Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

MACHINE LEARNING TECHNIQUES FOR NETWORK ANALYSIS

by
Irfan Lateef

The network's size and the traffic on it are both increasing exponentially, making it difficult to look at its behavior holistically and address challenges by looking at link level behavior. It is possible that there are casual relationships between links of a network that are not directly connected and which may not be obvious to observe. The goal of this dissertation is to study and characterize the behavior of the entire network by using *eigensubspace* based techniques and apply them to network traffic engineering applications.

A new method that uses the joint time-frequency interpretation of *eigensubspace* representation for network statistics as features for identification and tracking traffic flows based on the link level activity is proposed. *Eigencoefficients* (frequency domain feature set) and *eigencomponents* (time domain features) are jointly utilized to quantify their combined significance on the representation of each link data (each component of the link traffic vectors) in the *eigensubspace*.

Several experiments are conducted using the joint time-frequency method to analyze the traffic data obtained from the Internet2 network. It is shown that the analysis with link-level resolution brings advantages for network traffic engineering applications. Specifically, this technique is applied to two scenarios: to identify large traffic flows and anomalous events in the network.

Furthermore, machine learning methods are investigated to identify network paths using *eigenanalysis* of link statistics as the feature set. The merit of this method is validated by applying the technique on various network experiments. *Eigenvectors* and *eigenflows* in the subspace are jointly used as factors (features) for linear regression to forecast the network link traffic. It is demonstrated that the

eigensubspace based autoregressive order two, AR (2), predictor is superior to the time-domain based predictor to forecast the link level traffic of a network.

The unique contribution of this dissertation is using joint time-frequency interpretation of *eigensubspace* as features for identification of patterns and anomalies in conjunction with machine learning methods to automate the process and improve the accuracy of the method. This idea is not only applicable to the network analysis as demonstrated in this dissertation, but also applies to various fields of knowledge including medicine, finance and engineering. All of these fields have very large data sets in time domain, as well as complex patterns and relationships that exist among and are not discernible to human mind. This opens up a big area of application research using a combination of eigensubspace and machine learning.

In the short term, the findings can be used to address 5G wireless energy optimization challenges wherein the problem involves a large number of communication channels serving an equally large number of users in time varying channel conditions. In the long term, the work can be expanded upon by using a Nonlinear autoregressive exogenous (NARX) machine learning model for forecasting in order to improve the accuracy, while also exploring other machine learning techniques such as Long short-term memory (LSTM) model.

MACHINE LEARNING TECHNIQUES FOR NETWORK ANALYSIS

by
Irfan Lateef

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy in Computer Engineering

Helen and John C. Hartmann Department of
Electrical and Computer Engineering

December 2021

APPROVAL PAGE

MACHINE LEARNING TECHNIQUES FOR NETWORK ANALYSIS

Irfan Lateef

Dr. Ali N. Akansu, Dissertation Advisor — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Nirwan Ansari, Committee Member — Date
Distinguished Professor of Electrical and Computer Engineering, NJIT

Dr. Edip Niver, Committee Member — Date
Professor of Electrical and Computer Engineering, NJIT

Dr. Abdallah Khreishah, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

Dr. Mehmet Ulema, Committee Member — Date
Professor of Computer Information Systems, Manhattan College

BIOGRAPHICAL SKETCH

Author: Irfan Lateef

Degree: Doctor of Philosophy

Date: December 2021

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Engineering,
  New Jersey Institute of Technology, Newark, NJ, 2021
- Master of Science in Telecommunications,
  New Jersey Institute of Technology, Newark, NJ, 2000
- Bachelor of Science in Electrical Engineering,
  Aligarh Muslim University, Aligarh, India, 1987

Major: Computer Engineering

Presentations and Publications:

I. Lateef, and A. N. Akansu, "Machine Learning in Eigensubspace for Network Path Identification and Flow Forecast," *IET Communications*, pages 1997-2006, 2021.

I. Lateef, and A. N. Akansu, "Link-level interpretation of eigenanalysis for network traffic flows," *In Proceedings of Conference on Information Sciences and Systems (CISS)*, pages 1-6, 2017.

Dedicated to my mother Waheedunnisa Begum who inspired me and my wife Farah who supported me.

ACKNOWLEDGMENT

I would like to express my deepest gratitude to Professor Ali Akansu, who not only served as my research advisor, providing invaluable insights and guidance, but also supported me with encouragement and motivation throughout the project. This work would not have been possible without his untiring support and patience.

I would also like to thank Professors Nirwan Ansari, Edip Niver, Abdallah Khreishah and Memhet Ulema for taking the time to serve on my dissertation committee and provide the necessary direction of the work.

A very special thanks to my colleagues Dr. Mustafa Torun and Dr. Onur Yilmaz helping me during the initial discussion and the lab setup.

I would like to acknowledge and thank Dr. Kathleen Meier-Hellstern, Dr. Gagan Choudhury and Dr. Simon Tse from AT&T for the valuable insights into the real-world network challenges and providing the network traffic data to validate the research.

I would like to thank the NOC group at Internet2 for providing the network traffic data and supporting me with the necessary clarification about the format of the data for initial ingestion and validation based on the network topology.

Finally, I would like to thank my four children who cheered me on from time to time. Anim and Ayan who read and corrected the English in my work, Afia who helped me with Adobe Photoshop and Afnan with software and computer support. This endeavor would not have come to fruition without the continuous and unwavering support and patience of my wife, Farah.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

Chapter **Page**

LIST OF TABLES

Table **Page**

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Network Performance Analysis and Network Traffic Engineering has been extensively studied for the last couple of decades using well-known time series techniques and traffic estimation methods. These methods have largely focused on localized data collection and analysis for traffic matrix prediction and other applications. The focus of this dissertation is to look at the entire network holistically and formulate the problem in terms of the state of the entire network using eigensubspace and machine learning techniques. This chapter provides an introduction to the main topics of this dissertation and an outline of the thesis.

1.1 Problem Description

Recent developments in the analysis of network traffic data have mostly focused on classification and characterization of flows by using the flow level statistics [1, 2, 3, 4, 5, 6, 7, 8, 9]. Principal Component Analysis (PCA) was used to analyze the IP flow statistics to identify anomalous behavior in the network [10]. They used PCA as a dimension reduction tool by reducing a very large number of flows into a few most significant *eigenflows* for analysis. Similarly, PCA was used for traffic classification with the IP flow statistics [3]. One of the major challenges with these techniques is the high number of flows in the network, and their lack of link-level granularity, which makes them impractical.

The merits of analyzing the traffic data with PCA to assess the state of the network for certain cases were presented in [10]. They argued that shifts in traffic behavior can be analyzed better by looking at multiple links. They collected the time series data of the Origin-Destination (O-D) pair IP flows traversing multiple links in the network and transformed them onto an *eigensubspace* using PCA. It

was demonstrated that the overall network behavior could be studied by analyzing the first few *eigenflows* (*eigencoefficients*). This scheme suffers from two systemic problems. First, one needs to collect packet level statistics on a per-flow basis for a large number of flows, there are $n(n-1)/2$ flows for a network of n nodes. This is computationally very expensive in terms of data collection and processing. Second, these eigenflows do not reveal the significance and the status of a specific link within the O-D flow in the network. Therefore, if an O-D flow is identified as anomalous, it is not possible to find which link in the path is the root cause of the problem. We address these issues by demonstrating the advantages of the proposed method in the following sections.

1.2 Principal Component Analysis (*Eigen Decomposition)*

The PCA for flow decomposition and prediction by adding extensions and improvement to the basic PCA was used in [11]. They have primarily employed robust PCA (RPCA) that allows the normal data to dwell in a low dimensional space [12]. This is due to the strong correlation among normal observations and special events, as well as allowing noise to dwell in a sparse subspace. This technique was applied to highway traffic data. This dissertation reinforces our perspective of using the PCA components for prediction, but the use of RPCA is not applicable for our case since the focus here is for telecommunication network traffic data. The PCA based network traffic prediction problem was also studied in [13]. In that study, the K-means clustering algorithm to separate the flows into relevant groups was utilized. It also suffers from the same problems discussed above.

1.3 Machine Learning based Analysis

An overview of machine learning (ML) methods to analyze networks and their flows is given in [14]. They presented the state-of-the-art deep learning architectures and

algorithms relevant to the network traffic control systems, and demonstrated a use case for intelligent traffic routing. They also discussed the applicability of deep learning for network flow prediction. Another survey on ML applications in all aspects of networking, including traffic classification, prediction, routing, congestion control, resource management, fault management and network security and their merits are given in [15].

1.4 Dissertation Outline

In this dissertation, we utilize the link level statistics to identify large flows in the network by using PCA (*eigendecomposition*) in a novel way [1]. This technique reduces the data collection overhead and provides a finer tracking of the flows at the link level resolution. It allows improved solutions to network problems. We also use linear regression to predict *eigenvectors* and *eigencoefficients* that are successfully used as features, as explained and verified through performance simulations for the real data of Internet2 network. The dissertation is organized as follows.

In Chapter 2, we provide the industry trend of the tools and techniques available for network performance analysis.

In Chapter 3, we present the theoretical foundation necessary for the discussion in the later chapters.

In Chapter 4, the concept of using *eigenanalysis* for data networks is introduced.

In Chapter 5, a novel time-frequency interpretation of the *eigenanalysis* is presented to identify large flows.

In Chapter 6, we introduce the concept of using machine learning to identify the large network flows.

In Chapter 7, the forecasting method in the network *eigensubspace* is introduced.

In Chapter 8, the computational complexity of the implementation is analyzed and discussed in detail.

In Chapter 9, we present the experimental results and discussion.

Finally, in Chapter 10, the summary of the results is presented, and the conclusions are highlighted.

CHAPTER 2

NETWORK APPLICATION BACKGROUND

Network performance management consists of measuring, modeling, planning, and optimizing networks to ensure that they perform at the targeted efficiency, reliability, and capacity. This dissertation reviews the current practices prevalent in the networking industry for measuring the real-time performance of data networks at different levels of the Open System Interconnection (OSI) model and for different physical and logical layer technologies. It summarizes the usage of this information for network traffic monitoring and traffic engineering applications by the network operators. It identifies some limitations with the current data and practices that need to be addressed. Finally, we present some new data analysis technologies and performance indicators that can be collected and used to improve the current techniques. Even though the information presented is applicable to all types of data networks, the focus of this dissertation is the application and usage in service provider networks.

2.1 Introduction to Traffic Engineering

Internet traffic engineering is defined as the part of network engineering that deals with performance evaluation and optimization of operational IP networks. Traffic Engineering consists of measurement, characterization, modeling, and control of Internet traffic based on scientific principles. The optimization of the network is achieved by evaluating the performance of the network in two dimensions: the traffic level and resources level. Traffic oriented measurements traditionally include the delay, delay variation, packet loss and throughput. The throughput can be expressed statistically as peak rates, mean rates, burst sizes, or as some deterministic notion of effective bandwidth. The resource measurements include the CPU loads, queue

overflows, packet drops and other factors that contribute to the overall performance of the nodes in the network. This optimization is currently implemented in the network through careful capacity and traffic management. Capacity management includes capacity planning, routing control, and resource management. Network resources of particular interest include link bandwidth, buffer space, and computational resources. Certain aspects of capacity management, such as capacity planning, are long-term activities, ranging from days to possibly years. Routing control functions operate in medium term, ranging from milliseconds to days. Finally, the packet level processing functions (e.g., rate shaping, queue management and scheduling) operate at very short-term level, ranging from picoseconds to milliseconds while responding to the real-time statistical behavior of traffic.

Traffic management includes (1) nodal traffic control functions such as traffic conditioning, queue management, scheduling, and (2) other functions that regulate traffic flow through the network or that arbitrate access to network resources between different packets or between different traffic streams.

2.2 Methods and Techniques for Traffic Engineering

This section provides a list of methods and techniques used for network traffic engineering application as given below:

- **Dynamic Routing** - This technique employs *Shortest Path First* (SPF) routing algorithms where costs are based on link metrics. The link metric based on static quantities may be assigned administratively according to local criteria. The link metric based on dynamic quantities may be a function of a network congestion parameters, such as delay or packet loss.

- **Equal Cost Multi-Path (ECMP)** - It is a technique that attempts to address the deficiency in the SPF interior gateway routing systems [RFC-2328].

- **Overlay Model** - The overlay model essentially decouples the logical topology that routers see from the physical topology that is underlying the network. Even

though the overlay model was originally designed for ATM and frame-relay, it has been well adapted to Multiprotocol Label Switching (MPLS) and widely used mechanism for traffic engineering.

- **Constraint-based Routing** - This method computes routes through a network subject to the satisfaction of a set of constraints and requirements. It seeks to optimize overall network performance while minimizing costs.

- **Integrated Services (Intserv)** - This model requires resources, such as bandwidth and buffers, to be reserved a priori for a given traffic flow to ensure that the quality of service requested by the traffic flow is satisfied. The integrated services model includes additional components beyond those used in the best-effort model, such as packet classifiers, packet schedulers, and admission control.

- **Resource Reservation Protocol (RSVP)** - This is a soft state signaling protocol [RFC-2205]. It supports receiver initiated establishment of resource reservations for both multicast and unicast flows. It has been modified and extended to reserve resources for the aggregation of flows in order to set up MPLS explicit label switched paths, as well as to perform other signaling functions within the Internet.

- **Differentiated Services (Diffserv)** - The purpose of this protocol is to devise scalable mechanisms for categorization of traffic into behavior aggregates, which ultimately allows each behavior aggregate to be treated differently, especially when there is a shortage of resources such as link bandwidth and buffer space [RFC-2475].

- **MPLS** - It is a routing protocol that uses labels instead of full IP addresses to forward data packets from one node to the next. This is an advanced forwarding scheme which also includes extensions to conventional IP control plane protocols. MPLS extends the Internet routing model and enhances packet forwarding and path control [RFC-3031].

All the above described technologies are currently used by Tier 1 service providers for traffic engineering applications. Most often the dynamic routing is applied by changing the Interior gateway protocols (IGP), namely, *Open Shortest Path First* (OSPF) and *Intermediate System to Intermediate System* (IS-IS, also written ISIS)

metrics for "best effort traffic" and MPLS with Resource Reservation Protocol-Traffic Engineering (RSVP-TE) is employed for traffic with well known characteristics.

Traffic engineering and optimization is a continuous and evolving process [1] and requires continual development of new technologies and new methodologies for network performance enhancement. The network traffic characteristics have changed in several ways. The traffic itself has increased many folds, the nature of the traffic has evolved from asymmetrical to symmetrical, user traffic has changed from irregular web traffic for short intervals to regular multimedia and streaming data traffic over longer periods of time. The traffic and the business models are also continuously changing to adapt to the evolving trends in Internet usage, like the proliferation of social media, rapid adoption of cloud services and virtualization technologies, and replacement of regular video with Video On Demand (VoD) from sources like YouTube, Netflix and Hulu.

2.3 Traditional Network Performance Monitoring and Analysis

Real-time fault and performance monitoring has been an essential tool for ensuring the efficient utilization of the network and delivering the desired performance to the customer and users of the network. For this purpose, several standards, techniques and tools have been developed by the industry and widely deployed. The data collected by these tools has also been used in non-real-time applications like planning, forecasting, and traffic engineering of the network in the long term. With the recent advances in the hardware and software technologies, a large amount of new data has become available that can be augmented with the existing data that is being collected. This enhanced data set can be used to identify new trends and behaviors and improving the accuracy of the current models. This chapter reviews the standards used, the data collected by these new technologies, and also how they relate to each

other in creating a holistic picture of the state of the network. This section is divided into several subsections:

- Data Measurement Techniques
- Network Monitoring
- Flow Monitoring
- Limitations
- New Ideas

2.3.1 Data Measurement Techniques

The performance metrics are collected at different layers:

- Data link layer (Layer2)
- Network layer (Layer3)
- Transmission layer (Layer4), and
- Application layer(Layer7)

The various techniques used in the measurement of the metrics have been defined in the framework for Internet Protocol Performance Metric (IPPM) [16]. The metrics collected at different layers of the network using Simple Network Management Protocol (SNMP) are as follows:

Data Link Layer - It provides the functional and procedural means to transfer data between network entities and to detect and possibly correct errors that may occur in the physical layer. Specifically, two technologies are considered at this layer:

- Synchronous Optical NETworking (SONET) and Optical Transport Network (OTN) in the optical layer and

- Ethernet in the electrical layer

For the optical layer, there are currently two separate technologies being used in the optical transport, the SONET technology which has been traditionally employed for transporting voice and other digital information over optical network and the OTN technology which has been developed to enable the use of wavelength-division multiplexing (WDM) in the optical networks. Since both these technologies will be present in the networks for the foreseeable future, it is reasonable to assume that network providers will continue measurement and monitoring them in their network. Some of the metrics measured in this layer as defined by ITU-T G.8013/Y.17131 are as follows [17]:

- Frame Loss Ratio is expressed as percentage, of the number of frames not delivered divided by the total number of frames during the time interval T
- Frame Delay is the one-way delay of the frame
- Frame Delay Variation is the measure of the variation in the frame delay between a pair of service frames

For the electrical layer, Ethernet protocol defined by IEEE 802.3 standards is the most commonly used method for medium access control in the data link layer. For the Ethernet links, the following metrics can be measured [18]:

- Link utilization in bits/sec
- Link Errors are defined as errors/sec
- Link Packets are defined as packets/sec

Network Layer - It is also called Internet Protocol (IP) layer based on the protocol used to deliver the functionality at this layer. The network layer provides

the functional and procedural means of transferring variable length data sequences from a source host on one network to a destination host on a different network, while maintaining the quality of service requested by the transport layer. The octets/bytes entering an interface at any given node provides the interface effective ingress bandwidth utilization of that link. The octets/bytes leaving an interface provide the effective egress bandwidth utilization of the link. If there are no losses in the lower layers, then the octets arriving at the interface should be the same as the octets leaving the corresponding interface on the other end of the network link. Therefore, a connection between New York (NY) and Los Angeles (LA) will be analyzed as two separate links in two directions. The data collected at this layer is specified by the RFC1213 [19]

- ifInOctets
- ifInErrors
- IfOutOctets
- ifOutErrors

A second metric available from the network layer using the ping tool is the round trip time (RTT) taken by a packet. This gives an indication of the congestion in the network.

Transport Layer - It is also known as the Transmission Control Protocol (TCP) layer, based on the protocol predominantly used in this layer. The TCP provides a communication service at an intermediate level between an application program and the IP. It provides host-to-host connectivity at the transport layer of the Internet model. The network management systems (NMS) generally collect the connection related metrics as defined by RFC4022 [20]. In the context of network traffic analysis, these metrics are not very helpful. However, the routers also provide a

fine-grained flow level metrics collection from the router interface. For example, Cisco provides the NetFlow protocol and Juniper provides the JFlow protocol for collecting the statistics of the TCP/IP flows passing through the router [21][22]. These statistics provide an insight into what applications are consuming the bandwidth and which flows are a major consumer of bandwidth. This can be used for identifying the paths in the network which are heavily used and rerouting/load balancing the flows on alternate paths. These are some of the key metrics used in this dissertation for the research on network performance analysis.

- IPv4 Source Address
- IPv4 Destination Address
- IPv4 TOS
- Protocol Type
- Source Port
- Destination Port
- ICMP Type
- Input Intf SNMP Index
- IPv4 Source Mask
- IPv4 Destination Mask
- Source AS
- Destination AS
- TCP Flags
- Output Interface SNMP Index

- IPv4 Next Hop
- Number of Bytes
- Number of Packets
- Flow Start Time
- Flow End Time

Application layer - This layer interacts with software applications that implement a communicating component. The applications written at this layer which are used in network monitoring are One-Way Active Measurement Protocol (OWAMP) as defined in RFC4656 [23] and Two-Way Active Measurement Protocol (TWAMP) RFC5357 [24]. These applications provide the following two metrics in the network.

- One-Way Delay
- Two-Way Delay

The metrics from this layer are not used in the work presented in this dissertation. The data from different layers is collected using different SNMP, NETCONF and vendor specific protocols.

2.3.2 Network Monitoring

A monitoring system monitors the network for problems caused by overloaded systems and/or crashed servers, network connections impairments at physical or logical level due to physical disruptions (cable cuts or damage due to wear) and logical disruptions due to failure of active and passive devices in various layers of the network. The monitoring can be done at every node/device and link/interface, as called network performance monitoring. A second way of monitoring the health of the link in

the network is by monitoring the traffic at the edge of the network and building traffic matrices. This method is called Network Tomography [8]. It has been widely studied and is not the focus of this study [25][26]. A third way is to monitor the routing behavior of the network; this area is called Route Analytics which includes the systems, algorithms and tools to monitor the network [27]. This is beyond the scope of this work. In this Section, the capabilities, features and limitations of some common tools are discussed.

The network monitoring tools are categorized based on the information that is being monitored. At a high level we can have the following categories:

- **Up/Down Monitoring** - This is a simple yet powerful mechanism where alarms and traps are used to monitoring the line condition and find out if it is up/down. This is the condition we would like to predict before it happens based on the other metrics in the network. Our focus is not on this type of monitoring. There are several tools that facilitate this type of monitoring. For example, WhatsupGold [28], Nmap [29] and Ping are very popular in the user community. The comparison of the tools in this category is beyond the scope of this dissertation.

- **Performance Monitoring/SNMP Monitoring** - This kind of monitoring is done when the line is up and status is green, but the conditions are not perfect. The data collected for this type of monitoring is listed in the data measurement Section above. This is the focus of the current research topic, where the data is collected and analyzed in a novel way providing a new perspective of the network not shown before.

There are several network management systems (NMS) that facilitate the data collection, analysis and monitoring of the network. A comprehensive list of tools for gathering data from the network are listed at the Stanford University Website [30]. A comprehensive list of tools and their comparison is given in the references [31][32]. Some of the popular tools are discussed as follows:

- **Multi Router Traffic Grapher (MRTG)** [33] - This tool typically collects data using SNMP at a given interval (generally 5 mins) and displays it as a function of time. It does not perform any analysis across the links.

- **Paessler Route Traffic Grapher (PRTG) Network Monitor**[8] - This is good tool for monitoring small networks using windows platforms and can also use the NetFlow enabled devices for flow monitoring. However, it does not scale to large networks.

- **OpenNMS** [34] - OpenNMS is by far the best and most comprehensive network management tool that supports performance measurement, event and notification management, alarm correlation and automated discovery and provisioning of the network devices. However, it does not provide a correlation of alarms with the performance and link level measurements and certainly not multilink and multilayer correlation.

- **HP Openview Network Node Manager** [35], HP OpenView Network Node Manager is a network monitoring tool that uses SNMP and other technologies to retrieve the information it requires. It provides a simple and intuitive network status summary, maps of networks both physical and virtual and quick view of incoming alarms. It can provide a good alarm correlation based on network topology, and it can handle networks of any size and complexity. It is probably the most widely deployed application of its kind.

- **SolarWinds** [36], It is another commercially available tool that does network performance monitoring using SNMP and bandwidth monitoring using tools like NetFlow, JFlow and IP Flow Information Export [37]. Orion Network Performance Monitor is part of SolarWinds, it provides correlation among different tools.

- **IBM Netcool** - This is a good network management tools with standard features, it does not have any network topology specific analytics in the software suite.

- **SevOne** [38]- This is also a popular performance measurement collection tool and widely used in the industry. It is a highly scalable distributed architecture based on appliances real and virtual placed in the network collecting and analyzing data with support for big data technologies.

2.3.3 Flow Monitoring

Flow monitoring is the fine-grained collection of the network traffic data at the TCP/IP flow level, enabling the understanding of flows across the network. This kind of data collection is enabled by devices that support NetFlow /JFlow /IPFIX and export data in this format to the monitoring system. The tools discussed in this section do not take into account the availability of this kind of data in their monitoring system. These systems have not been designed to exploit the availability of this information from the network. One of the tools that does flow monitoring is Guavus. This is the software that comes closest to analyzing the network wide statistics. The other tool is PeakFlow from Arbor Networks. These two software tools are discussed below.

- **Guavus** - Guavus Big Data analytics platform is a grid based, scalable, and highly available computing architecture, it offers a powerful solution for data analytics using an innovative paradigm of stream processing and analytics for collection, processing and advanced visualization. The analytics is based on Lakhina's Ph.D. thesis [10] [39] on network wide analytics using Principal Component Analysis (PCA) for anomaly detection in the network traffic. We are proposing improvement to this work as suggested in this thesis.

- **Arbor Network PeakFlow** - Arbor provides a commercial tool called PeakFlow that collects and analyzes network wide flow level statistics. This tool is focused on security management and distributed denial of service (DDOS) protection and threat management system. It also claims to perform traffic engineering by correlating the topology to the historical performance statistics. It is not clear if it uses the flow level data with link level statistics.

2.3.4 Limitations

The tools discussed above generally have the following features: automated network discovery, network performance monitoring at a node level, link level and interface level. There are two main limitations with these tools.

- Firstly, the tools do not analyze the data at the network wide level and provide correlations of behavior.

- Secondly, they do not provide a multi-layer correlation of statistics for fault isolation and analysis.

In the later chapters, we propose some ideas to address these concerns. There are several reasons for the lack of network wide analytics. Until recently, the infrastructure to collect and process the data in a large network was almost non-existent and prohibitively expensive. With the advent of cloud computing, this kind of big data analytics have come within the realm of practical possibilities.

2.3.5 New Ideas

This dissertation explores ideas to leverage such technologies. Another reason is the virtualization of network functions has now made it possible to process the raw data at the collection point within a cloud instance. This greatly reduces the processing requirements on the collecting nodes and eliminates the need for CPU resources in the network elements. The multi-layer analytics have largely been unattractive due to the lack of information at different layers. Lately, the separation of Layer-2 transport infrastructure and Layer-3 routing infrastructure have had added some true meaning and value to this kind of analysis. The recent development of newer metrics like one-way and two-way delay at the application layer have not been incorporated into older platforms.

Hence, there is a strong motivation to pursue research in the area of network-wide analysis using the current metrics data in a scalable fashion.

CHAPTER 3

THEORETICAL FOUNDATION

This chapter presents the theoretical foundation and the principles behind the *eigendecomposition*, and its application in signal processing and dimensionality reduction. These concepts are applied to network traffic engineering problems in this dissertation. We present the theoretical framework in this chapter and later use it in the development of analytic techniques. This chapter includes discrete autoregressive order one, AR(1), signal model, orthogonal transform and *eigendecomposition* for the AR(1) signal model statistics.

3.1 Discrete AR(1) Signal Model

Autoregressive discrete process of order one, AR(1), is one of the most widely used signal model for the performance analysis and comparative evaluation of signal processing techniques. It is the first order approximation for many real-world signals like images, network traffic data time series and others. The AR(1) process can be expressed as [1]

$$x(n) = \rho x(n-1) + \xi(n) + c \tag{3.1}$$

where $\xi(n)$ is the white noise sequence with zero-mean and variance σ_ξ^2,

$$E\left\{\xi(n)\xi(n+k)\right\} = \sigma_\xi^2 \delta_{n-k} \tag{3.2}$$

and c is a constant and δ_{n-k} is the Kronecker delta function. The first order correlation coefficient ρ of the AR(1) model for wide-sense stationary (WSS) with $-1 < \rho < 1$ is defined as

$$\rho = \frac{R_{xx}(0)}{R_{xx}(1)} = \frac{E\{x(n)x(n+1)\}}{E\{x(n)x(n)\}} \tag{3.3}$$

The mean of $x(n)$ is calculated as

$$\mu_x = E\{x(n)\} = \frac{c}{(1-\rho)} \tag{3.4}$$

and, the variance is calculated as

$$\sigma_x^2 = E\left\{x(n)^2\right\} - \mu_x^2 = \frac{\sigma_\xi^2}{(1-\rho^2)} \tag{3.5}$$

The auto-correlation sequence for the AR(1) model is given as

$$R_{xx}(k) = E\{x(n)x(n+k)\} = \sigma_x^2 \rho^{|k|}; k = 0, \pm 1, \pm 2, \ldots \tag{3.6}$$

The resulting Toeplitz correlation matrix of size $N \times N$ for AR(1) source is shown to be in the form [1]

$$R_x = \sigma_x^2 \begin{bmatrix} 1 & \rho & \rho^2 & \cdots & \rho^{N-1} \\ \rho & 1 & \rho & \cdots & \rho^{N-2} \\ \rho^2 & \rho & 1 & \cdots & \rho^{N-3} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \rho^{N-1} & \rho^{N-2} & \rho^{N-3} & \cdots & 1 \end{bmatrix} \tag{3.7}$$

3.2 Orthogonal Signal Expansions

Let us define a set of basis vectors $\left\{\underline{e}_0, \underline{e}_1, \ldots, \underline{e}_{N-1}\right\}$ in the N dimensional Euclidean space. These vectors are assumed to be linearly independent, such that a linear combination of

$$c_0\underline{e}_0, c_1\underline{e}_1, \ldots, c_{N-1}\underline{e}_{N-1} \tag{3.8}$$

will not exist if and only if $c_0 = c_1 = \cdots = c_{N-1} = 0$. In other words, any given vector cannot be expressed as a linear combination of any other vector.

Let us define a signal samples $\{f(k)\}$ that can be represented by a weighted sum of component sequences as given below

$$f(n) = \sum_{k=-\infty}^{\infty} f(k)\delta(n-k) \tag{3.9}$$

where $\delta(n-k)$ is the Kronecker delta sequence:

$$\delta(n-k) = \begin{cases} 1, & n-k=0 \\ 0, & otherwise \end{cases} \tag{3.10}$$

The norm of $\underline{f}$ for a finite number of dimensions is defined as [1]

$$norm(\underline{f}) = \left[\sum_{k=0}^{N-1} |f(k)|^2\right]^{1/2} \tag{3.11}$$

Based on the above equations, it can be said that, $\{f(k)\}$ can represent a point in the N dimensional Euclidean space spanned by the basis vectors $\left\{\underline{e}_0, \underline{e}_1, \ldots, \underline{e}_{N-1}\right\}$.

Now, let us assume that $\{x_n(k), 0 \leq n, k \leq N-1\}$ represent a family of N linearly independent sequences on the interval $[0, N-1]$. This family of sequences are considered to be orthogonal if [1]

$$\sum_{k=0}^{N-1} x_n(k)x_s^*(k) = c_n^2\delta(n-s) = \begin{cases} c_n^2 & n=s \\ 0 & otherwise \end{cases} \tag{3.12}$$

where c_n is the norm of $\{x_n(k)\}$. The orthonormal family of sequences corresponding to the sequence defined in the Equation 3.12 is obtained by the normalization as

follows

$$\phi_n(k) = \frac{1}{c_n} x_n(k) \qquad 0 \leq n \leq N-1 \tag{3.13}$$

As a corollary to Equation 3.13, it can be shown that

$$\sum_{k=0}^{N-1} \phi_n(k)\phi_s^*(k) = \delta(n-s) \tag{3.14}$$

Therefore, it can be said that, any nontrivial set of functions satisfying Equation 3.14 forms an orthonormal basis for the linear vector space. Consequently, $\{f(k)\}$ can be uniquely represented as [1]

$$f(k) = \sum_{n=0}^{N-1} \theta_n \phi_n(k), \qquad 0 \leq k \leq N-1 \tag{3.15}$$

where

$$\theta_s = \sum_{k=0}^{N-1} f(k)\phi_s^*(k), \qquad 0 \leq s \leq N-1 \tag{3.16}$$

The set of coefficients $\{\theta_s,\ \ 0 \leq s \leq N-1\}$ are known as the spectral coefficients of $\{f(k)\}$ relative to the given orthonormal family of basis functions. These are called *generalized Fourier* coefficients, even when the family of functions represented by $\{\phi_n(k)\}$ are not sinusoidal [1].

3.3 Least-Squares Interpretation

In this section, a least squares interpretation is provided to the orthogonal signal expansion presented in the previous section. The set of coefficients $\{\theta_n\}$ as defined in Equation 3.16 also provides the least-squares approximation to $\{f(k)\}$ [1]. Let us suppose an approximation of $\{f(k)\}$ can be made by the superposition of the first L of the N basis sequences, using weighting coefficients $\{\gamma_i,\ \ i = 0, 1, \ldots, L-1\}$. Then the optimal least-squares approximation for these coefficients is given as below

$$\{\gamma_i = \theta_i, \quad i = 0, 1, \ldots, L-1\} \tag{3.17}$$

Now, let us assume that the approximation of the orthonormal basis function can be defined as

$$\hat{f}(k) = \sum_{r=0}^{L-1} \gamma_r \phi_r(k) \tag{3.18}$$

and the corresponding error in approximation is defined as

$$\epsilon(k) = f(k) - \hat{f}(k) \tag{3.19}$$

Then, the $\{\gamma_r\}$ can be chosen to minimize the sum squared error as shown below

$$J_L = \sum_{k=0}^{N-1} |\epsilon(k)|^2 \tag{3.20}$$

Expanding the above equation by substituting the values from Equation 3.19 and invoking orthonormality and setting the partial of J_L with respect to γ_s to zero gives the following solution (For the proof, see Section 2.1.2 of [1])

$$\gamma_s = \sum_{k=0}^{N-1} f(k)\phi_s(k) \equiv \theta_s \tag{3.21}$$

A simple sketch depicting the relationship is shown in Figure 3.1 for the case of $N = 3, L = 3$.

The Figure 3.1 demonstrates that $\left\{\hat{f}(k)\right\}$, the least square approximation to $\{f(k)\}$ is the orthogonal projection of $\{f(k)\}$ onto the two-dimensional subspace spanned by basis sequences $\{\phi_1(k)\}, \{\phi_2(k)\}$. It may also be noted that these results are valid for infinite dimensional spaces and finite dimensional spaces, as long as the norms of the sequences (L^2) are bounded [1].

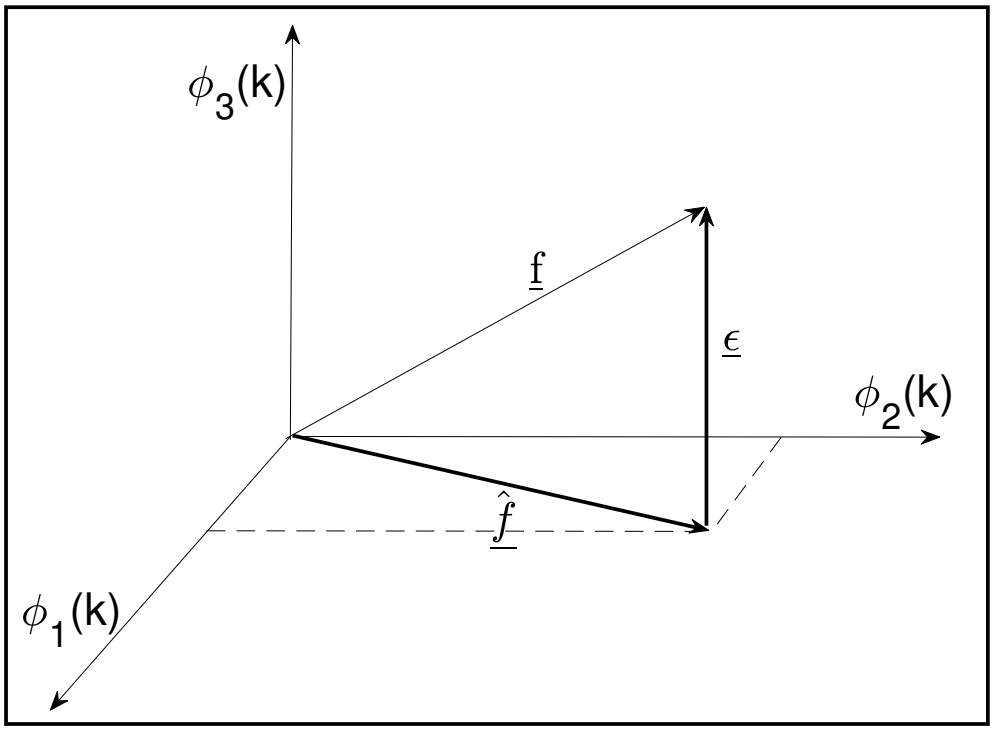


Figure 3.1 Orthogonality principle demonstration.
Source:[1]

3.4 Block Transforms

A vector-matrix representation of the orthonormal expansions described in the previous section provides an a compact form for block transform for various matrix operations and interpretation. The signal and spectral vectors representations given as below

$$\underline{f}^T = [f_0, f_1, \ldots, f_{N-1}] \tag{3.22}$$

$$\underline{\theta}^T = \theta_0, \theta_1, \ldots, \theta_{N-1} \tag{3.23}$$

Let us assume that the real orthonormal sequences $\phi_r(k)$ are represented by the rows of the transformation matrix $\phi(r, k)$

$$\Phi = [\phi(r,k)] : k, n = 0, 1, \ldots, N-1 \tag{3.24}$$

It can be shown that

$$\underline{\theta} = \Phi \underline{f} \tag{3.25}$$

and

$$\underline{f} = \Phi^{-1}\underline{\theta} = \Phi^{T}\underline{\theta} \tag{3.26}$$

therefore it can be said that

$$\Phi^{-1} = \Phi^{T} \tag{3.27}$$

Using subspace orthonormality property, it can be stated that

$$\Phi\Phi^{-1} = \Phi\Phi^{*T} = I \tag{3.28}$$

The above equation, implies that the inverse of Φ is its conjugate transpose, defines a *unitary* matrix where $*T$ indicates the conjugate transpose of a matrix.

3.5 Eigendecomposition of Correlation Matrix

An eigenvalue λ and its paired eigenvector ϕ of an $N \times N$ correlation matrix R_x satisfy the matrix equation

$$R_x\phi = \lambda\phi \tag{3.29}$$

$$R_x\phi - \lambda I\phi = (R_x - \lambda I)\,\phi = 0 \tag{3.30}$$

such that $(R_x - \lambda I)$ is singular. Namely

$$det\,(R_x - \lambda I) = 0 \tag{3.31}$$

R_x of AR(1) process given in Equation 3.7 , is real and symmetric matrix, and its eigenvectors are linearly independent. Therefore, the determinant is a polynomial of degree N in λ , and has N roots and Equation 3.30 has N solutions for ϕ that result in the eigenpair set $\{\lambda_k, \phi_k\}\,; 0 \leq k \leq N-1$. Therefore, the eigendecomposition of R_x is expressed as [1]

$$R_x = A_{KLT}^T \Lambda A_{KLT} = \sum_{k=0}^{N-1} \lambda_k \phi_k \phi_k^T \tag{3.32}$$

where $\Lambda = diag\,(\lambda_k)\,; k = 0, 1, \ldots, N-1$, and k^{th} column of A_{KLT}^T matrix is the k^{th} eigenvector ϕ_k of R_x with the corresponding eigenvalue λ_k.

CHAPTER 4

EIGENANALYSIS AND EIGENFLOWS OF A NETWORK

4.1 Eigenanalysis and Eigenflows

The *eigendecomposition* of $N \times N$ matrix $\mathbf{R}$ is expressed as [1]

$$\mathbf{R}_\theta = \mathbf{\Phi R \Phi^T} = \mathbf{\Lambda} \tag{4.1}$$

where Λ is the diagonal matrix with its elements as the *eigenvalues,* and equivalently, $\mathbf{R}_\theta$ is the covariance matrix of *eigencoefficients.* $\mathbf{\Phi}$ is the *eigenmatrix* of $\mathbf{R}$ and populated by the *eigenvectors* as its columns, and $\mathbf{T}$ indicates matrix transpose operation. It defines the resulting N-dimensional orthonormal *eigensubspace* for the given $\mathbf{R}$ [40].

In this dissertation, *eigensubspace* representations of random vectors, that describe traffic variables like link bandwidth at measurement instances in megabits per second (Mbps), are used to analyze and understand the status of communication networks. Let's assume a network of N links, and the snapshot of the link traffic vector at a periodic measurement time point is expressed by $\mathbf{r^T} = [r_1, r_2, ..., r_N]$. The empirical correlation matrix $\mathbf{R}$ of link traffic is calculated as for the predefined historical measurement data window of W samples as [1]

$$\mathbf{R} = \begin{bmatrix} r_{11} & \cdots & r_{1N} \\ \vdots & \ddots & \vdots \\ r_{N1} & \cdots & r_{NN} \end{bmatrix} \tag{4.2}$$

where $r_{ij} = cov(r_i r_j)/(\sigma_{r_i}\sigma_{r_j})$ for W and σ_{r_i} and σ_{r_j} are the standard deviations of r_i and r_j, respectively.

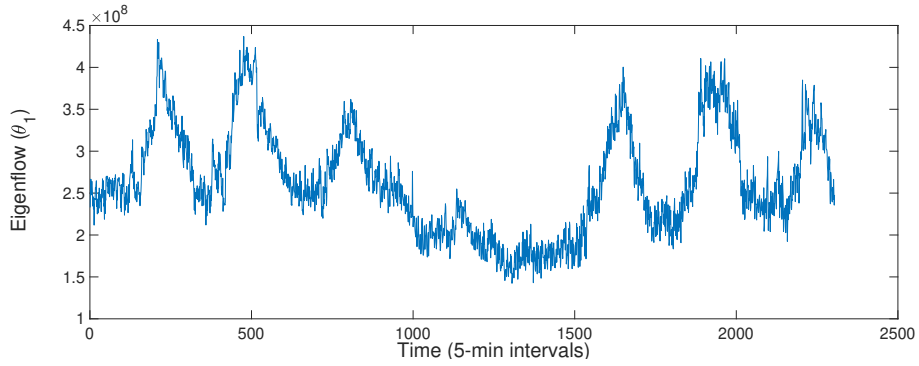


Figure 4.1 The variation of the first *eigenflow* in time.

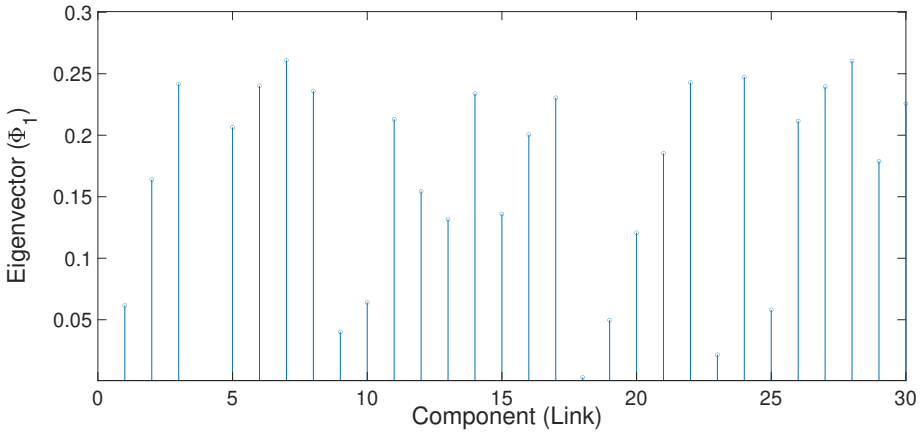


Figure 4.2 The corresponding components of the first *eigenvector* for one-week duration of Internet2 link data where N=30.
Source: [40]

It is used in Equation (4.1) to create the *eigensubspace* expressed in $\mathbf{\Phi}$ matrix. The measurements are repeated periodically to update the *eigensubspace* due to the statistical variations (non-stationarity) of the network dynamics. The snapshot of *N link bandwidths* populate the column vector $\underline{c}^n$ where n is the *measurement time* index in the regular clock with the assumption of the stationarity during the update period until *n+1*. Then, one can project this link traffic vector onto the currently defined *eigensubspace* as (forward transform) [1]

$$\underline{\theta^n} = \mathbf{\Phi^T}\underline{\mathbf{c}}^{\mathbf{n}} \tag{4.3}$$

where the most significant eigenvector, also known as the principal *eigenvector*, is commonly the first column of the *eigenmatrix* Φ. The link variables are expressed through the inverse transformation operator as

$$\underline{c}^{\mathbf{n}} = \mathbf{\Phi}\underline{\theta}^{\mathbf{n}} \tag{4.4}$$

Now, we can rewrite this expression more explicitly as follows

$$\begin{bmatrix} c_1^n \\ \vdots \\ c_N^n \end{bmatrix} = \begin{bmatrix} \Phi_{11} & \cdots & \Phi_{1N} \\ \vdots & \ddots & \vdots \\ \Phi_{N1} & \cdots & \Phi_{NN} \end{bmatrix} \begin{bmatrix} \theta_1^n \\ \vdots \\ \theta_N^n \end{bmatrix}$$
$$= \begin{bmatrix} \theta_1^n\Phi_{11} + \theta_2^n\Phi_{12} + & \cdots & +\theta_N^n\Phi_{1N} \\ \vdots & \ddots & \vdots \\ \theta_1^n\Phi_{N1} + \theta_2^n\Phi_{N2} + & \cdots & +\theta_N^n\Phi_{NN} \end{bmatrix} \tag{4.5}$$

Note that the components of the first eigenvector in (4.5) are $[\Phi_{11}\,\Phi_{21}\,...\Phi_{N1}]$ as the first column vector of Φ.

We compute *eigenflows* $\{\theta_k^n, k = 1, 2, \cdots N\}$ that characterize the *entire network* at time n by using Equation (4.3). Figure 4.1 displays the variations of the most significant *eigenflow*, θ_1^n at time n, for the network. The components of the *principal eigenvector* are displayed in Figure 4.2 for the case where one week of the Internet2 link data is used [41]. These components represent the contribution from the corresponding links into the most significant *eigenflow* of the network.

As a corollary to Equation (4.4), we express the k^{th} link traffic for the measurement at time n in the *eigensubspace* is expressed as

$$\mathbf{c_l^n} = \sum_{\mathbf{k=1}}^{\mathbf{N}} \theta_{\mathbf{k}}^{\mathbf{n}}\mathbf{\Phi_{kl}}\, l = 1, 2, ..., N \tag{4.6}$$

In Chapter 5, we give a new interpretation of *eigenflows* with the link-level focus for the network using the *eigensubspace* representation framework described above.

CHAPTER 5

INTERPRETATION OF EIGENFLOWS

5.1 Link-Level Interpretation of Eigenflows

The vector $\underline{\theta^{\mathbf{n}}}$ in Equation (4.3) represents the *eigenflows* (*eigencoefficients*) in the network at time n. For example, the first *eigenflow* is the inner product of the first *eigenvector* and traffic measurement vector $\underline{c}^n$ at time n with link attributes as its components. A high value of an *eigenvector* component is related to high contribution of that link into the given *eigenflow* due to the least squares (L2 norm) based optimality of the *eigendecomposition* [1].

Figure 5.1a shows the component values of all *eigenvectors* of the link data matrix where N=30. In contrast, Figure 5.1b displays the corresponding joint time-frequency matrix $\mathbf{\Psi}$ as defined in Equation (5.3) below where the vertical axis is the component indices of *eigenvectors* and the horizontal axis is the *eigenvector* indices [1, 2]. In contrast to the traditional PCA based studies reported in the literature [42, 43, 44], our experiments show that a few most significant *eigenvectors* and *eigenflows* (dimension reduction) are not always able to capture all the network characteristics. They are rather spread out to a larger subset, as shown in Figure 5.1a. We highlight that some components of *eigenvectors* with low *eigenvalues*, that may not survive the dimension reduction step, can be significant for the representation of certain link traffic in the *eigensubspace*. Therefore, we need to jointly look at the variations of all *eigenvectors* and all *eigenflows* in time in order to better track network dynamics and anomalies with the highest level of representation granularity.

Eigenflow (frequency domain) interpretation of network dynamics does not emphasize link or path specific (time/signal domain) features in a network. These features are equally critical to assess the state of the network. We propose the

joint time-frequency interpretation [2] of *eigenanalysis* for network engineering, as explained in the next Section [1].

5.2 Representation of a Link in Eigensubspace

Let $\{c_l^n\}$ represent link measurements at time instance n for the entire network, l = 1,2, ... N. Let us define a more detailed *eigensubspace* representation of the network status where the product of *eigenflows* and *eigenvectors* are interpreted for analysis as shown in the matrix

$$\mathbf{\Psi} = \mathbf{\Phi\Theta} \tag{5.1}$$

where

$$diag\{\mathbf{\Theta}\} = \mathbf{I}\underline{\theta^n} = diag\{\Theta\} \tag{5.2}$$

and, I is $N \times N$ identity matrix.

It is explicitly shown as

$$\mathbf{\Psi} = \begin{bmatrix} \theta_1^n\Phi_{11} & \cdots & \theta_N^n\Phi_{1N} \\ \vdots & \ddots & \vdots \\ \theta_1^n\Phi_{N1} & \cdots & \theta_N^n\Phi_{NN} \end{bmatrix} \tag{5.3}$$

where θ_k^n is the k^{th}*eigenflow* and Φ_{kl} is the l^{th} component of the k^{th} *eigenvector* corresponding to the l^{th} link of the network at time n.

The sum of the elements of the l^{th} row of $\mathbf{\Psi}$ is equivalent to the traffic measurement on the l^{th} link c_l^n as given in Equation (4.6). $\mathbf{\Psi}$ is called the *joint time-frequency matrix*. Figure 5.1b displays the *joint time-frequency matrix* for the link measurements of the network that we analyze in the dissertation [1, 2]. We will

use this matrix to identify large network data flows through the network topology, as described in the next section.

5.3 Identification of Large Network Flows

We demonstrate the use of the *joint time-frequency matrix* to identify large network flows. The identification process is comprised of the following steps.

1. Identify the column vector containing the most significant components. This is done by taking the L2-norm of the column vectors and sorting them. It is seen from Figure 5.1b that the $7th$ column vector of $\mathbf{\Psi}$ has the most significant components for this case.

2. Find the components of the vector identified in Step.1 with significant contributions by setting a threshold $\alpha = 0.35$. The components of that vector are plotted as a bar graph in Figure 5.2a.

3. Map the components to the network topology [40].

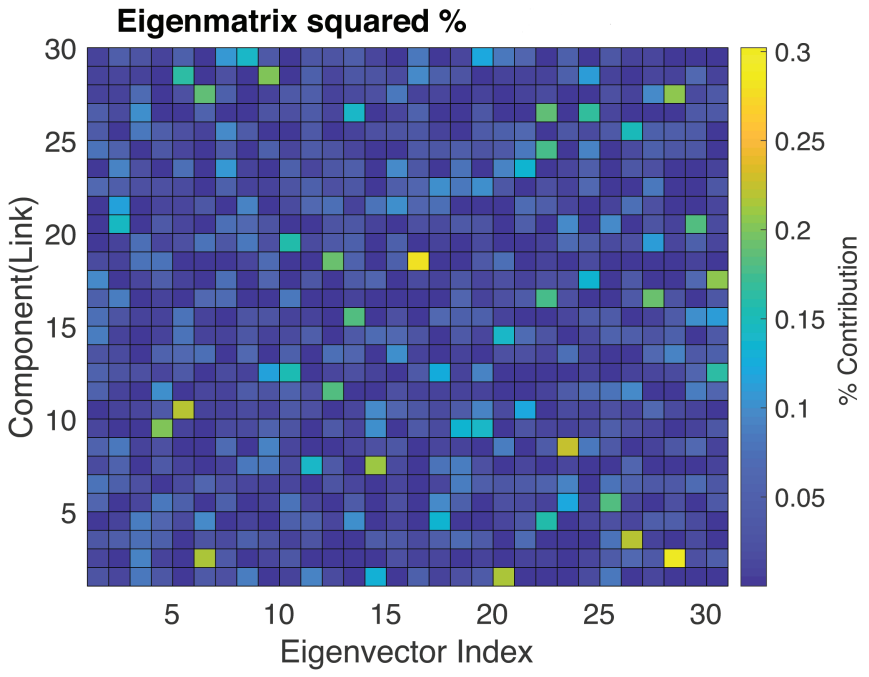
4. Identify the paths in the network created by the referenced eigenvector components. This can be done by verifying that more than one link corresponding to the components are connected together to form a path. It is seen in Figure 5.2b that the links identified in Step.3 traverse a path taken by a flow from DC to LA. We also observe that there is a second path taken by another flow from Seattle to Houston.

5. Go back to Step.1 and repeat the process for the next significant column vector of $\mathbf{\Psi}$.

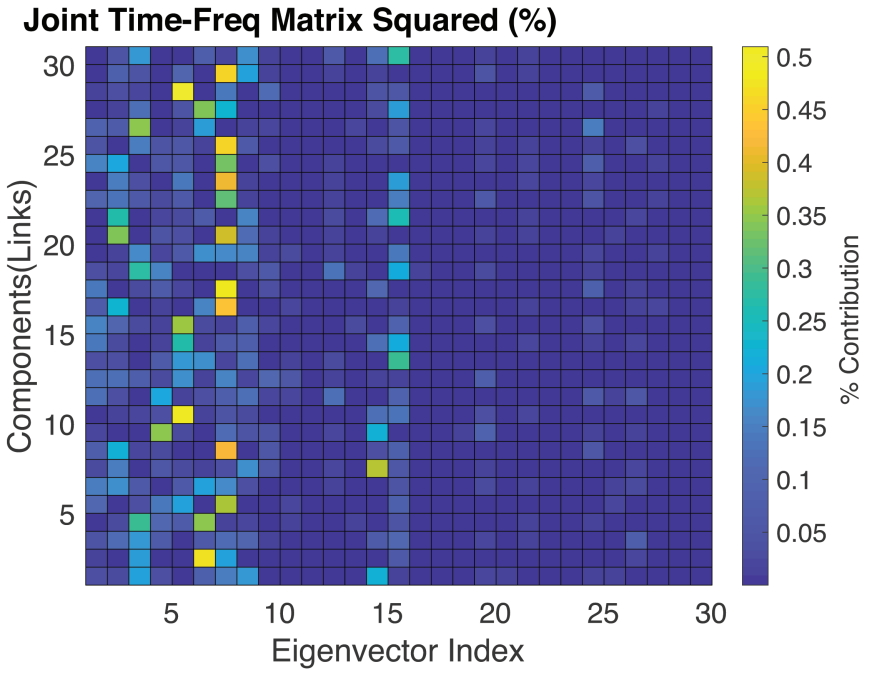
Note that Link6 is mapped in the Figure 5.2b even though it is below the threshold because its two adjacent links are significant contributors to a large flow. By inference, it has to pass through this link. While performing Step.4, there may be some links that are above the threshold but are not connected to any other link to form a path. Such links can be safely ignored from further analysis, mapping or plotting. For example, in Figure 5.2b Link5 and Link20 are ignored.

The link-level interpretation of *eigenanalysis* and its mapping on to the topology demonstrates that large flows in the network can be detected. It helps us to better understand the network activity in a real-world scenario with a more efficient implementation than the currently used methods [3, 4, 5]. Note that the selection of the threshold to identify significant links requires some experimental study on the network of interest and may also be automated.

There are various applications of this research. Network traffic has been growing drastically in the last decade with the advent of smartphones and proliferation of video applications. Due to this, it has become a huge challenge to characterize, forecast and engineer network traffic. The traffic is largely comprised of many small flows and a few large flows over a period of time with their specific network requirements. The identification of these large flows as addressed in the dissertation has a major impact in the overall network performance and security. If the flow is from a trusted source on the expected path, then the information is used for traffic engineering and optimization applications. The link level granularity of data is instrumental in long-term network planning applications. It is also used in congestion control, resource management, fault management and Quality Of Service (QOS) management to name a few. Besides the network applications mentioned above, there are several other uses in the areas of application monitoring, security awareness and intrusion detection, policy validation and service assurance. For example, if a large flow is identified and the source or the path is anomalous, then it could be a security violation or a Denial of Service (DOS) attack. It is extremely difficult to build these applications that infer the network behavior by merely looking at the traffic on a single link or to the entire network. Thus, the joint time-frequency interpretation of network traffic data in the *eigensubspace* gives better insight and features in analyzing link level and network wide behavior.

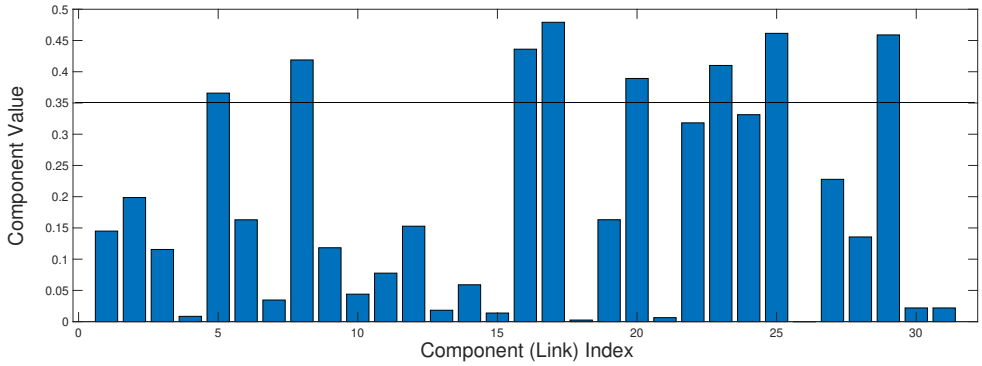


(a)

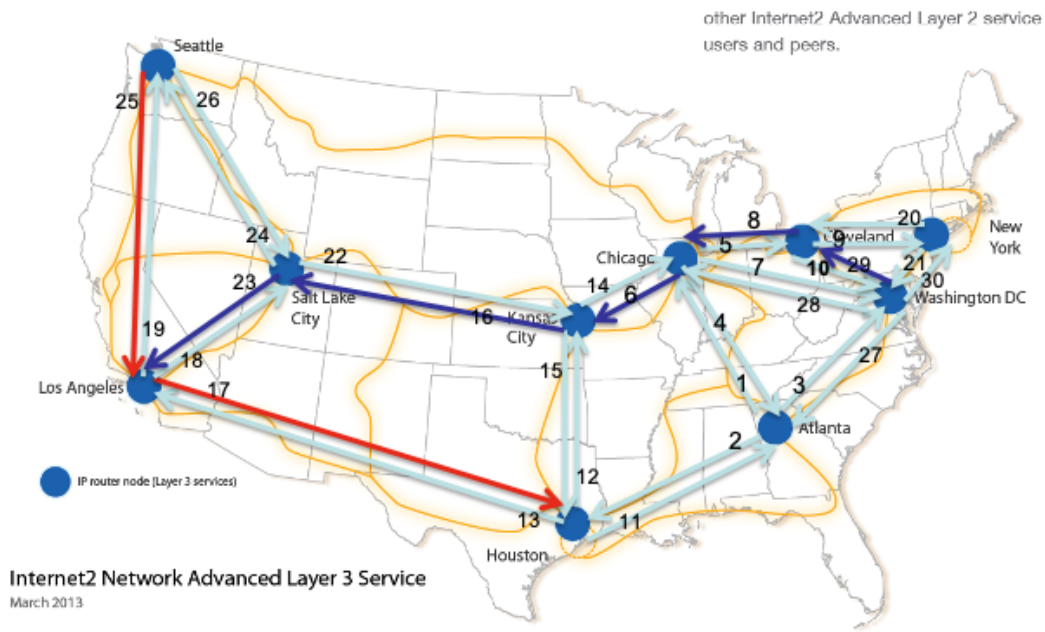


(b)

Figure 5.1 (a) Components of Eigenvectors, and (b) Joint time-frequency representation of eigenanalysis where N=30.



(a)



(b)

Figure 5.2 (a) 7th column vector of $\mathbf{\Psi}$ in Figure 5.1b, and (b) Mapping of the significant components of the 7th column vector to network paths.

CHAPTER 6

MACHINE LEARNING TECHNIQUES FOR FLOW IDENTIFICATION

6.1 ML to Identify Paths with Network Features in Eigensubspace

The use of machine learning (ML) methods for network traffic classification and flow prediction was reported in [14]. We extend that work to investigate the use of network *eigenfeatures* in such learning methods. In Section 5.3, we used manual thresholding of the *joint time-frequency matrix* elements defined in the *eigensubspace* to identify large network flows. The manual thresholding as described in Section 5.3 suffers from several short comings:

- The threshold needs to be set heuristically based on historical data. Therefore, the threshold detection will fail as soon as the trend diverges from the historical trend. Hence, it may produce inaccurate results in the long-run when the non-stationarity in data becomes significant.

- The value of threshold is crucial for this method, and it becomes a major challenge due to the exponential growth in broadband traffic and increased data types. It may be an interesting research topic to employ ML method for threshold selection that is beyond the scope of this study.

- Finally, the usage patterns are driven by many events and large geographies where the thresholding method may not work as the center of gravity of source and destinations of the traffic change.

Hence, we came up with the ML method that learns from its past experience and quickly adapts to new trend in the traffic characteristics automatically without manual intervention.

In this section, we present the method to automate the identification process. Instead of thresholding, we employ a machine learning technique utilizing the same column vectors of the *joint time-frequency matrix*, see Equation (5.3), to identify

paths carrying large traffic flows. It is based on an artificial neural network (ANN) using a multilayer perceptron (MLP) model.

The MLP is a class of feedforward (FF) artificial neural network [45]. A single neuron multilayer perceptron (MLP) model is shown in Figure 6.1. Its inputs $x^T = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}$ are weighted by the coefficients $w^T = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}$ and summed together in ν. The output y is obtained through the activation function g(ν) with the threshold T. Each node in a neuron uses a nonlinear activation function except for the input nodes [45].

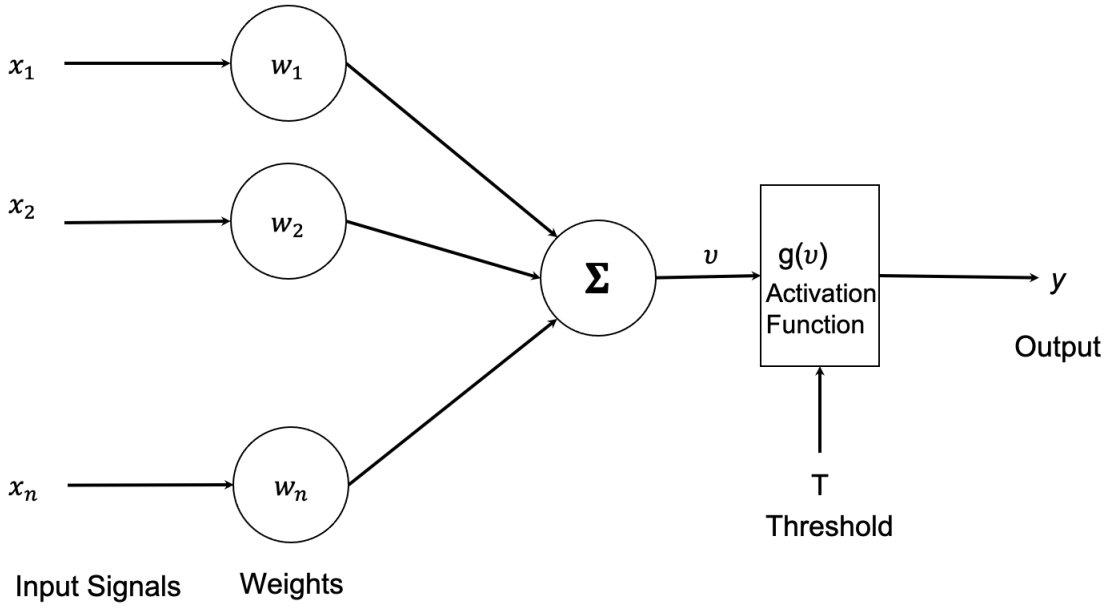


Figure 6.1 Multilayer perceptron model.
Source:[46]

The MLP model is expressed as follows

$$\nu = \sum_{i=0}^{n} x_i w_i = \mathbf{x^T w} \tag{6.1}$$

where

$$\mathbf{x} = \begin{bmatrix} x_1 & x_2 & \cdots & x_n \end{bmatrix}^T \in R^n$$

$$\mathbf{w} = \begin{bmatrix} w_1 & w_2 & \cdots & w_n \end{bmatrix}^T \in R^n$$

$$y = g(\nu) \tag{6.2}$$

There are several activation functions commonly used in machine learning algorithms. We mostly used the sigmoid function in this study given as [46]

$$g(\nu) = \frac{1}{1 + e^{-\nu}} \tag{6.3}$$

We also experimented hyperbolic tangent and other activation functions as discussed later [47].

In this study, we use the neural network that is comprised of the input layer, one or more hidden layers and the output layer as displayed in Figure 6.2.

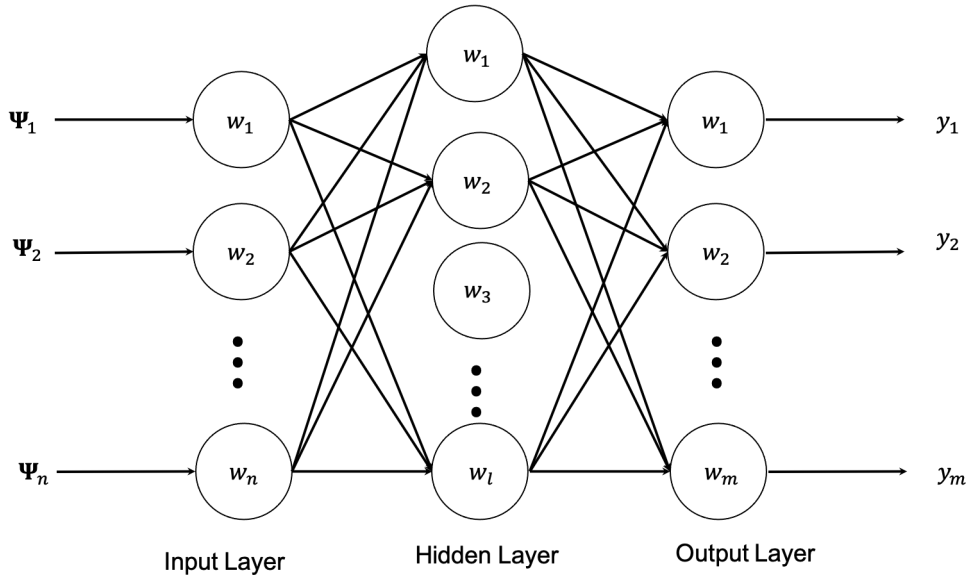


Figure 6.2 Artificial Neural Network Model.

We use the column vectors of the *joint time-frequency matrix* of the network statistics defined in Equation (5.3) as the input feature set to the neural network.

The network path numbers are labeled as the outputs of the neural network model. We have a training set of p pairs of $\mathbf{\Psi_i}, y_i$ where $\mathbf{\Psi_i}$ is a column vector of the *joint time-frequency matrix* and y_i is a network path index corresponding to that column vector where$i$=1 ... p. The network paths are identified by using the method described in 5.3. Herein, we intend to calculate the weights of the neurons that map a given *joint time-frequency vector* to the corresponding path index at the output.

The training process is an optimization problem expressed as [46]

$$min\frac{1}{2}\sum_{i=1}^{p}(y_i - \mathbf{\Psi_i^T w})^2$$

It can be rewritten as

$$minJ(w) = \frac{1}{2}\left\|\mathbf{y} - \mathbf{\Psi^T w}\right\|^2 \tag{6.4}$$

In Equation (6.4), we assumed that the activation function is the identity map to simplify the optimization model [46]. We use this supervised ANN algorithm in the *eigensubspace* where built-in dimension reduction is inherently achieved.

The optimization problem defined in (6.4) is solved to find the optimal sets of weights used by the neural network model to identify the paths. We used the feed forward with back propagation (FFBP) technique for learning as described in the references [48, 49, 47]. This technique is implemented as an unconstrained optimization that uses a gradient descent algorithm. The gradient descent algorithm used to find the optimal weights of the model is written as [50]

$$w^{(k+1)} = w^{(k)} + \alpha^{(k)}\Psi_i e^{(k)} \tag{6.5}$$

$$e^{(k)} = y_i - \Psi_i^T w^{(k)}$$

where

k is the index of iteration step,

$\alpha^{(k)}$ is the learning rate,

$e^{(k)}$ is the error between actual and predicted value.

It is noted that, we use the batch gradient descent (BGD) [50] in Equation (6.5) due to its superior performance over scaled conjugate gradient (SCG) [51], Levenberg-Marquardt (LM) [52, 53] and Bayesian regularization (BR) algorithms [54, 55]. It yields an unbiased estimate of gradients and theoretically guaranteed to converge to the global minimum along with a straight trajectory if the loss function is convex.

Similarly, we used the *tan sigmoid* (TS) and *Elliott Sigmoid* (ES) activation functions [52]. We also used the *minimization loss function* of *mean squared error* (MSE) and *cross-entropy error* (CEE) in our experimental studies and identified the best ones for the task at hand [56, 57].

We experimented with various optimization algorithms and activation functions to train of the network. We varied the number of neurons in the hidden layers and the size of training data features in the *eigensubspace*, Equation (5.3), to evaluate the convergence and performance of the algorithm.

Note that each $\mathbf{\Psi_i}$ column vector of Equation (5.3) corresponds to a network path identified with the label y_i. Using the network traffic data, we identified and labeled the large network flows traversing a certain path in the network during the training step. We had limited data for only 200 input vectors with 100 network paths from the actual network. This dataset was too small for these experiments. Therefore, we generated a dataset of 100 times larger size by adding 20% Gaussian noise to the actual network measurements $\mathbf{\Psi_i}$ and keeping the same output label for a given pair. This allows us to train the algorithm with sufficient amount of training data.

We found that more than 20,000 raw data samples are required to train our model, which uses a feature set of 30, with the SCG training algorithm. The model is

built using three layers, input layer, output layer and one hidden layer. There are 30 neurons in each layer, employing the TS activation and the CEE loss function. Figure 6.3 displays CEE error performance of this ANN and its convergence for the network path identification problem, using column vectors of the *joint time-frequency matrix* in Equation (5.3), as a function of training data size (epochs).

As a result of this investigation, we conclude that the use of *eigenfeatures* is an effective tool for network traffic classification. For the traffic characteristics present in the data used, the three layers (input, output and one hidden) model using the SGC algorithm with CEE loss function provides the fastest convergence.

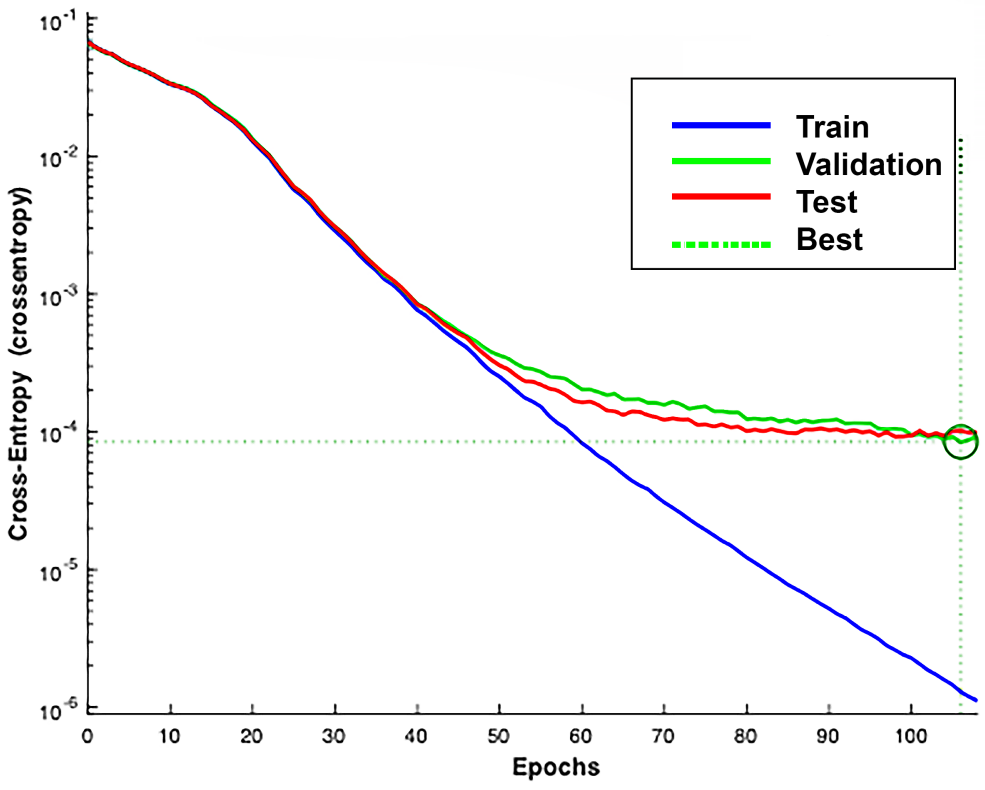


Figure 6.3 Model validation performance.

CHAPTER 7

NETWORK FORECASTING

7.1 Network Forecast in Eigensubspace

Using variations of *eigenvectors* and *eigencoefficients*, we train a linear regression model to predict their future values which are tied to link parameters of the network. A subset of the link traffic data provided by a large service provider is used for this study. The network characteristics are summarized as follows.

- There are 50 nodes in the network {n1, . . . , n50}
- These 50 nodes are fully connected to each other resulting in 2,450 links labeled as $\{ \; l_1 \quad l_2 \quad l_3 \quad \ldots \quad l_{2,450} \; \}$
- The link traffic measurement data (average bandwidth utilized) was collected for each link at 5-minute time intervals for one-hour duration (12 samples)
- The data was collected for the same hour of the day for 92 days (three months)
- Initially, we use a subset consisting of 24 links to run our experiments

In this analysis, we write link traffic measurements in the 3-D data array L(l, t, d) where

l – link index l =1, 2, . . . , N where N=24,

t – time index of 5-minute intervals for the same one-hour period in each day t =1, 2, . . . , T where T=12,

d – day index, d=1, 2, . . . , D where D=92.

This 3-dimensional (3-D) array might also be represented as $L^d_{l,t}$ with the same subscripts and superscript as defined. For convenience, we drop subscript and/or superscript and use the following notation.

L^d – matrix of link traffic measurements for day d, and for all 24 links, where each row vector of the matrix is twelve-dimensional and populated by the corresponding measurements of the 5-minute intervals for the given hour of the day. In this experiment, we only used the hour between 9:00 PM and 10:00 PM.

L_l^d – corresponds to a single row vector of L^d matrix as mentioned above for link l.

The empirical correlation matrix of size 24×24 for the link traffic on a given day d is computed as

$$\mathbf{R_L^d} = \begin{bmatrix} r_{11} & \cdots & r_{1N} \\ \vdots & \ddots & \vdots \\ r_{N1} & \cdots & r_{NN} \end{bmatrix} \tag{7.1}$$

where i and j are the link indices, and the pairwise correlations $\mathbf{r_{ij}} = E\left\{\mathbf{L_i^d L_j^d}\right\}$ are calculated based on the 12 measurements of the 5-minute intervals. The *eigenmatrix* $\mathbf{\Phi^d}$ of the network for day d is expressed as

$$\mathbf{R_L^d \Phi^d} = \mathbf{\Lambda^d \Phi^d} \tag{7.2}$$

Then, the daily *eigencoefficient* matrix $\mathbf{\Theta^d}$ of size $N \times T$ comprised of *eigencoefficient* vectors calculated for each 5-minute interval of 9:00PM to 10:00PM as its columns is defined in the current subspace $\mathbf{\Phi^d}$ as

$$\mathbf{\Theta^d} = \left[\mathbf{\Phi^d}\right]^T L^d \tag{7.3}$$

Now, we drop the superscript d for convenience and expand the equation where the first row of $\left[\mathbf{\Phi^d}\right]^T$is the principal *eigenvector* as follows

$$\mathbf{\Theta} = \begin{bmatrix} \theta_{11} & \cdots & \theta_{1T} \\ \vdots & \ddots & \vdots \\ \theta_{N1} & \cdots & \theta_{NT} \end{bmatrix} = \tag{7.4}$$

$$\begin{bmatrix} \Phi_{11} & \cdots & \Phi_{N1} \\ \vdots & \ddots & \vdots \\ \Phi_{1N} & \cdots & \Phi_{NN} \end{bmatrix} \begin{bmatrix} L_{11} & \cdots & L_{1T} \\ \vdots & \ddots & \vdots \\ L_{N1} & \cdots & L_{NT} \end{bmatrix}$$

The first column vector of the eigencoefficient matrix in Equation (7.4) corresponds to the first 5-minute time interval and written as

$$\begin{bmatrix} \theta_{11} \\ \vdots \\ \theta_{N1} \end{bmatrix} = \begin{bmatrix} \Phi_{11}L_{11} + \Phi_{21}L_{21} & \cdots & \Phi_{N1}L_{N1} \\ \vdots & \vdots & \vdots \\ \Phi_{1N}L_{11} + \Phi_{2N}L_{21} & \cdots & \Phi_{NN}L_{N1} \end{bmatrix} \tag{7.5}$$

Similarly, its second column corresponds to the second 5-minute time interval as

$$\begin{bmatrix} \theta_{12} \\ \vdots \\ \theta_{N2} \end{bmatrix} = \begin{bmatrix} \Phi_{11}L_{12} + \Phi_{21}L_{22} & \cdots & \Phi_{N1}L_{N2} \\ \vdots & \vdots & \vdots \\ \Phi_{1N}L_{12} + \Phi_{2N}L_{22} & \cdots & \Phi_{NN}L_{N2} \end{bmatrix} \tag{7.6}$$

and so on. As a result, the elements of the first row vector of the $N \times T$ *eigencoefficient* matrix where N=24 and T=12 in this example. Similarly, the second row corresponds to the second most significant *eigencoefficients* for those 5-minute time intervals, and so on. A subset of the rows of $\mathbf{\Theta}$ matrix given in Equation (7.4) are used for linear regression and prediction of link values as discussed later in this section.

From Equation (4.4), the traffic of link l, for day d, and the specific 5-minute time interval t is calculated as follows

$$\mathbf{L_{l,t}^{d}} = \sum_{\mathbf{k=1}}^{\mathbf{V}} \mathbf{\Theta_{kl}^{d} \Phi_{lk}^{d}} \tag{7.7}$$

where V is the predefined number of most significant vectors (dimension reduction based on explained variance, V$\leq$ N) running on the dimension index k as used to approximate the link traffic values in the subspace. When we drop the day superscript d, for t =1 and V=N (no dimension reduction) (7.7) is expanded as

$$\begin{bmatrix} L_{11} \\ \vdots \\ L_{N1} \end{bmatrix} = \begin{bmatrix} \theta_{11}\Phi_{11} + \theta_{21}\Phi_{21} & \cdots & \theta_{N1}\Phi_{N1} \\ \vdots & \vdots & \vdots \\ \theta_{11}\Phi_{1N} + \theta_{21}\Phi_{2N} & \cdots & \theta_{N1}\Phi_{NN} \end{bmatrix} \tag{7.8}$$

In Equation (7.8), only the first column of the Θ matrix has been used to compute the link traffic values for the first 5-minute time slot. Similarly, the link values for the second 5-minute time interval is computed by using the second column of the Θ matrix as follows

$$\begin{bmatrix} L_{12} \\ \vdots \\ L_{N2} \end{bmatrix} = \begin{bmatrix} \theta_{12}\Phi_{11} + \theta_{22}\Phi_{12} & \cdots & \theta_{N2}\Phi_{1N} \\ \vdots & \vdots & \vdots \\ \theta_{12}\Phi_{N1} + \theta_{22}\Phi_{N2} & \cdots & \theta_{N2}\Phi_{NN} \end{bmatrix} \tag{7.9}$$

Hence, the *link traffic measurement matrix* is defined for each 5-minute time slot, and for all the links when T=12 and N=24 as

$$L_{l,t}^{d} = \begin{bmatrix} L_{11} & L_{12} & \cdots & L_{1T} \\ \vdots & \vdots & & \vdots \\ L_{N1} & L_{N2} & \cdots & L_{NT} \end{bmatrix} \tag{7.10}$$

Now, we use the *eigenvectors* and *eigencoefficients* of the previous two days to predict the link traffic for the next day as [58, 59].

$$\widehat{\Phi}^{d+1} = \rho_1 \Phi^d + \rho_2 \Phi^{d-1} + \varepsilon_\Phi \tag{7.11}$$

$$\widehat{\Theta}^{d+1} = \vartheta_1 \Theta^d + \vartheta_2 \Theta^{d-1} + \varepsilon_\Theta \tag{7.12}$$

where ρ_1 and ρ_2 are the autoregressive order two, AR(2), regression model parameters used to predict the *eigenmatrix* Φ,

ε_Φ is the prediction (white) noise corresponding to the AR(2) process of the eigenmatrix Φ .

ϑ_1 and ϑ_2 are the AR(2) regression model parameters to predict the eigencoefficients matrix Θ

ε_Θ is the prediction (white) noise corresponding to the AR(2) process of the *eigencoefficients* matrix Θ.

By using the predicted $\widehat{\Phi}^{d+1}$ and $\widehat{\Theta}^{d+1}$ in Equation (4.4), we compute $\widehat{L}_l^{d+1}$ as expressed in Equation (7.13) below

$$\widehat{L}_l^{d+1} = \sum_{k=1}^{V} \widehat{\Theta}_{kl}^{d+1} \widehat{\Phi}_{lk}^{d+1} \tag{7.13}$$

where V is the reduced dimension in the *eigensubspace*, V$\leq$ N, to approximate the time series with the permissible prediction error (or explained variance).

The two network engineering applications that use the *joint time-frequency* features of a network as described in its *eigensubspace* are discussed in the following two sections of the dissertation.

CHAPTER 8

COMPUTATIONAL COMPLEXITY AND PERFORMANCE ANALYSIS

8.1 Computational Complexity

We analyzed the computational complexity of the proposed techniques and compared them with the prior work in [60]. The are two fundamental differences in the collection of time series data. First, the method in [10] collects O-D flows data which can be very large in a given network, versus in our proposed methods it is link level which scales only with the number of links in the network. For example, a network of n nodes, there are $n(n-1)/2$ possible O-D flows in a maximally connected network as compared to $(n-1)$ links in a minimally connected network. Second, the O-D flow data is collected by sampling the arriving packets, therefore it depends on the link bandwidth whereas link level data is collected at regular intervals of 5 minutes regardless of the link bandwidth.

Assuming these flows are going over 100Gbps links with a packet size of 1,500 bytes, there are 8.3M packets/sec passing through each interface. If a sampling rate N=1,000 is used, then there are 8.3K packet headers/second being collected per flow. With a standard 20 bytes packet header size, the total data collected is approximately 600Mbytes/hour. For n=10 nodes, there are 45 O-D flows, which results in 27 Gb/hour data collection volume. Clearly, this is an insurmountable challenge for using this technique. It calls for further increase in sampling rate, aggregation and pre-processing at local nodes. This also adds to the infrastructure overhead and bandwidth load on the network. For the proposed method, the same network with n=10, we have 9 links to monitor every 5 minutes, i.e., 108 samples/hour. Furthermore, this data is already being collected, and therefore, there is no overhead

to the network. Therefore, in terms of computational complexity, our method is several orders of magnitude simpler than O-D flow analysis.

It is noted that the O-D flows use Deep Packet Inspection (DPI) methods to infer the path of the flow using the origin and destination of the packets. They utilize an additional step of routing table lookup to identify the path in the network. The computation complexity analysis of this method is studied well. It is known to be computationally expensive and it cannot be deployed in very large networks [61, 62]. A performance comparison of the TomoGravity method used by O-D flows, the traditional PCA and the Deep architecture Long Short-Term Memory traffic matrix (DLSTM) prediction method was reported in [63]. They demonstrated the superiority of the PCA and LSTM based methods over the O-D flow (TomoGravity) with respect to the temporal relative prediction error.

8.2 Performance Comparison of Time Series and PCA Based Network Forecasts

In this Section, we develop an autoregressive, order two, AR(2), model to predict the link traffic of a network based on its historical data [14]. We built AR(2) predictors in the time domain and also in the *eigensubspace*, and compared their performances as explained in the following two Subsections.

A. Linear Regression in Time Domain

We employed a vector autoregressive model for linear regression in the time domain [64]. The 3-D data set $L_{l,t}^{d}$ introduced in 7.1 is labeled as L_t. We regressed the link traffic in the time domain for day d by using the data of the previous days $d-i$, $i = 1, 2, ..., M$ by using the vector autoregressive (VAR) model expressed as [64, 65]

$$\widehat{L}_t^d = C_t + \sum_{i=1}^{M} \psi_{t,i} L_t^{d-i} + \xi_t \tag{8.1}$$

where

C_t is a column vector of constant offsets,

$\psi_{t,i}$ is a vector of AR parameters of the model,

M is the order of the autoregressive model for t, M=2 for AR(2) model,

t is the time index of 5-minute intervals for the same one hour period in each day t =1, 2, ... , T where T=12,

ξ_t is a column vector of white noise.

Equation (8.1) is rewritten for AR(2) as follows

$$\widehat{L}_t^d = C_t + \psi_{t,1} L_t^{d-1} + \psi_{t,2} L_t^{d-2} + \xi_t \tag{8.2}$$

This VAR model is written for all the T column vectors of the time intervals as

$$\begin{aligned} \widehat{L}_1^d &= C_1 + \psi_{1,1} L_1^{d-1} + \psi_{1,2} L_1^{d-2} + \xi_1 \\ \widehat{L}_2^d &= C_2 + \psi_{2,1} L_2^{d-1} + \psi_{2,2} L_2^{d-2} + \xi_2 \\ &\vdots \\ \widehat{L}_{12}^d &= C_{12} + \psi_{12,1} L_{12}^{d-1} + \psi_{12,2} L_{12}^{d-2} + \xi_{12} \end{aligned} \tag{8.3}$$

The column vectors $\widehat{L}_1^d$ to $\widehat{L}_{12}^d$ are combined to form the forecasted link traffic matrix for 5-minute intervals for day d. Then, we estimate the parameters of AR(2) by using Equation (8.3).

Figure 8.1 displays the link measurements for the link $l = 24$ of day d along with their forecasted values by the AR(2) model in the time domain. It is seen from the Figure that the downward trend of the link traffic is forecasted with about 10% prediction error. We will repeat the same experiment in the *eigensubspace* in the next section and show its merit.

B. Linear Regression with *Joint Time-Frequency Features in Eigensubspace*

In this experiment, the regression is performed by using the $\mathbf{\Theta}$ and eigenmatrix $\mathbf{\Phi}$, and the link traffic is predicted as per (7.13). We used the same data set as the one used in the time-domain based experiment. The traffic measurements are mapped onto the currently defined *eigensubspace*. A small subset of *eigenflows* (dimension reduction) are used in the parameter calculations of AR(2) model. The *eigensubspace* based AR(2) predictor results are also displayed in Figure 8.1. It is observed from the Figure that the *eigensubspace* based linear regression model forecasts the network traffic more accurately than the time domain based prediction [60].

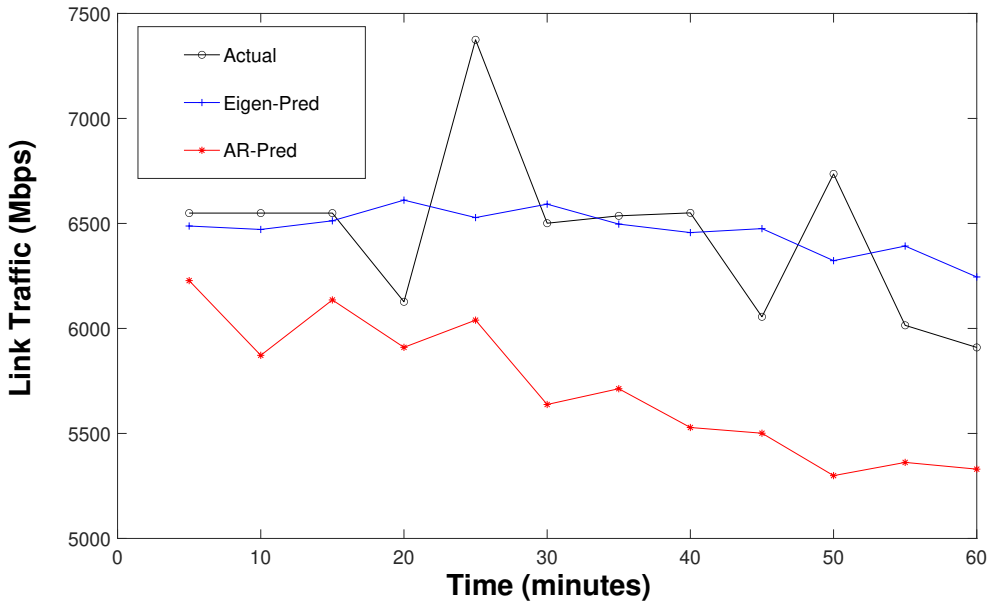


Figure 8.1 Performance comparisons of the two AR(2) based link traffic predictors in 5-minute measurement intervals for one-hour of historical data window size. The red curve is for the time-domain implementation, while the blue one is for the implementation in the *eigensubspace*.

CHAPTER 9

EXPERIMENTS AND RESULTS

This chapter presents the experiments conducted to evaluate the effectiveness of *eigenanalysis* and machine learning algorithms on the network traffic data.

9.1 Eigenanalysis Experiments and Results

This Section presents the *eigenanalysis* experiments and results using the network data.

9.1.1 Description of Data and Network

For the purpose of this study, the data was collected from the Internet2 network consisting of 10 nodes and 30 links. The link data collected for every five-minute interval is used as the 95 percentile value of the bandwidth utilized by a given link. The data was collected for a period of one week in order to capture daily and weekly behavior of the network. It is noted that, the ingress and egress statistics were collected for each link interface. It was found that the egress at one end of the link is the same as the ingress at the other end. Hence, only the egress statistics in were used our study.

Figure 5.2b shows the network topology as given by [64] consisting of the backbone links under consideration in this study. As indicated earlier, an eight-hour time window is chosen for calculation of the *eigensubspace (eigenvectors)* and a two-hour refresh rate. It is observed that the overall dynamics of the network does not change significantly in less than two hours to justify the recalculation of *eigensubspace (eigenvectors)*. A few experiments were conducted on this data to demonstrate the effectiveness of the proposed method as follows.

9.1.2 Experiment1

One link and multiple links down cases In this experiment, the effect of any link down on the rest of the network is demonstrated. Therefore, one of the link values is set to zero and its effect on network covariance and the resulting *eigenmatrix* is studied. In the next experiment, setting the measured traffic values for links 15, 22 and 26 to zero indicating that the links are down, and to see that those links are shown as anomalies by the proposed method.

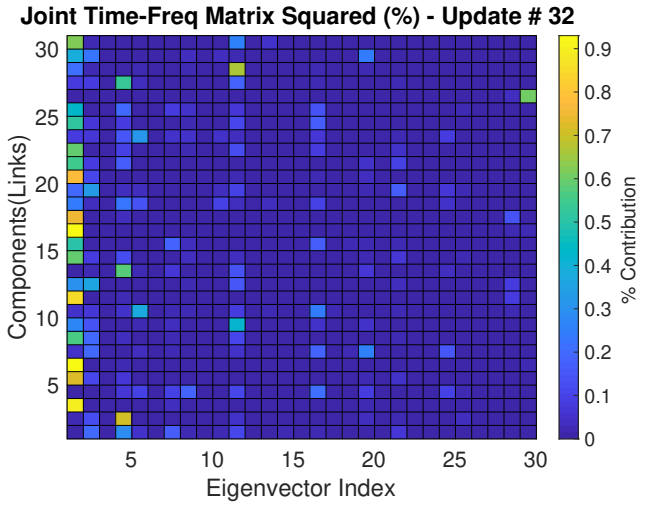


Figure 9.1 One link down representation in time.

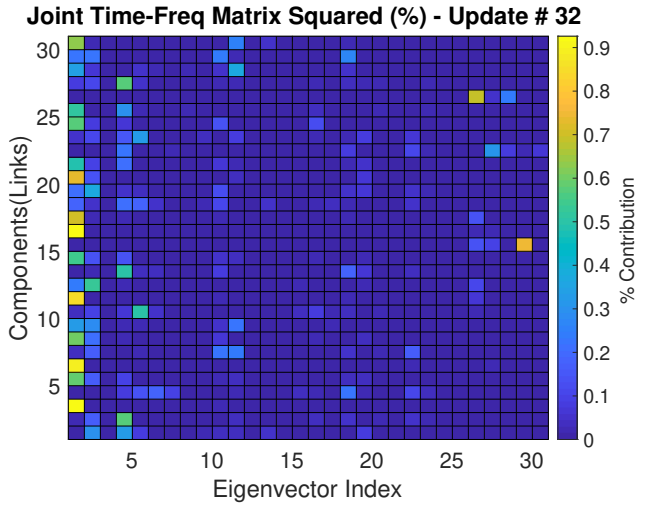


Figure 9.2 Three links down representation in time.

It is observed from Figures 9.1 and 9.2 that these three links going down are mapped as large percentage contribution in the least three significant *eigenvectors* in the method. This example clearly shows us the strength of the link-level granularity in traffic analysis. Moreover, it highlights the dynamically changing nature of non-stationarities in network statistics, where traditional *eigenanalysis* based techniques cannot capture.

9.2 Machine Learning Experiments and Results

In order to have a high level of confidence in the model, it is important to accurately choose the parameters of the model. For this, we run several experiments conducted to identify the optimal learning algorithm, number of hidden neurons and the type of activation functions. In this section, the findings and conclusions are presented.

9.2.1 Experiment 1

In this experiment, the following parameters were used as displayed in Table 9.1 below with a small number of samples.

Table 9.1 Levenberg-Marquardt Model Validation Performance

Exp. Attribute Description	Attribute Value
Number of input samples	2000
Number of neurons in hidden layer	20
Random Noise added to samples	20
Training Algorithm	Levenberg-Marquardt
Activation Function	Tan Sigmoid
Loss Function	Mean Squared Error

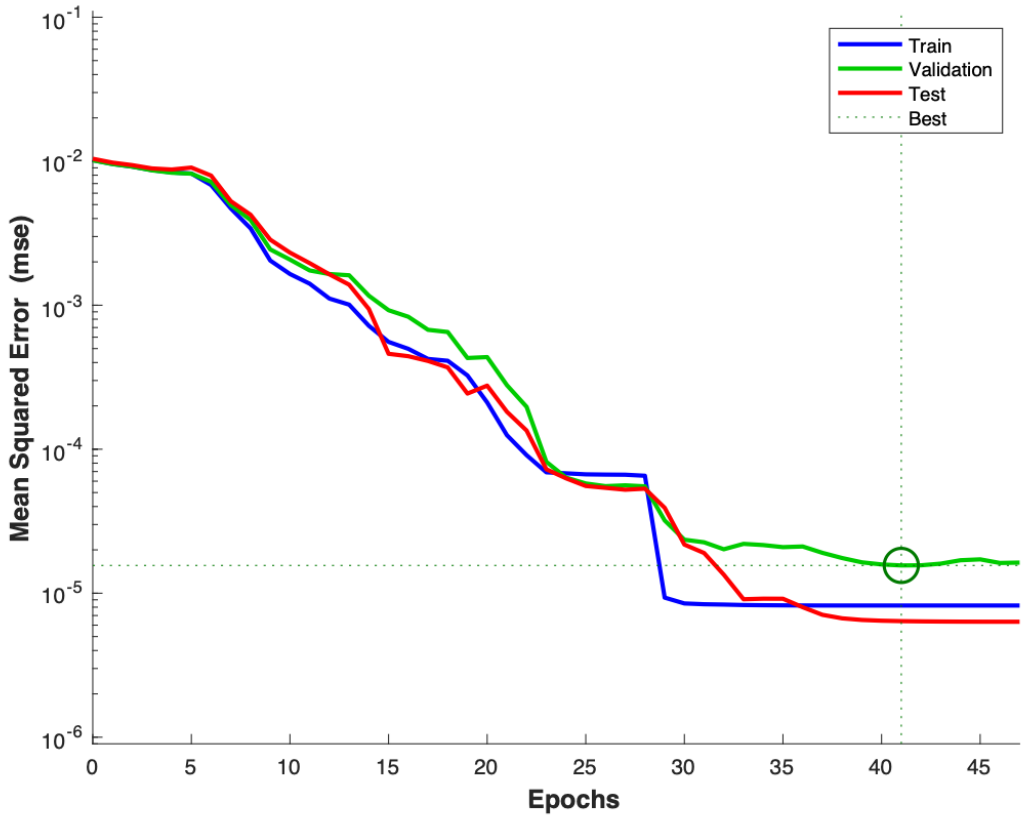


Figure 9.3 Levenberg-Marquardt Model Validation Performance.

9.2.2 Experiment 2

In this experiment, the following parameters were used as shown in Table 9.2. The sample size was increased and the number of neurons in the hidden layer and noise samples decreased.

Table 9.2 Scaled Conjugate Gradient with Small Samples

Exp. Attribute Description	Attribute Value
Number of input samples	5888
Number of neurons in hidden layer	10
Random Noise added to samples	10
Training Algorithm	Scaled Conjugate Gradient
Activation Function	Tan Sigmoid
Loss Function	Cross-Entropy Error

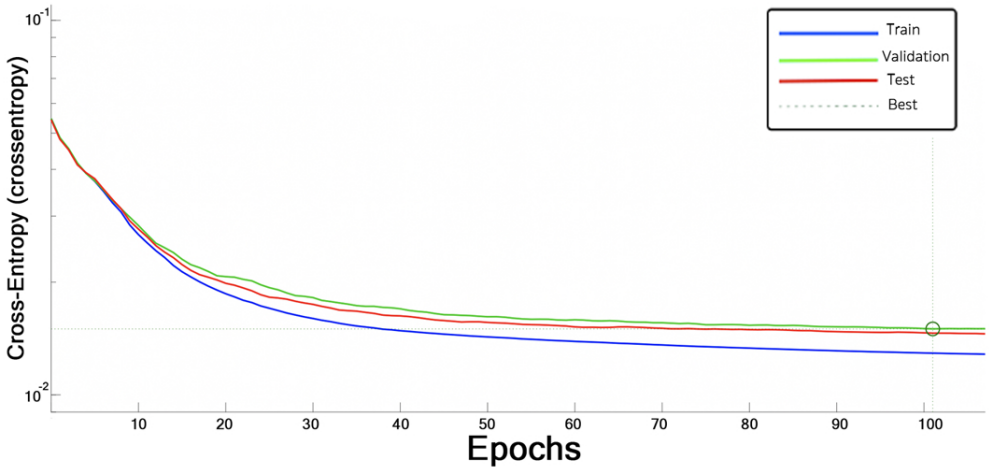


Figure 9.4 SCG with small samples - Model Validation Performance.

9.2.3 Experiment 3

In this experiment, the following parameters were used as shown in Table 9.3. The sample size was increased and the number of neurons in the hidden layer and noise samples were doubled from the previous experiment.

Table 9.3 SCG with Large Samples

Exp. Attribute Description	Attribute Value
Number of input samples	23552
Number of neurons in hidden layer	20
Random Noise added to samples	20
Training Algorithm	Scaled Conjugate Gradient
Activation Function	Tan Sigmoid
Loss Function	Cross-Entropy Error

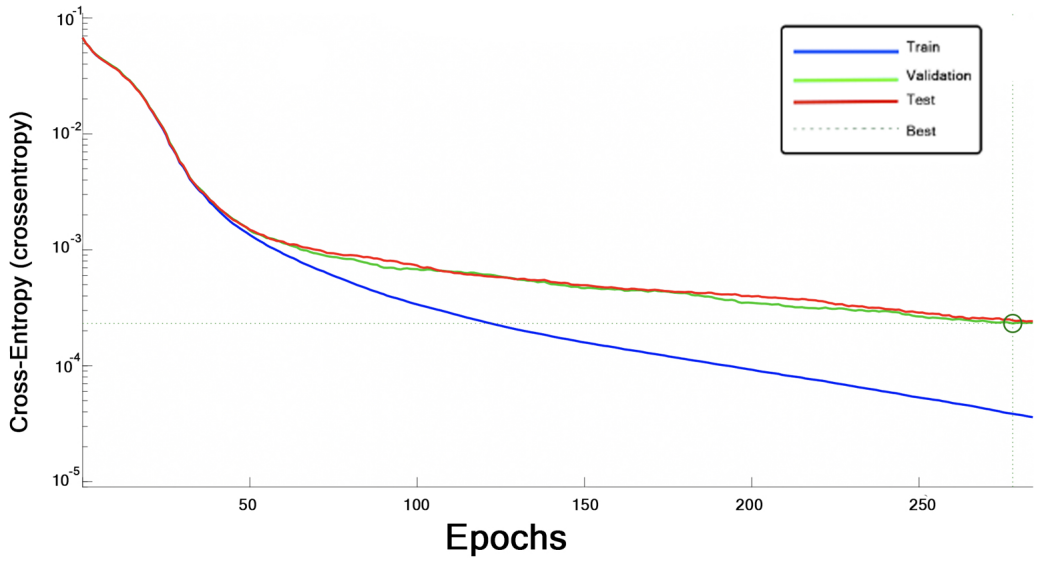


Figure 9.5 SCG with large samples - Model Validation Performance.

9.2.4 Experiment 4

In this experiment, the following parameters were used, which are the same as in the previous experiment, and the experiment was repeated to make sure it works with these values.

Table 9.4 SCG with Large Samples (repeat experiment)

Exp. Attribute Description	Attribute Value
Number of input samples	23552
Number of neurons in hidden layer	20
Random Noise added to samples	20
Training Algorithm	Scaled Conjugate Gradient
Activation Function	Tan Sigmoid
Loss Function	Cross-Entropy Error

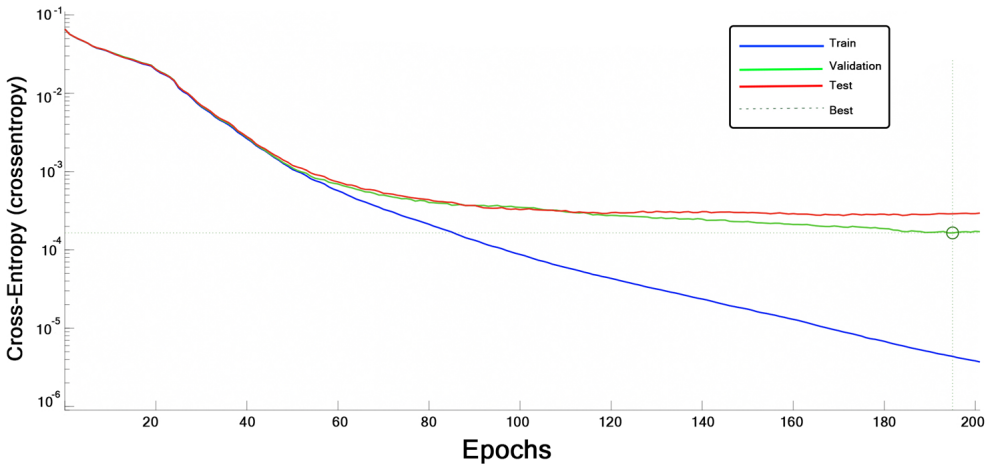


Figure 9.6 SCG with large samples (repeated) - Model Validation Performance.

9.2.5 Experiment 5

In this experiment the following parameters were used as shown in Table 9.5. The number of neurons in the hidden layer were increased from the previous experiment.

Table 9.5 SCG with 30 Neurons in the Hidden Layers

Exp. Attribute Description	Attribute Value
Number of input samples	23552
Number of neurons in hidden layer	30
Random Noise added to samples	20
Training Algorithm	Scaled Conjugate Gradient
Activation Function	Tan Sigmoid
Loss Function	Cross-Entropy Error

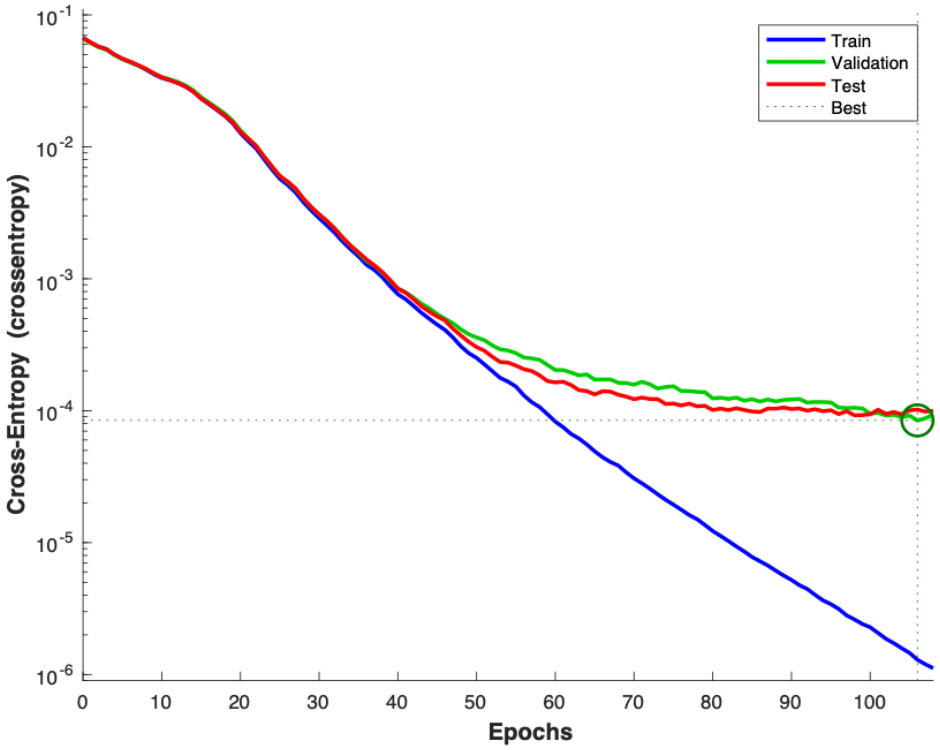


Figure 9.7 SCG with 30 neurons in the hidden layers - Model Validation Performance.

9.2.6 Experiment 6

In this experiment, the following parameters were used as shown in Table 9.6. The number of neurons in the hidden layer were increased to 40 hidden layers from the previous experiment.

Table 9.6 SCG with 40 Neurons in the Hidden Layers

Exp. Attribute Description	Attribute Value
Number of input samples	23552
Number of neurons in hidden layer	40
Random Noise added to samples	20
Training Algorithm	Scaled Conjugate Gradient
Activation Function	Tan Sigmoid
Loss Function	Cross-Entropy Error

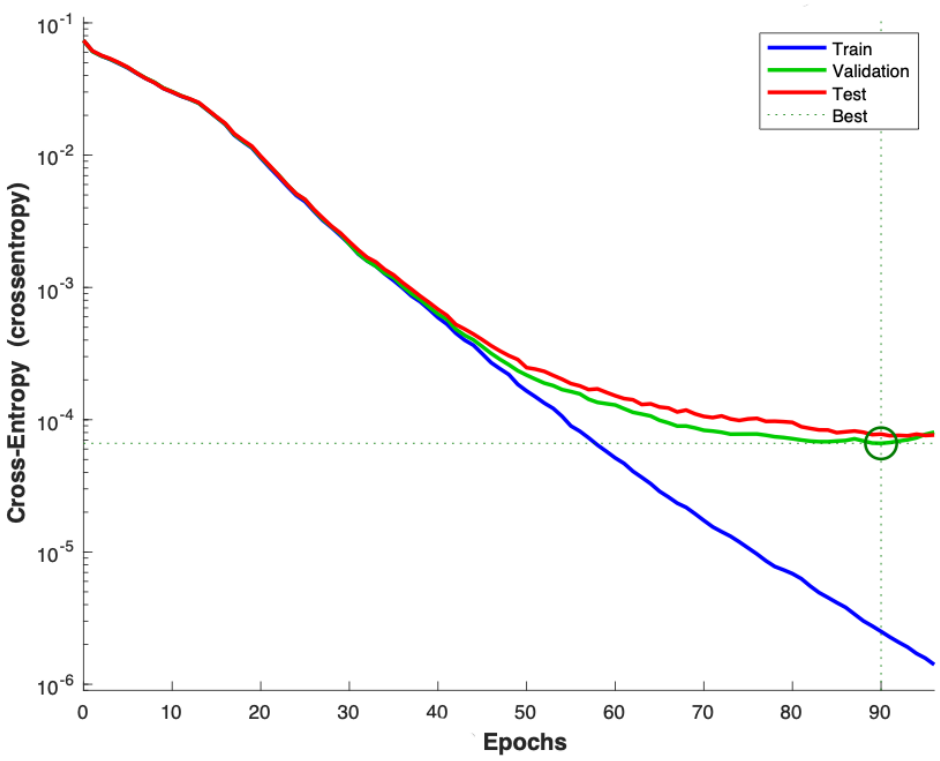


Figure 9.8 SCG with 40 neurons in the hidden layer - Model Validation Performance.

9.2.7 Experiment 7

In this experiment, the following parameters were used as displayed in Table 9.7. The training algorithm was changed to Bayesian Regularization and the loss function to Mean Squared Error.

Table 9.7 Bayesian Regularization - Model Validation Performance

Exp. Attribute Description	Attribute Value
Number of input samples	23552
Number of neurons in hidden layer	30
Random Noise added to samples	20
Training Algorithm	Bayesian Regularization
Activation Function	Tan Sigmoid
Loss Function	Mean Squared Error

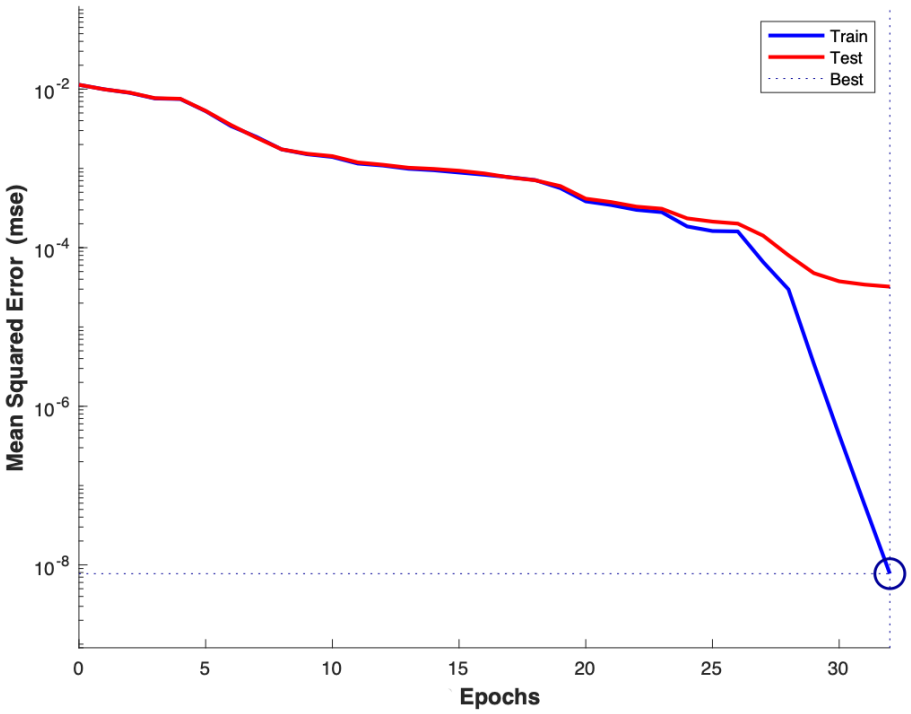


Figure 9.9 Bayesian Regularization - Model Validation Performance.

CHAPTER 10

CONCLUSIONS

Origin-Destination (O-D) flow based *eigensubspace* methods are computationally costly. More importantly, they lack the link-level resolution required to efficiently track the dynamics of a heterogeneous network [40]. We developed a granular method to monitor the variations of network traffic to identify anomalies. The method utilizes the *eigendecomposition* of the empirical correlation matrix of link traffics in a given network. This *joint time-frequency* interpretation of *eigensubspace* representation of traffic data provides additional insights to better understand the overall network behavior as well as the individual links. We demonstrate in this dissertation that the link-level focus on the statistical analysis leads to identify local anomalies as the building blocks of elephant flows in the network. We also show that the *eigensubspace* based network forecast outperforms the methods that use the time domain based measurements and predictions.

ANN with FFBP was employed to identify network paths by using *joint time-frequency* (*eigencoefficient* weighted) *eigenvectors* as features. Specifically, for the network model with 30 inputs (links/vectors) and 100 possible outputs (network paths), the following observations have been made:

1. A training set of approximately 25,000 inputs is required to accurately train an ANN model for network path identification.

2. The same number of neurons are required in the hidden layers as the number of neurons in the input layer.

3. The performance of the learning algorithm does not degrade with up to 20% of random noise in the input vector components.

4. The Scaled Conjugate Gradient (SCG) algorithm with Cross-entropy loss function (CEE) shows better performance and speed of convergence in our experiments for the given data set. The rate of convergence differs from one sample data set to another, but always converges.

5. Elliott Sigmoid (ES) activation function works faster than tan sigmoid (TS) due to elimination of the exponential (e) factor in the function.

It is concluded that an artificial neural network with feed forward and backward propagation model using scaled conjugate gradient optimization can be used as an effective machine learning technique to identify large network flow paths using the proposed *joint time-frequency matrix* as its input data features in the *eigensubspace*. This novel technique provides new insights to the operators for automating the network engineering operations.

It is also concluded that *eigensubspace* based forecasting is computationally more efficient and accurate than time domain based techniques employing autoregressive models.

REFERENCES

[1] A. N. Akansu and R. A. Haddad. *Multiresolution Signal Decomposition: Transforms, Subbands, and Wavelets, San Diego*. CA Academic Press, 1992.

[2] R. A. Haddad, A. N. Akansu, and A. Benyassine. Time-frequency localization in transforms, subbands, and wavelets: a critical review. *Optical Engineering*, 32(7):1411–1430, 1993.

[3] R. Yan and R. Liu. Principal component analysis based network traffic classification. *Journal of Computers*, 9(5):1234–1240, May 2014.

[4] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot. A pragmatic definition of elephants in internet backbone traffic. In *Proceedings of the 2nd Association for Computing Machinery (ACM) Special Interest Group on Data Communication (SIGCOMM) Workshop on Internet measurement*, pages 175–176, November 2002.

[5] K. Papagiannaki, N. Taft, S. Bhattacharyya, P. Thiran, K. Salamatian, and C. Diot. On the feasibility of identifying elephants in internet backbone traffic. *Sprint Labs, Sprint ATL, Tech. Rep. TR01-ATL-110918*, November 2001.

[6] T. Jin, C. Tracy, M. Veeraraghavan, and Z. Yan. Traffic engineering of high-rate large-sized flows. In *Proceedings 14th IEEE International Conference on High Performance Switching and Routing (HPSR)*, pages 128–135, July 2013.

[7] R. Hayashi, T. Miyamura, K. Shiomoto, and S. Urushidani. Impact of traffic correlation on the effectiveness of multilayer traffic engineering. *Proceedings of Asia-Pacific Conference on Communications*, October 2005.

[8] A. Coates, A. O. Hero, R. Nowak, and B. Yu. Internet tomography. *IEEE Signal Processing Magazine*, 19.3:47–65, May 2002.

[9] M. Joshi and T. Hadi. A review of network traffic analysis and prediction techniques. *arXiv preprint*, July 2015. arXiv:1507.05722.

[10] A. Lakhina. *Network-Wide Traffic Analysis: Methods and Applications*. Ph.d. thesis, Boston University, Boston, Massachusetts, 2007.

[11] X Xing, X Zhou, H Hong, W Huang, K Bian, and K Xie. Traffic flow decomposition and prediction based on robust principal component analysis. *International Conference on Intelligent Transportation Systems*, 59:2219–2224, 2015.

[12] E. J. Candès, X. Li, Y. Ma, and J. Wright. Robust principal component analysis. *Journal of the ACM (JACM)*, 58(3):1–37, 2011.

[13] R. H. Filho and J. E. B. Maia. Network traffic prediction using pca and k-means. In *2010 IEEE Network Operations and Management Symposium-NOMS*, pages 938–941, 2010.

[14] Z. M. Fadlullah and et al. State of the art deep learning Evolving machine intelligence toward tomorrowś intelligent network traffic control systems. *IEEE Communications Surveys & Tutorials*, 19(44):2432–2455, 2017.

[15] R. Boutaba, M. A. Salahuddin, N. Limam, S. Ayoubi, N. Shahriar, F. Estrada-Solano, and O. M. Caicedo. A comprehensive survey on machine learning for networking: evolution, applications and research opportunities. *Journal of Internet Services and Applications*, 9(1):1–99, 2018.

[16] V Paxson, G Almes, J Mahdavi, and M Mathis. Framework for IP Performance Metrics - RFC 2330. *IETF, Standard Specification*, May 1998. Accessed on: June 26, 2021. [Online]. Available: `https://datatracker.ietf.org/doc/html/rfc2330`.

[17] ITU-T. Y.1731. Recommendation G.8013/Y.1731, ITU-T, Aug 2015. Accessed on: Oct 18, 2021. [Online]. Available: `https://www.itu.int/rec/T-REC-Y.1731/en`.

[18] IS Association et al. IEEE 802.3-2018 - IEEE Standard for Ethernet (Revision of IEEE 802.3-2015), 2018.

[19] K McCloghrie and M. T. Rose. Management Information Base for Network Management of TCP/IP-based internets: MIB-II. STD 17, RFC 1213, March 1991.

[20] R. Raghunarayan. Management Information Base for the Transmission Control Protocol (TCP). RFC 4022, March 2005.

[21] B. Claise, G. Sadasivan, V. Valluri, and M. Djernaes. Cisco Systems NetFlow Services Export Version 9. RFC 3954, October 2004.

[22] Juniper Networks. Juniper Flow Monitoring. *Juniper Networks Application Note, Online Website*, 2011.

[23] S Shalunov, B Teitelbaum, A Karp, J Boote, and M Zekauskas. A One-way Active Measurement Protocol (OWAMP)-RFC 4656. *IETF, Standard Specification*, 2006. Accessed on: June 26, 2021. [Online]. Available: `https://www.rfc-editor.org/rfc/rfc4656.txt`.

[24] J. Babiarz, R. M. Krzanowski, K. Hedayat, K. Yum, and A. Morton. A Two-Way Active Measurement Protocol (TWAMP). *IETF, Standard Specification*, October 2008. Accessed on: June 26, 2021. [Online]. Available: `https://datatracker.ietf.org/doc/html/rfc5357`.

[25] R. Castro, M. Coates, G. Liang, R. Nowak, and B. Yu. Network Tomography: Recent Developments. *Statistical Science*, 19(3):499 – 517, 2004.

[26] Y. Vardi. Network tomography: Estimating source-destination traffic intensities from link data. *Journal of the American Statistical Association*, 91(433):365–377, 1996.

[27] G. Hooten. Real-world uses of route analytics. *Online Website*, 2004. Accessed on: Oct 18, 2021. [Online]. Available: `https://archive.apnic.net/meetings/21/docs/sigs/routing/routing-pres-hooten-analytics.pdf`.

[28] WhatsUpGold. Progress whatsup gold. *Online Website*, 2021. Accessed on: June 26, 2021. [Online]. Available: `https://www.whatsupgold.com/`.

[29] G. Lyon. Nmap: The network mapper–free security scanner. *Nmap.org*, 2016. Accessed on: June 26, 2021. [Online]. Available: `https://https://nmap.org/`.

[30] L. Cotterell. Network monitoring tools. *Stanford University*, 2021. Accessed on: June 26, 2021. [Online]. Available: `https://www.slac.stanford.edu/xorg/nmtf/nmtf-tools.html`.

[31] Wikipedia. Comparison of network monitoring systems. *Online Website*, 2000. Accessed on: July 5, 2021. [Online]. Available:`http://en.wikipedia.org/wiki/Comparison_of_network_monitoring_systems`.

[32] IBM. IBM software and solutions. *IBM Manual*, 2000. Accessed on: July 5, 2021. [Online]. Available: `http://www-03.ibm.com/software/products/us/en/netcool-network-management/`.

[33] T. Oetiker and D. Rand. MRTG: The Multi Router Traffic Grapher. In *LISA*, volume 98, pages 141–148, 1998.

[34] The OpenNMS Group. Meridian OpenNMS - Open Source Network Monitoring Platform. *Online Website*, 2000. Accessed on: July 3, 2021. [Online]. Available: `https://www.opennms.com/`.

[35] Wikipedia. HP openview. *Online Website*, 2000. Accessed on: July 3, 2021. [Online]. Available: `https://en.wikipedia.org/wiki/HP_OpenView`.

[36] Solarwinds. IT service management without the friction. *SolarWinds User Manual*, 2000. Accessed on: July 5, 2021. [Online]. Available: `http://www.solarwinds.com/`.

[37] P. Aitken, B. Claise, and B. Trammell. Specification of the IP Flow Information Export (IPFIX) Protocol for the Exchange of Flow Information. RFC 7011, September 2013.

[38] SevOne. Sevone network data platform. *Online Website*, 2000. Accessed on: July 5, 2021. [Online]. Available:`https://www.sevone.com/products`.

[39] A. Lakhina, M. Crovella, and C. Diot. Mining anomalies using traffic feature distributions. *ACM SIGCOMM Computer Communication Review*, 35(4):217–228, 2005.

[40] Internet2. perfSONAR User Guide. *Online Website*, April 2016. Accessed on: July 5, 2021. [Online]. Available: `https://docs.perfsonar.net/index.html`.

[41] U. DemÅ¡ar, P. Harrisand C. Brunsdon, A. S. Fotheringham, and S. McLoone. Principal component analysis on spatial data: An overview. *Annals of the Association of American Geographers*, 103:1(103:1):106 –128, 2013.

[42] G. H. Golub and H. A. Van der Vorst. Eigenvalue computation in the 20th century. *Journal of Computational and Applied Mathematics - Special Issue on Numerical Analysis*, III(1-2):35 – 65, November 2000.

[43] MATLAB. MATLAB Vector Autoregression (VAR) Models. *Online Website*, April 2019. Accessed on: July 5, 2021. Available: `https://www.mathworks.com/help/econ/introduction-to-vector-autoregressive-var-models.html`.

[44] J. Liu, P. Gao, J. Yuan, and X. Du. An effective method of monitoring the large-scale traffic pattern based on rmt and pca. *Journal of Probability and Statistics*, 2010.

[45] F. F. Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. *American Journal of Psychology*, 76(No.4), 1963.

[46] E. K. P. Chong and S. H. Zak. *An Introduction to Optimization,3rd Ed. New York*, pages 247–265. John Wiley & Sons, 2008.

[47] M. J. Realff J. H. Lee, J. Shin. Machine learning: Overview of the recent progresses and implications for the process systems engineering field. *Computers & Chemical Engineering*, 114(No. 4):111–121, 2018.

[48] Y. Chauvin and D. E. Rumelhart. *Backpropagation: Theory, Architectures, and Applications*. Lawrence Erlbaum Associates Inc, 1995.

[49] J. Pospichal D. Svozil, V. Kvasnicka. Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39:43–66, 1997.

[50] Y. Bengio. Practical recommendations for gradient-based training of deep architectures. *Neural Networks: Tricks of the Trade*, 7700, 2012.

[51] M. F. Moller. A scaled conjugate gradient algorithm for fast supervised learning. *Neural Networks*, 6:525–533, 1993.

[52] D. Marquardt. An algorithm for leastsquares estimation of nonlinear parameters. *SIAM Journal on Applied Mathematics*, 11(No. 2):431–441, 1963.

[53] M. T. Hagan and M. Menhaj. Training feedforward networks with the marquardt algorithm. *IEEE Transactions on Neural Networks*, 5(No.6):989–993, 1999.

[54] D. J. C. MacKay. Bayesian interpolationÂ . *Neural computation*, 4(No.3):415–447, 1992.

[55] F. D. Foresee and M. T. Hagan. Gauss-newton approximation to bayesian learning. In *Gauss-Newton Approximation to Bayesian Learning*. Proceedings of the International Joint Conference on Neural Networks, June 1997.

[56] T. P. Vogl, J.K. Mangis, A.K. Rigler, W.t. Zink, and D.L. Alkon. Accelerating the convergence of the backpropagation method. *Biological Cybernetics*, 59:257–263, 1998.

[57] C. M. Bishop. *Pattern Recognition and Machine Learning*, pages 235–236. Springer Verlag New York NY, 2006.

[58] E. Zivot and J. Wang. Vector autoregressive models for multivariate time series. *Modeling Financial Time Series with SPLUS*, pages 385–429, 2006.

[59] D. Hawkins. On the investigation of alternative regressions by principal component analysis. *Journal of the Royal Statistical Society. Series C AppliedStatistics*, 22(3):275–286, 1973.

[60] I. Lateef and A. N. Akansu. Link-level interpretation of eigenanalysis for network traffic flows. In *Proceedings of Conference on Information Sciences and Systems (CISS)*, pages 1–6, 2017.

[61] C. Niccolò, C. Luigi, and R. Fulvio. Optimizing deep packet inspection for high-speed traffic analysis. *Journal of Network and Systems Management*, 19(1):7–31, 2011.

[62] C. Niccolo, E. Alice, G. Francesco, R. Fulvio, and S. Luca. An experimental evaluation of the computational cost of a dpi traffic classifier. In *GLOBECOM 2009-2009 IEEE Global Telecommunications Conference*, pages 1–8. IEEE, 2009.

[63] Z. Jianlong, Q. Hua, Z. Jihong, and J. Dingchao. Towards traffic matrix prediction with lstm recurrent neural networks. *Electronics Letters*, 54(9):566–568, 2018.

[64] Internet2. Internet2 network infrastructure topology. *Online Website*, April 2019. Accessed on: July 5, 2021. [Online]. Available: `https://internet2.edu/network/state-and-regional-r-e-networks/`.

[65] H. Lutkepohl. *Stable Vector Autoregressive Processes*, pages 13–16. Springer, New York, NY, 2005.