variables and plays a determinant role in the design, management and implementation of the previous mentioned applications.

Based on the machine learning technique of Regression, we are trying to predict jitter for video application both in a wireless home network and in a wireless phone network. We use the Wireshark tool to capture packets of You Tube video sessions and we use these data to conduct our experiments by using two Regression algorithms, Boosting and Bagging. Our aim is to prove that jitter is predictable and that the results are useful, especially to those who design and implement wireless networks and video application on them.

Acknowledgements

Towards the end of this effort, I would like to express my sincere gratitude to those who contributed and helped me to complete this thesis.

First of all I would like to thank my supervisor, Assistant Professor Mr Argyriou Antonios, for his support and guidance and also for his cooperation during these years. Also, I would like to thank Assistant Professor Mr. Korakis Athanasios and Associate Professor Mr. Potamianos Gerasimos, for their participation in the committee and for the time they have dedicated to me.

Last, I owe a warm thank you to my parents, Dimitris and Spyridoula, my sister Konstantina, Haris and to all my friends for their love, their support, their help and their encouragement during this route.

Contents

List of Figures

List of Tables

To my family, Haris and my friends...

Chapter 1

Introduction

1.1 Wireless Networks

Wireless networks are considered an integral component of the communication field today. These networks enable various applications, such as the delivery of "streaming" audio and video, and real-time audio and video (e.g. Skype or video conference). In contrast with traditional network delivery, this class of applications requires a more timely transfer of messages from sender to receiver, in a continuous manner. In streaming applications, the user desires to watch the video or listen to the audio pretty much when it arrives. Real time applications require "deliver on time", and any delay, especially a huge one, renders the data useless even if delivered.

To support these applications, each network carries a sequence of packets from a source to one or more destinations. In interactive applications packet flows usually entail audio and/or video flows in both directions, while a streaming application is most likely to send packets in only one direction. All packets that transmit in a specific transport connection or media stream, are defined as *traffic flow* [1]. Understanding the characteristics of traffic flows is important for network design, traffic modeling, resource planning and network control.

Moreover, streaming and real-time applications have stringent requirements relative to Quality of Service (QoS). The following characteristics are attributed to a flow: reliability (or packet loss), bandwidth, delay and jitter [2].

Reliability

Packets may be lost during transmission, which entails retransmission. Data corruption is significant in wireless communication. However, the sensitivity of applications to reliability is not the same. Applications like streaming can sustain a certain percentage of packet loss.

Bandwidth

Bandwidth represents the capacity of a connection. Each application requires a sufficient bandwidth according to its throughput needs. For example, in video conferencing, which send millions of bits per second bandwidth requirements is the key to a successful deployment.

Delay

Delay, also called latency, is considered a characteristic of performance for networks and communications. As delay, is defined how long it takes for a bit of data to transmit across the network from a node to another or from an endpoint to another.
Delay is measured in fractions or multiples of seconds and is the sum of the below parts

- *Processing delay* - time a router needs to process the packet header and determine where to send it.
- *Queuing delay*- time spent by the packet in routing queues/buffer.
- *Transmission delay* - time needed to push data onto link, measured from the first bit of data until the last one to pushed onto the link. Defined as: $\frac{Size of packet}{Bandwidth}$.
- *Propagation delay* - time taken by data to traverse the link. Defined as: $\frac{Distance}{Speed of Light}$

Most of the times, processing delay is considered negligible. Figure 1.1, illustrates the parts of delay in an network.

Jitter

Except from delay, there is also jitter. According to [3], jitter is *the variability of packet delay within the same packet stream.* When a system is non deterministic, meaning that access to the network is not provided at fixed time intervals, delay varies and thus resulting jitter, as the difference in packet delay. In this thesis, we study the impact of jitter in video streaming, that has to do with huge amount of data. In such transmissions, delay

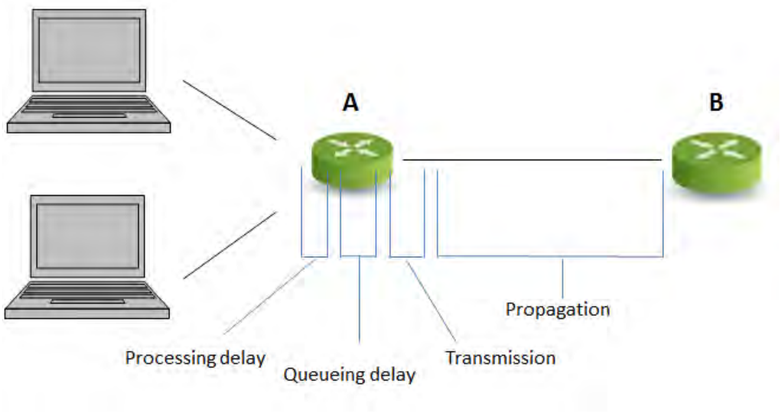

FIGURE 1.1: Delay parts in a network

jitter is unknown and time varying factor constrains and reduces degrees of freedom in system design [4].

Being more specific, our problem deals with transmission and propagation delay and jitter.

1.2 Motivation for this Thesis

In the last few years, data-oriented networks implemented on distributed platforms, in which real-time traffic is generated by audio or video applications, are more and more important. Moreover, wireless and mobility technologies for data transmission gets more attention. Traffic flows transmit over various network technologies, which are composed of traditional cable or fiber links.

This work is motivated by the desire to understand what the needs of those applications are, particularly in wireless environment. As referred above, the ability of wireless or mobile network to support real time and streaming applications depends on various parameters. Among them, jitter is the much thornier problem as it is the variable, or the congestion-based component of latency. While traditional popular applications, like telnet, FTP, email, web browsing, may be more tolerant to this effect, real-time traffic (like VoIP or videoconferencing) will be significantly degraded and is likely to become unacceptable.

To put it more correctly, the packets are transmitted in a continuous steady stream across the network and received at the other end, having very changeable packet inter-arrival intervals. Figure 1.2 illustrates this undesirable state. The mechanism to deal with this at the receiver is the playout delay buffer. The playout delay buffer stores the packets for a while, and provides them to be played back at the right time. There are

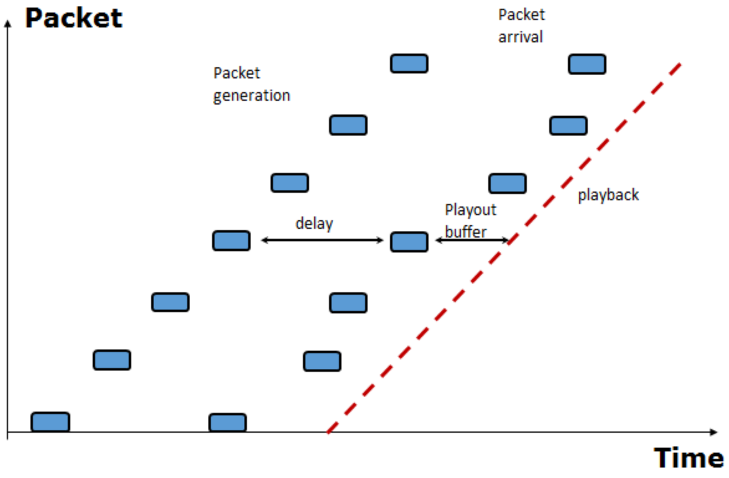

FIGURE 1.2: Packet Arrival with Jitter

limits on delay time, depending on application (e.g for audio application limit can be set on 150 ms for one-way [5]). Packets delayed too long in the network are not allowed to enter the jitter buffer and are dropped. As depicted in Figure 1.3, packets being generated at a steady rate, but arrival time varies due to jitter. Jitter is handled by the playout delay buffer.

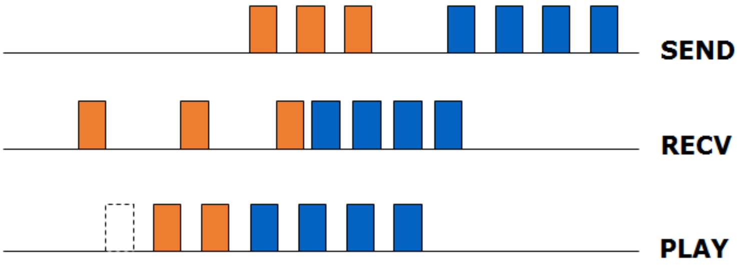

FIGURE 1.3: Adaptive Playout Delay

We are particularly interested in understanding the traffic flow behavior and to predict delay jitter in wireless or mobile networks using machine learning.

1.3 Purpose of this Thesis

As technology moves forward in a fast pace, wireless networks are more and more widespread for technical use, let alone in everyone 's daily life. Along with use of wireless networks, arises the use and the reputation of various applications and especially the ones using video. Thus, the ability to hold knowledge a priori for these networks and the applications on them would be a great asset for network providers and would offer better quality of service and user experience. In this thesis, the purpose is to extract knowledge from wireless networks' traffic flows, using machine learning techniques. This kind of knowledge is relative to network communication analysis and data mining from traffic flows of wireless networks.

To examine these issues, we have exploited the machine learning technique of Regression, and especially the algorithms of Boosting and Bagging, and our final aim is to decide, after a series of experiments, whether jitter is predictable or not for video application.

Initially, we collect data, from real wireless and mobile traffic flows, for the experiments using a network protocol analyzer, called Wireshark [6], and after we perform experiments in the data collected by taking advantage of the two algorithms, Boosting and Bagging. Our pursuit is to produce results that are almost equal to the real jitter of the application. These results will improve the design, management and implementation of wireless networks and also the implementation of applications that use wireless networks. Of course, the ulterior aim is all of the above to have as good as possible user experience as possible for the final user.

1.4 Structure

This thesis is organized in 5 chapters. In Chapter 2 and Chapter 3 we introduce basic concepts and issues related to terms and issues of knowledge extraction and machine learning techniques. More specifically, in Chapter 2 we introduce basic concepts of knowledge extraction and machine learning terms and after a detailed description of Regression technique along with the description of each algorithm used during the experiments. In it we are also cite the related work.

Chapter 4 is a detailed presentation of the experiments held, the data collection procedure, the filtering of the data and the application of the algorithms, and then we cite the results of those experiments. In Chapter 5, which is the last chapter of this thesis, we state general conclusions from our search and also propose and discuss future work.

Chapter 2

Background and Basic Concepts

2.1 Knowledge Extraction process

Knowledge extraction is the procedure of creating useful knowledge from structured (relational databases, XML) or unstructured (text, document, images) data sources. It is the process of automatically searching large volumes of data for patterns that can be considered knowledge about those data [7]. Knowledge discovery developed out from data mining and these two terms are closely related as far as methodology and terminology are concerned [8]. The procedure is also known as deriving knowledge from input data and is iterative and interactive.
Input data can be separated in the following categories:

- Databases: relational data, database, document warehouse, data warehouse
- Software: source code, configuration files, build scripts
- Text: concept mining
- Graphs: molecule mining
- Sequences: data stream mining, learning from time-varying data
- Web

Data Stream Mining is the process of knowledge extraction from continuous and rapid data records. Data streams are ordered sequences of instances and the challenge is that they can be read only once or a few times by data mining applications. The goal is to predict the class or the value of a new instance in the data stream, given some knowledge about the class membership or values of previous instances. To achieve this

goal, machine learning techniques are used so as to learn the prediction task from labeled examples in an automated fashion. Data stream mining is considered a subfield of data mining and machine learning as well [9], [10]. One of the most characteristic examples of data streams is the network traffic flows, which are studied in this thesis.
In Figure 2.1 below, we can see the whole procedure of knowledge extraction, along with the additional steps that this procedure includes, which are:

- Data preparation: collect required data and organize the required data.
- Data selection: from the whole data set, chose data that are useful to the problem examined.
- Data cleaning: remove data that may cause unwanted or even wrong results, i.e data that have values that diverge from the whole.
- Incorporation of previous knowledge: use knowledge acquired in previous step to optimize the current step.
- Proper interpretation of results of mining: interpret data in a way that has sense in the solution of the problem.

All these additional steps are essential since they ensure that useful knowledge is derived from data.

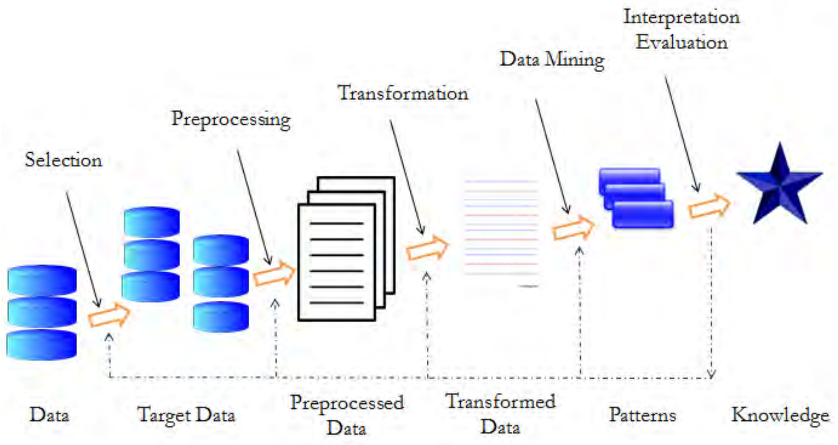

FIGURE 2.1: Knowledge Discovery Process

2.2 Machine Learning

Machine Learning is a field of Artificial Intelligence that deals with algorithms and methods enabling a machine to " learn", i.e. gain the ability to acquire further knowledge

by interacting with the environment in which it operates and also the ability to improve, through repetition, the way which performs an process.
Using it, enables the creation of customizable programs which operate based on the automated data analysis and not on the intuition of the programmers who made them. Arthur Samuel (1959), considers Machine Learning as a field of study which gives the computers the ability to learn without being programmed.
A more general definition has been given by Tom M. Mitchel (1997), according to whom *"A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P if its performance at tasks in T, as measured by P, improves with experience E"*.
For instance, delay prediction problem according to the definition stated above could be defined as:

- *Task T:* Predict delay time as close as possible to the real delay time.
- *Performance Measure P:* Delay that has been correctly predicted between packets' arrival.
- *Experience E:* Known delay time between arrival of packets.

Machine Learning algorithms are mainly classified into three categories, depending on the learning "signal" or "feedback" the learning system receives. These categories are [11]:

- *Supervised Learning:* algorithm constructs a function which depicts given inputs to known - desired outputs (training set), with aim to generalize the function for inputs with unknown output (test set).
- *Unsupervised Learning:* the model is constructed without giving labels to the learning algorithm, thus having to find in its own the structure in its inputs.
- *Reinforcement Learning:* the program interacts with a dynamic environment and "learns" a strategy of actions for a given observation, meaning it has to perform a certain goal. Also, it is provided with feedback in terms of reward or punishment, as navigating through the problem space.

Our search will be based in supervised learning. In supervised learning, a labeled training set, a known dataset, is used to make predictions - build up the system model. Training set is composed of input data and response values and from it the algorithm use, seeks to build a model that can predict response values of a new set. What the model does is to represent the learned relation between input, output and system parameters. To validate the model created, a test set is used. The use of larger training sets can lead to higher predictive power, which can be generalized for new sets [12].

Supervised Learning algorithms can be separated in two categories according to their type of results:

- *Classification*: for categorical response values, in which data can be separated into specific "classes"
- *Regression*: for continuous response values

Classification is the procedure of mapping an object in a class, among different predefined. Input data for the classification are a collection of records. Each record, also known as sample, consists of a tuple (x, y), where x is the set of attributes and y a label for the class. We can define classification as the procedure of learning a target function f to map each attribute set x in a predefined class with label y.

Regression estimates relationships among dependent and independent variables, in order to discover how the changes in the value of the independent variable affect the dependent.

Our search has to do with continuous values, so we will use Regression, which is described in details below.

2.2.1 Regression

Regression is a process for estimating relationships among variables, dependent and independent. It models and analyzes variables in order to find how the value of a dependent variable changes when one of the independent variables is varied, with other independents held fixed. Mainly, regression estimates the conditional exception of the dependent variable given the independent, which is the average value of the dependent value when the independents are fixed. In a few words, regression allows making prediction from data by learning the relationship between features of data and some - observed, continuous valued response.

Common regression algorithms are:

- Linear regression
- Nonlinear regression
- Generalized linear models
- Decision trees

- Neural networks

In this thesis, we focus on decision trees and we use the algorithms of Boosting and Bagging for our experiments.
Decision trees, is a predictive learning approach which aims to create a model that predicts the value of a target variable based on several input variables. Except regression trees that we use, there also classification trees when the outcome that they predict is a class to which the data belongs.
Boosting and Bagging are considered as machine learning ensemble meta-algorithms that construct more than one decision tree. The term ensemble is used for methods that use multiple learning algorithms to obtain better predictive performance than the one that could be obtained from any of the constituent learning algorithms alone. The term meta-algorithm describes high level procedures, designed to find, generate or select a heuristic that provides a good solution to an optimization problem [13].

Boosting

Boosting is an ensemble meta-algorithm which takes a weak learner and converts it into a strong one. Kearns and Valiant (1988, 1989), [14], [15], wondered if a set of weak learners can be combined into a strong one and Schapire (1990), answered affirmative to that question [16], thus leading to the development of boosting. A weak learner is a classifier which is only slightly correlated with true classification, i.e. labels examples better than random guessing, whereas a strong learner is a classifier that is arbitrarily well correlated with the true classification.
Boosting methods focus on producing series of classifiers. In order to choose the training set, each member is based on the performance of the earlier classifier of the series. Since examples that were incorrectly predicted are chosen more often than those correctly predicted, boosting attempts to produce classifiers whose results are better for examples with poor performance at current ensemble.
Generally, boosting has the following steps: Booster is provided with a set of labeled, training examples $(x_1, y_1), ..., (x_n, y_n)$, where each instance x_i is associated with a label y_i; in our problem, x_i is the arrival time of each packet and y_i is the predicted delay between packets' arrival. O each round $t = 1, ..., T$, the booster devises a distribution D_t over the set of examples, requesting a weak hypothesis h_t with error ϵ_t with respect to D_t, i.e. $\epsilon_t = Pr_{i \sim D_t}[h_t(x_i) \neq y_i]$. Hence, the relative importance for each example, for the current round, is specified by the distribution D_t. After T rounds, the booster has to combine the weak hypotheses into a single prediction rule. [17]. The exact algorithm used in our case will be presented in the following chapter.

Bagging

Bagging, comes from **B**ootstrap **Agg**regat**ing**, is also a machine learning ensemble meta-algorithm, which combines classifications of randomly generated training sets so as to decrease variance and avoid overfitting, leading to accuracy improvement. L. Breiman (1994), defined bagging as *the procedure of generating multiple version of a predictor and use these to get an aggregated predictor.* Depending if it is used for classification or regression, the aggregation does plurality when predicting a class and averages over the versions when predicting a numerical outcome.

Data of form $(y_n, x_n), n = 1, ..., N$ consist a learning set $\mathcal{L}$, with y's either being class labels or numerical responses. To form a predictor $\phi(x, \mathcal{L})$, a procedure that uses this learning set is used; if x is the input, y is predicted by $\phi(x, \mathcal{L})$. Given a sequence of learning sets $\mathcal{L}_k$, with N observations consist each of them from the underlying distribution as $\mathcal{L}$, we have to use the $\mathcal{L}_k$ to get a better predictor than the one of the single learning set $\phi(x, \mathcal{L})$. The only limitation is that we are only allowed to use the sequence of predictors $\phi(x, \mathcal{L}_k)$.

If y is numerical, $\phi(x, \mathcal{L})$ is replaced by the average of $\phi(x, \mathcal{L}_k)$ over k, i.e by $\phi_A(x) = E_{\mathcal{L}}\phi(x, \mathcal{L})$ where $E_{\mathcal{L}}$ is the expectation over $\mathcal{L}$ and A in ϕ_A is aggregation. If $\phi(x, \mathcal{L})$ predicts a class $j \in 1, ..., J$, then aggregating the $\phi(x, \mathcal{L}_k)$ is done by voting. Let $N_j = \# \{k; \phi(x, \mathcal{L}_k) = j\}$ and take $\phi_A(x) = argmax_j N_j$.

However, having replicates of $\mathcal{L}$ is not always possible and we can have only a single learning set $\mathcal{L}$. In situations like these, the process is done by taking repeated bootstrap samples $\mathcal{L}^{(B)}$ from $\mathcal{L}$ and form $\phi(x, \mathcal{L}^{(B)})$, thus leading to ϕ_A. If y is numerical take, ϕ_B as $\phi_B(x) = \alpha v_B \phi(x, \mathcal{L}^{(B)})$. If y is a class label, then let the $\phi(x, \mathcal{L}^{(B)})$ vote to form $\phi_B(x)$. This procedure is called bootstrap aggregating.

2.3 Related Work

Delay is a major problem in our everyday life, from transportation to communication, and for this reason research in this field is notable. Researchers try to estimate or even predict delay in flights [18] and transportation in general and more specifically and technologically oriented in networks and in real time applications, such as video streaming, online gaming, etc.

In their work, Hongyan et al. [19], are trying to predict delay in internet using autoregressive model and neural networks. Having analyzed communications features of Internet, they use the autoregressive model and adaptive linear neural network to predict the uncertain delay. Their simulations for both methods result to promising ways of delay prediction and especially statistics for neural networks are better than the ones for autoregressive model.

In [20], Liu et al. are proposing a method that estimates latency given 3D latency matrices sampled over time. Their distance - feature decomposition algorithm, decomposes latency matrices into a distance component and a network feature component and furthermore, takes advantage of the structured pattern inherent in 3D sampled data so as to increase accuracy in estimation. Compared to other latency prediction techniques it outperforms them.

Xiong, Wu and Jia in their research, [21], made it more specific and aim to predict delay for real time video transmission over TCP. Their approach is based in a stochastic, prediction model which predicts the sending delays of video frames. As a result to this, they propose an adaptive transmission scheme, for real time video, which dynamically adjusts video frame rate and play out buffer size according to network bandwidth that is available. Nunes et al., [22], have exploited the machine learning algorithms so as to estimate end to end route trip time (RTT) and specifically the technique of Experts Framework. In their approach, each "expert" of the several ones available, guesses a fixed value and the weighted average of these experts estimates the RTT. Weights are updated after every RTT measurement, as the difference between the estimated and actual RTT. Their proposal adapts quickly to changes in RTT, shows reduction in retransmitted packets and increase in throughput. Also they were able to achieve through experiments, higher accuracy for the machine learning technique among other well known estimators such as Eifel RTT Estimator and standard TCP. Although all of the above researchers try to approach delay estimation problem, even [18] for flights, either they do not use any machine learning method or algorithm, or they approach a very generic problem or a different problem, such as Hongyan et al. [19], who despite using machine learning techniques they try to approach the delay in Internet in general. Simmilar to them, Nunes et al. [22], use a machine learning algorithm and try to estimate RTT which is more specific, however remains a different problem from the one investigated in this thesis.

The upcoming chapter, is a detailed presentation of the work we have made and all the algorithms and techniques used and described in general previously.

Chapter 3

Delay - Jitter Prediction

3.1 Introduction

As traffic flows travel through a wireless network, many problems can occur. Information loss, congestion and delay or jitter are among the major problems. Depending the application we use, each problem is more or less sensitive.

When it comes to video applications, delay and jitter play a determinant role in the design, management and evaluation of the application and the quality of service provided. As a result, the ability to predict delay and jitter would be of great importance as far as the design, management and implementation of the application are concerned.

Since video applications along with wireless networks exist in our daily lives, the ability to optimize the problem of transmitting video over wireless networks is of great importance. It will give network providers the ability to improve their quality of service and it will offer final users the opportunity to use the application they want immediately and effortlessly. More specifically, the receiver must receive frames at time and any late frames can cause problems and gaps in the reconstructed video.

This need urged us to investigate the problem of jitter prediction and its appliance in streaming and real - time applications on wireless environment. In addition to the above, except from improving Quality of Service and user experience, the ability to predict jitter can improve communications, since knowing it a priori can decrease the possibility of transmission errors that it causes.

Jitter is caused by a variety of mechanisms, such as systems phenomena, data - related phenomena or random noise phenomena. More specific causes of jitter could be: channel loss and reflection, random noise, crosstalk or inter-symbol interference.

Consequences that jitter has, vary from a simple delay in communication transmission, for example video could not load, live streaming is buffering, to more serious problems such as transmission errors and communication loss.

Having jitter is not always bad and acceptable levels of jitter exist. The important is to keep jitter in low levels so as to maintain steady data streams. Systems that are affected by jitter have buffers to handle it. In streaming media such as online video, jitter buffers work properly since they are not required to disseminate real-time data as in Voice over IP. On the other hand, applications such as Voice and Video over IP, require lower buffering and better network performance, thus lower jitter. Jitter is the the absolute value of the difference between the forwarding delay of two consecutive received packets belonging to the same stream. To calculate jitter, the following four parameters are required:

- Transmit time of the first packet in the pair.
- Receive time of the first packet in the pair.
- Transmit time of the second packet in the pair.
- Receive time of the second packet in the pair.

So, if we have packets A and B that are consecutively sent in the same data stream, jitter could be expressed as:

$jitter = |(Tx_B - Tx_A) - (Rx_B - Rx_A)|$

According to Figure 2.1, to collect data we use Wireshark, a network protocol analyzer tool, to perform traffic measurements by means of passive network monitoring. Passive measurement do not produce additional traffic or modify the traffic that is already on the network.

Wireshark is a free and open source network tool used to capture and analyze packets. It provides the collected data about packets as detailed as possible. It is used to monitor “what is going on inside a network cable” [6] and assists users in troubleshooting, analysis, software and communication protocol development and education. It has graphical interface, many more filtering options, and “understands” the structure of different networking protocols. Thus, it is capable to display the contents of all fields within a protocol message.

3.2 Prediction

All of the above mentioned researches where in the same field but they were not as specific as our work. We have focused in jitter, since we try to predict delay within the same packet stream of a video and not the general delay of the network used over different applications. Aim of this thesis is to determine if jitter for video application can be predicted and which method is the appropriate for it. In contrast with other approachs, we use a machine learning technique, in particular we apply Regression, and two different algorithms - Boosting and Bagging, in real collected data and we try to evaluate their results so as to decide whether jitter in video is predictable or not.

One of the advantages of using machine learning is that we avoid the use of parametric function. Moreover, it allows us to modify our experiments as we want and towards the variable we want to examine at a given time. Since jitter has to do with continuous values, we use Regression as already mentioned and specifically we use Regression Trees to evaluate our data. Since we are examining jitter in video applications, the data we use, are real data, captured via Wireshark from You Tube and analyzed with the use of Matlab. We consider two different target networks; in the first, data come from a wireless home network (ADSL connection) and in the second, from a mobile phone network (3G connection). Thus, we examine the behavior of different traffic flows and evaluate the algorithms' performance over various types of wireless networks.[1]

The problem we examine can be split in two subproblems. The first one is to collect data and organize them, so as to be effectively handled by the algorithms used. The second one is to apply the algorithms in the data so as to decide if jitter is predictable. Both of the algorithms, use multiple learning to obtain better performance but differ in the way the build up their result. Except the comparison with the real results, we also compute mean square error to evaluate their performance. Details about the algorithms used are described in the below section.

3.2.1 Algorithms

Boosting is a sequential ensemble which add new models to the previous ones and aims to decrease bias. On the other hand, Bagging is a parallel ensemble and each new model is built independently and the procedure aims to decrease variance.

In the experiments we conducted the Bagging algorithm used was the one described in details in Chapter 2. For the Boosting algorithm though, we used a version called Least Square Boost (LS Boost) and the exact algorithm is [23]:

[1] More details about data used and the set of experiments, will follow in the next Chapter

Algorithm 1 : LS Boost

```
$F_0(x) = \bar{y}$
for $m = 1$ to $M$ do:
    $\tilde{y}_i = y_i - F_{m-1}(x_i), i = 1, N$
    $(\rho_m, a_m) = \arg\min_{a,\rho} \sum_{i=1}^{N} [\tilde{y}_i - \rho h(x_i; a)]^2$
    $F_m(x) = F_{m-1}(x) + \rho_m h(x; a_m)$
End for
End Algorithm
```

3.3 Jitter Prediction Model

In our approach we combined machine learning techniques over data that we collected, in order to predict jitter. The data sets that we used have been filtered, so as to keep only the necessary information for the experiments and to be more precise and accurate. By filtered, we mean that from the data captured, we used in our experiments only the attributes that the algorithms needed, i.e. the serial number of the packet and the time of its arrival.

Moreover, in our approach the algorithms used were the ones provided from the Matlab platform and we have not used a variation of them to our experiments, since our aim in first place was to examine whether our hypothesis over the prediction of jitter is applicable or not. Another important feature that both algorithms have, is that they do not demand a lot of information to operate over the data. As already mentioned, from the data sets collected we used only two attributes to train and test them. Furthermore, the ability to adapt and operate with either larger or smaller data sets, was another strong asset for both algorithms, which in our case was very useful due to lack of system resources (mainly memory resources).

To the best of our knowledge, this is the first approach in terms of data used, applied algorithms and relative applications which tries to handle the jitter problem. Being more specific, we have tried to predict jitter for a specific class of applications, video, whereas related bibliography focused on general delay prediction over Internet in general, [19], or used a different approach, [21]. In our approach, we used and compared two machine learning algorithms, in different line with others, Boosting and Bagging, for a problem that was also not present in any search. Last, but not least, we have chosen to use streaming logs from You Tube sessions for the experiments, so as to have the opportunity to handle them according to the requirements of the algorithms and also to have the ability to use the exact data that fit in our approach to the problem of jitter in video.

Chapter 4

Experimental Setup and Evaluation

In this thesis, we present a measurement study based on large dataset of streaming traffic during a You Tube video session. As a streaming session initiates, Wireshark captures its packets. For this study we have used data from a wireless home network and from a wireless phone network. We collected a large number of datasets from various traffic flows in these networks. Here, we analyze and present some of them.

4.1 Experiments' Set up

Through the experiments, our aim was to combine results from many weak learners into one, high - quality ensemble predictor. For this reason, we used Matlab (version 2014a) as a platform for our experiments. To create, and train, an ensemble we used the *fitensmble* function : $ens = fitensmble(X, Y, model, numberens, learners)$

- X is the matrix with the input data. Each row represents and observation, each column represents a predictor variable.
- Y is the responses' vector and has the same number of observations as X has.
- model is the a character vector which represents the type of ensemble.
- numberens is the number of weak learners in *ens* from each element of *learners*. This means that number of elements in *ens* in *numberens* times the number of elements in *learners*.

- learners is a character vector, template or cell array that represents the weak learner used.

Since we are using Regression, the type of ensemble, i.e. model, was *LS Boost* for the Boosting and *Bag* for the Bagging and the learner used was *Tree.*
The ensemble can be graphically represented in Figure 4.1:

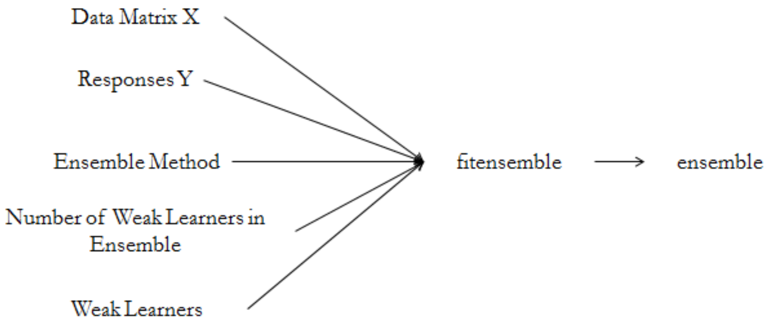

FIGURE 4.1: Fit ensemble

First, a training set was used to build the prediction model and after, the model created, was used in the test set so as to predict jitter. In the upcoming section there is a detailed description of the training sets and test sets, along with the conditions of the experiments.

4.2 Training Sets and Test Sets

Since we are focusing on jitter in video application, all of our data are from video streams via You tube. Six different training sets were created to train both the Boosting and the Bagging predictor. Sets are packets coming from traffic of a wireless network with speed 4Mbps for the ADSL connection and sets of packets coming from traffic of a wireless mobile phone network with 3G connection. All data have been collected with the use of Wireshark, which is a packet analyzer tool for recording incoming, out coming traffic via an interface, such as an Ethernet card, in real time with aim the further analysis of the packets [6].
Wireshark collects packets in real time and display their attributes, such as source IP, destination IP, protocol, size of packet and other information in human-readable format. We are interested in video streaming data flows, consisted of sequences of packets and the flow attributes we focus on are serial number and the time of arrival of each packet, that deal with network traffic in real time.
Following the model of $fitensemble$ function that described previously, our data matrix

X has two columns, one for the serial number of the packet and one for the time of arrival, response vector Y has the jitter between each packet arrival, *model* is either Boosting or Bagging, *learner* is Tree and after numerous times of repetitions for each experiment we have concluded that $numberens = 300$. In response vector Y, jitter is the difference between the arrival times of the packets.

For the wireless home network, size of training set varies from 64,000 packets to 144,000 packets and for the mobile phone network sizes are from 14,000 packets to 63,000 packets. The only reason that we used "smaller" training sets in the second case, was the quality of the connection which did not allow us to capture bigger data sets. At this point we should also mention that due to memory restrains, we had not the ability to execute experiments with larger training data sets and this case in left as a future work.

To evaluate the performance of the predictors, we used the training sets as test sets, with the use of the *cvpartition* function: $cv = cvpartition(n, \text{“}HoldOut\text{”}, p)$. This function, creates a random partition for holdout validation on n observations. This partition, divides the observations into a training set and a test, or holdout, set. Parameter p is a scalar, we used the default value $p = 0.1$, which means that if $0 < p < 1$, *cvpartition* randomly selects approximately $n * p$ observations for the test set.

Experiments held out with the same parameters for both algorithms and for each kind of wireless network used, thus meaning that the only values that changed each time was the type of predictor and the training set.

4.3 Evaluation

To measure the accuracy of the predictors we used two functions, *ResubstitutionLoss* and *Loss*. These functions are metrics, showing how accurate is the prediction. The first function, *ResubstitutionLoss*, computes the loss in the training set, whereas *Loss* computes the loss in the test set.

Both *resubLoss* and *loss* use mean square error to compute the error:

$$mse = \frac{\sum_{j=1}^{n} w_j (f(x_j) - y_j)^2}{\sum_{j=1} n w_j},$$

where:

- n, number of rows of data
- x_j, the j-th row of data
- y_j, the true value for x_j
- $f(x_j)$, the predicted response for the x_j,
- w, a vector of weights, which by default are equal to 1

In tables, Table 4.1, Table 4.2, is represented briefly the **Mean Square Testing Error** for the two algorithms, in each of the network used and for each training set. For the wireless phone network we can see that in four out of six training sets, Boosting has better performance, i.e smaller error, from Bagging. In the mobile home network however, it seems that the algorithms have similar performance.

Wireless Phone Network		
-	Boosting	Bagging
Training Set 1	0.033653	0.024535
Training Set 2	0.014833	0.015923
Training Set 3	0.017394	0.025037
Training Set 4	0.011386	0.015880
Training Set 5	0.014457	0.020889
Training Set 6	0.049391	0.031837

Table 4.1: Mean Square Testing Error for all Training Sets

Wireless Home Network		
-	Boosting	Bagging
Training Set 1	0.004214	0.006852
Training Set 2	0.001852	0.002394
Training Set 3	0.009490	0.032506
Training Set 4	0.009905	0.003389
Training Set 5	0.006213	0.005831
Training Set 6	0.000423	0.000394

Table 4.2: Mean Square Testing Error for all Training Sets

Except from the tables, Table 4.1, Table4.2, we have also graphic representations of our experiments that helped us to reach in some conclusions.

In the following figures, we compare the actual values of the jitter with the predicted ones and the loss for the training and the test sets for three different training and test sets for each network. For Figures 4.2 - 4.7 data from a wireless phone network were used, and specifically training sets 1, 3 and 5. Applying Boosting and Bagging algorithms, we attempt to predict jitter. As you may see, both algorithms perform similarly and their performance is enough satisfying, since the prediction curve (red line) approximates the real data perfectly while boosting prediction is more sensitive to data changes. In addition, we obtain that none of them achieves to fit accurate in real data, particularly in peaks. Accordingly, Figures 4.8 - 4.13 refer to wireless home network and specifically training sets 2,4,6. In this case, results are once more satisfying and we still observe that algorithms do not fit accurate in peaks.

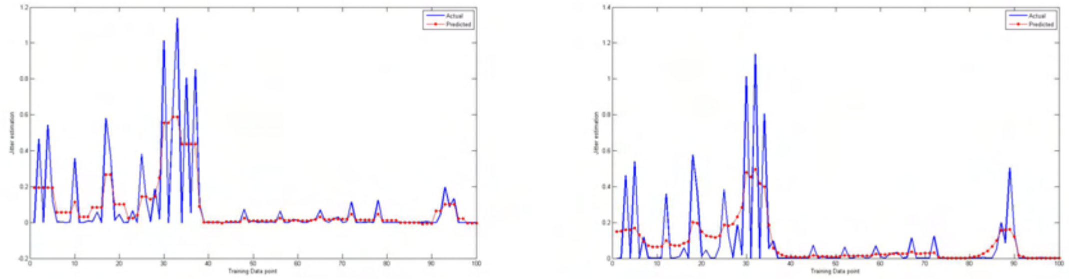

Figure 4.2: Actual - Predicted Data Test Set 1: Boosting vs Bagging, Wireless Phone Network

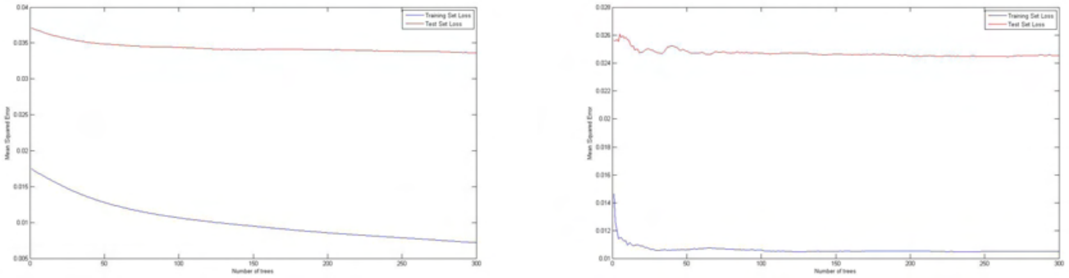

Figure 4.3: Training vs Test Set Loss - Set 1, Wireless Phone Network

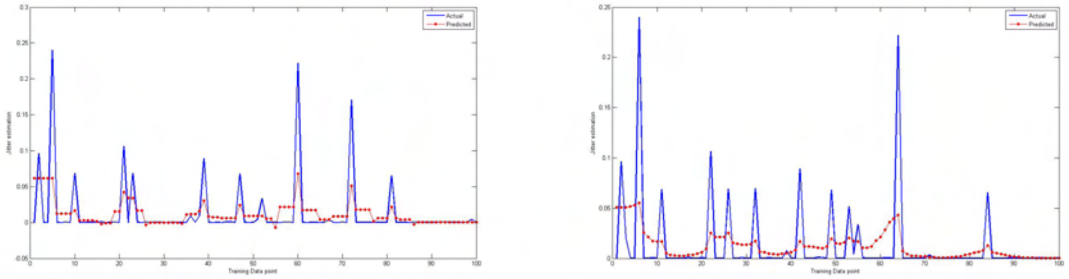

Figure 4.4: Actual - Predicted Data Test Set 3: Boosting vs Bagging, Wireless Phone Network

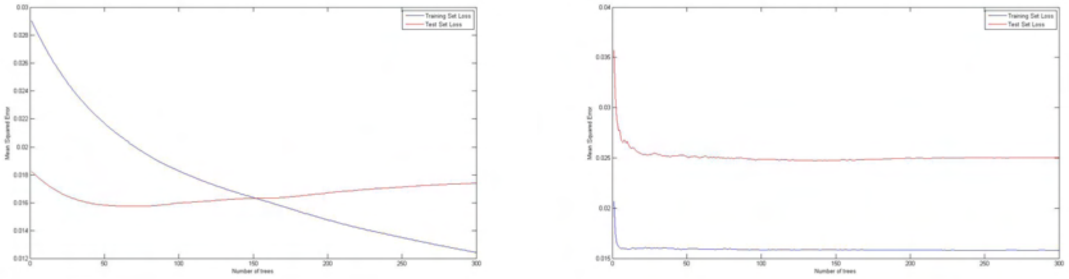

Figure 4.5: Training vs Test Set Loss - Set 3, Wireless Phone Network

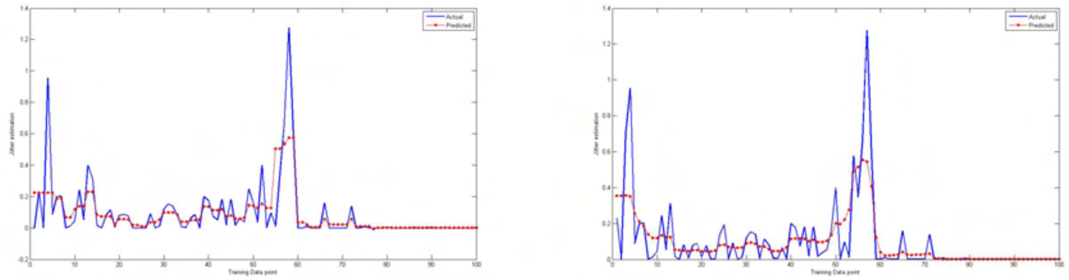

FIGURE 4.6: Actual - Predicted Data Test Set 5: Boosting vs Bagging, Wireless Phone Network

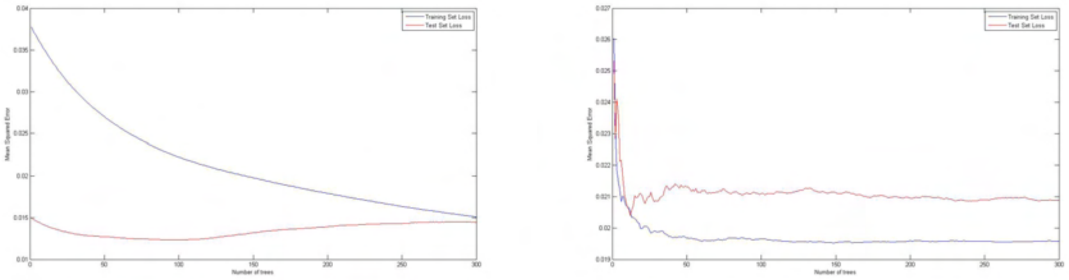

FIGURE 4.7: Training vs Test Set Loss - Set 5, Wireless Phone Network

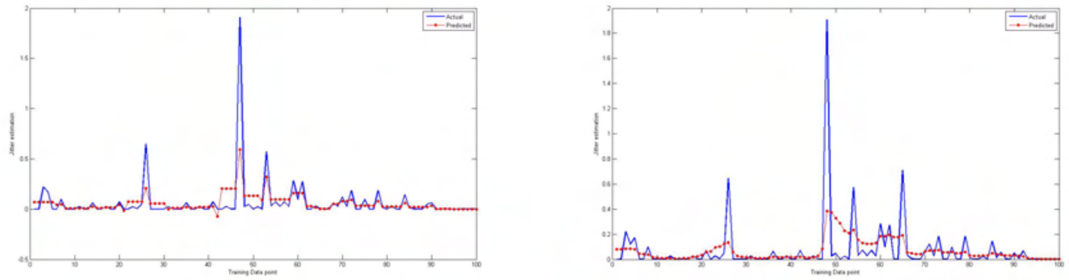

FIGURE 4.8: Actual - Predicted Data Test Set 2: Boosting vs Bagging, Wireless Home Network

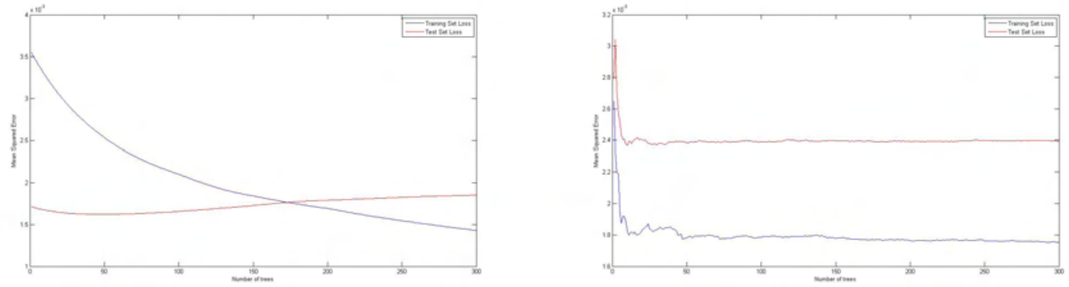

FIGURE 4.9: Training vs Test Set Loss - Set 2, Wireless Home Network

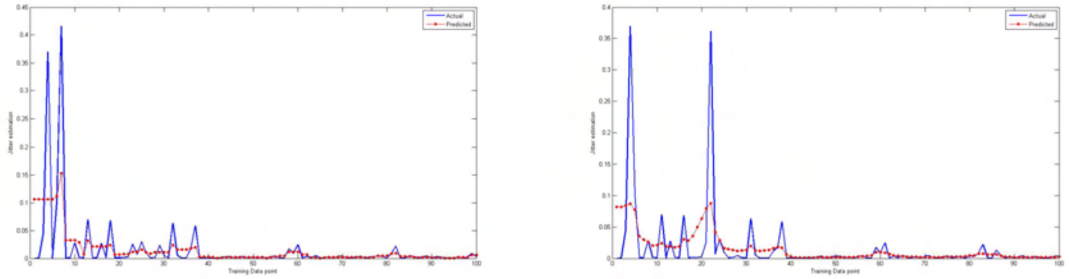

FIGURE 4.10: Actual - Predicted Data Test Set 4: Boosting vs Bagging, Wireless Home Network

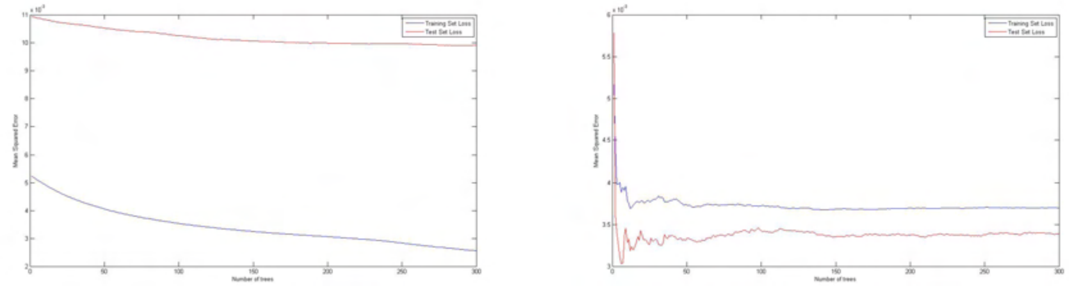

FIGURE 4.11: Training vs Test Set Loss - Set 4, Wireless Home Network

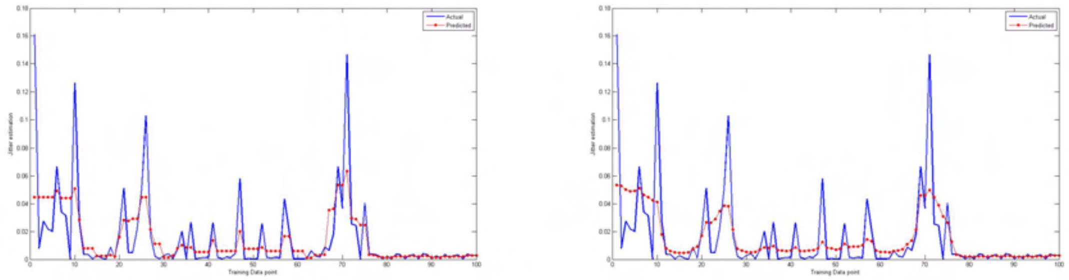

FIGURE 4.12: Actual - Predicted Data Test Set 6: Boosting vs Bagging, Wireless Home Network

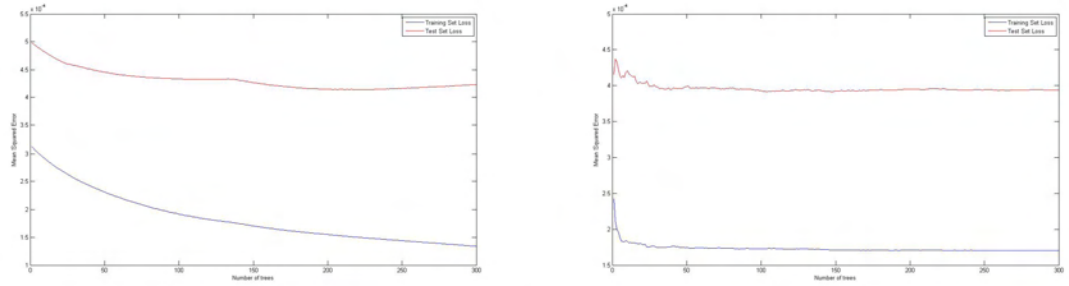

FIGURE 4.13: Training vs Test Set Loss - Set 6, Wireless Home Network

Comparing results from Figure 4.2, Figure 4.4, Figure 4.6, Figure 4.8, Figure 4.10, Figure 4.12, we can see that our algorithms approach actual results in a very good percentage, except from cases where jitter is higher than the usual. Generally cases like these are known as outliers and are a common problem in data mining. On purpose we did not eliminate these incidents by filtering our data, because our aim was to examine the performance of both algorithms in cases like those.

Notable are also the results in Figure 4.3, Figure 4.3, Figure 4.5 , Figure 4.7, Figure 4.9, 4.11, Figure 4.13, where training and test set loss are computed and compared. The first observation is that after a certain number of learning cycles, both losses tend to stabilize. There were also cases, as in Figure 4.11 in which test set loss was lower than that of the training set. This means the the learner was able to create the ensemble with decreased variance and as a result mean square error was lower in the test set than the one of the training set. Issue is connected with the use of the *cvpartition* function, since data are chosen randomly for the test set.

Chapter 5

Conclusion and Future Work

Aim of this thesis was the prediction of jitter in video applications in wireless and mobile networks, based on machine learning techniques. Whereupon, we present results and conclusions of our search and we also provide directions for future work.

5.1 Conclusions

This thesis, focuses on jitter prediction using a machine learning technique, Regression analysis, and two meta-algorithms, Boosting and Bagging. Jitter in wireless networks can not be linear represented with the use of a parametric function and thus its computation is more complex. Machine learning provides the opportunity to avoid the use of parametric function in complex cases like this and allows flexibility, efficiency and effectiveness on data handling and problem's solution as a consequence. To examine the problem of jitter prediction we apply two different algorithms - Boosting and Bagging. The basic idea is that we have a set of experiments with different training set each time, which is also used as a test set, and by applying the chosen algorithms alternately, we compare their results with the real ones that we already have.We propose and implement test cases where a known algorithm is applied in a test set of specific data, packets derive from video application, so as to decide if jitter is predictable. Data filtering is needed so as to conduct the experiments, since additional information is not necessary in our approach to the problem.
Different experiments with various, mainly in size, data sets helped us conclude that jitter can be predicted for video applications. As far as the performance of the algorithms is concerned, the general conclusion is that both of the methods, have equally satisfying results regarding the delay - jitter prediction. According to Table 4.1, Table 4.2, both of them manage to maintain Mean Square Error low and according to Figure 4.7, Figure

4.11 both of them when computing Test Set loss can achieve better results than the ones in Training Set, i.e. test set loss is less than training's set loss. Nevertheless, this needs further investigation, since using Matlab's *cvpartition* command sets aside 90 percent of the data for training randomly, thus making the whole procedure stochastic.

Another point that needs to be mentioned is that when using larger data sets, performance was better for both algorithms. Unfortunately, due to memory issue this case was not investigated thoroughly and is left as future work.

5.2 Future Work

Accuracy of learners relates to size of training sets, as already mentioned. It would be interesting to conduct experiments with bigger data sets, so as to evaluate the performance of the algorithms under these circumstances. More data, means better training for the algorithm, thus leading into more accurate results.

In our approach, we have not examined the case where other users exist in the network. Our data where captured while only our device was using the network, minimizing thus the interference. In real time applications however, multi user applications and as a result interference is a major problem and should be taken in mind in future experiments.

Nowadays, quality of connection improves in a fast pace. However there places that connection is still poor, for example metro stations, dense built areas, etc. Both cases, better quality of connection and poorer should be taken into account for future work. Generally speaking, the quality of the connection in our cases can be considered adequate. In order for the results to be more accurate however, there should be experiments that use data from networks with various quality of connection.

Key role in our search plays the video application that we choose to investigate. An important case would be the one where jitter is being predicted for video streaming applications, i.e. live streaming, a case in which we observe both delay and jitter in high levels. Interesting would also be the case of jitter prediction in online gaming applications and generally the expansion in more applications, irrespective to video.

Bibliography

[1] Jarno Rajahalme, Shane Amante, Sheng Jiang, and Brian Carpenter. Ipv6 flow label specification. 2011.

[2] B. Forouzan. *Data Communications and Networking: Fifth Edition*. McGraw-Hill Higher Education, 2012. ISBN 9780077445393. URL https://books.google.gr/books?id=yT80AAAAQBAJ.

[3] James F. Kurose and Keith W. Ross. *Computer Networking: A Top-Down Approach (6th Edition)*. Pearson, 6th edition, 2012. ISBN 0132856204, 9780132856201.

[4] John G Apostolopoulos, Wai-tian Tan, and Susie J Wee. Video streaming: Concepts, algorithms, and systems. *HP Laboratories, report HPL-2002-260*, 2002.

[5] Rec ITU-T and I Recommend. G. 114. *One-way transmission time*, 18, 2000.

[6] Angela Orebaugh, Gilbert Ramirez, and Jay Beale. *Wireshark & Ethereal network protocol analyzer toolkit*. Syngress, 2006.

[7] William J Frawley, Gregory Piatetsky-Shapiro, and Christopher J Matheus. Knowledge discovery in databases: An overview. *AI magazine*, 13(3):57, 1992.

[8] Usama Fayyad, Gregory Piatetsky-Shapiro, and Padhraic Smyth. From data mining to knowledge discovery in databases. *AI magazine*, 17(3):37, 1996.

[9] Mohamed Medhat Gaber, Arkady Zaslavsky, and Shonali Krishnaswamy. Mining data streams: a review. *ACM Sigmod Record*, 34(2):18–26, 2005.

[10] Brian Babcock, Shivnath Babu, Mayur Datar, Rajeev Motwani, and Jennifer Widom. Models and issues in data stream systems. In *Proceedings of the twenty-first ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, pages 1–16. ACM, 2002.

[11] Stuart Russell and Peter Norvig. Artificial intelligence: a modern approach. 1995.

[12] Mohammad Abu Alsheikh, Shaowei Lin, Dusit Niyato, and Hwee-Pink Tan. Machine learning in wireless sensor networks: Algorithms, strategies, and applications. *IEEE Communications Surveys & Tutorials*, 16(4):1996–2018, 2014.

[13] David Opitz and Richard Maclin. Popular ensemble methods: An empirical study. *Journal of Artificial Intelligence Research*, 11:169–198, 1999.

[14] Michael Kearns. Learning boolean formulae or finite automata is as hard as factoring. *Technical Report TR-14-88 Harvard University Aikem Computation Laboratory*, 1988.

[15] M. Kearns and L. G. Valiant. Crytographic limitations on learning boolean formulae and finite automata. In *Proceedings of the Twenty-first Annual ACM Symposium on Theory of Computing*, STOC '89, pages 433–444, New York, NY, USA, 1989. ACM. ISBN 0-89791-307-8. doi: 10.1145/73007.73049. URL http://doi.acm.org/10.1145/73007.73049.

[16] Robert E Schapire. The strength of weak learnability. *Machine learning*, 5(2): 197–227, 1990.

[17] Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. In *European conference on computational learning theory*, pages 23–37. Springer, 1995.

[18] Juan Jose Rebollo and Hamsa Balakrishnan. Characterization and prediction of air traffic delays. *Transportation research part C: Emerging technologies*, 44:231–241, 2014.

[19] Li Hongyan, Wang Hong, and Gui Chao. Internet time-delay prediction based on autoregressive and neural network model. In *Communications, Circuits and Systems Proceedings, 2006 International Conference on*, volume 3, pages 1758–1761. IEEE, 2006.

[20] Bang Liu, Di Niu, Zongpeng Li, and H Vicky Zhao. Network latency prediction for personal devices: Distance-feature decomposition from 3d sampling. In *Computer Communications (INFOCOM), 2015 IEEE Conference on*, pages 307–315. IEEE, 2015.

[21] Yonghua Xiong, Min Wu, and Weijia Jia. Delay prediction for real-time video adaptive transmisson over tcp. *Journal of multimedia*, 5(3):216–223, 2010.

[22] Bruno AA Nunes, Kerry Veenstra, William Ballenthin, Stephanie Lukin, and Katia Obraczka. A machine learning approach to end-to-end rtt estimation and its

application to tcp. In *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, pages 1–6. IEEE, 2011.

[23] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.