DISTANCE APPROXIMATIONS BETWEEN HIGH AND MULTI–DIMENSIONAL STRUCTURES

by

Murat Semerci

B.S., Electrical & Electronics Engineering, Boğaziçi University, 2005

B.S., Computer Engineering, Boğaziçi University, 2005

M.S., Computer Engineering, Boğaziçi University, 2007

M.S., Computer Science, Rensselaer Polytechnic Institute, 2010

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

Graduate Program in Computer Engineering

Boğaziçi University

2019

DISTANCE APPROXIMATIONS BETWEEN HIGH AND MULTI–DIMENSIONAL STRUCTURES

APPROVED BY:

Prof. Ali Taylan Cemgil (Thesis Supervisor)	.....
Assist. Prof. Ebru Arısoy-Saraçlar	.....
Assist. Prof. Serap Kırbız-Şimşek	.....
Assoc. Prof. Arzucan Özgür	.....
Prof. Tuna Tuğcu	.....

DATE OF APPROVAL: 25.06.2019

ACKNOWLEDGEMENTS

I would like to thank my thesis supervisor, Prof. Ali Taylan Cemgil, for providing me my "fourth" attempt to pursue a PhD degree. There is no formal way to express my gratitude. He has always boosted my self-confidence and courage. I consider him not only as my academic advisor but also as a senior colleague who always points out the right direction and as a close friend who harshly criticizes me whenever I am wrong. Prof. Bülent Sankur, inarguably, has a great influence on me. I recognize him as my step thesis supervisor. He has always been so kind to devote me his precious time whenever I needed. He has been there to give me the impulse to drive forward whenever and wherever I got stuck. Prof. Cem Ersoy has also been so polite and understanding to track my progress throughout all my PhD adventure. He has contributed to my studies with his valuable comments and feedback. I also thank my thesis committe for their patience, and feedback they have committed for this thesis.

During this so-long journey (a total of 9 years), I have been lucky to meet great companions. I can never forget Dr. Doğaç Başaran and Dr. Nazlı Güney for supporting me at one of the biggest turmoil in my life. At the right time and point, they encouraged me to hold on and continue to pursue my PhD. I want to thank Barış Kurt, Yusuf Taha Ceritli and Mehmet Yamaç for their collaboration and cooperation. I would also announce my thankfulness to all the other members of Perceptual Intelligence and Media Lab. for their help, support and friendship.

The most significant round is definitely reserved for my family. My parents, Garip and Nuriye Semerci, have raised me to be the man I am now. They unquestionably support all my decisions, no matter what I decide to do. Feeling their presence and support behind me provide me all the power, patience and endurance I need. My sister, Selma Kalkan, and her esteemed husband, Adem Kalkan, have always been there, even in the darkest times, being my lighthouse to the exit. Their children, my niece and nephew, Havva Özge and Umut Efe have proven me the other tastes and enjoyments of the life. My parents-in-law, Harbiye Yalabuk and Ali Yalabuk, and brother-in-law,

Ufuk Yalabuk, have always believed in me and they never doubt my ability to succeed.

Undisputedly, my beloved wife, Elif Semerci, gets my highest gratitude. She has been the center of my universe since our marriage and she has been along with me during all my academic journey. She has always boosted me when I was down and she had the faith that one day I would be able to accomplish my PhD. She has cooled me down whenever I was full-charged to explode. The words are unutterable to express how lucky I feel to have her in my life. I can explicitly say that she has a substantial moral share on the creation of this thesis. Thanks for just being you!

ABSTRACT

DISTANCE APPROXIMATIONS BETWEEN HIGH AND MULTI–DIMENSIONAL STRUCTURES

In this thesis, we focus on *distance approximation* methods between high and multi-dimensional structures and their applications. Two novel methods using distance approximations are proposed and they are applied to anomaly detection in cyber security (Distributed Denial of Service -DDoS- attack and attacker detection) and tensor decomposition in object retrieval (image and video classification on scarce data). At first, we consider an autonomous cyber security system that consists of two components: A monitor for detection of DDoS attacks and a discriminator for detection of users in the system with malicious intents. A novel adaptive real time change-point detection model that tracks the changes in the Mahalanobis distances between sampled feature vectors in the monitored system accounts for possible DDoS attacks. A clustering model that runs over the similarity scores of behavioral patterns between the users is used for segregating the malicious from the innocent. Secondly, we propose a discriminative tensor decomposition with large margin (LMTD), which is a distance based model that finds the projection directions where the nearest neighbor classification accuracy is improved over the projected instances. We experiment the cyber security system in a simulated SIP communication environment. Both the attack and attacker detection components are compared with some competitors in the literature. The tensor decomposition is applied to the image and video retrieval problem, where the data is scarce, and its performance also is compared with other decomposition methods. The experimental results are reported for both applications. It is shown that the proposed methods perform higher accuracy rates than their competitors.

ÖZET

YÜKSEK VE ÇOK BOYUTLU YAPILAR ARASINDAKİ MESAFE YAKLAŞIMLARI

Bu tezde, yüksek ve çok boyutlu yapılar arasındaki *mesafe yakınsama* yordamlarına ve onların uygulamalarına odaklanıyoruz. İki yeni mesafe yakınsama kullanan yöntem önerilmekte ve onlar siber güvenlikte sıradışılık tespitine (Dağıtılmış Hizmet Reddi (DHR) saldırı ve saldırgan tespiti) ve nesne geri çağırmada gerey (tensör) ayrıştırmaya (kıt veride imge ve görüntü sınıflandırma) uygulanmaktadır. İlk olarak, iki bileşenden oluşan bir özerk (otonom) siber güvenlik sistemi düşünüyoruz: DHR saldırısı tespiti için bir izleyici ve sistemdeki kötü niyetli kullanıcıların tespiti için bir ayırt edici. Örneklenmiş öznitelik vektörleri arasındaki Mahalanobis uzaklığının değişimini takip eden bir özgün uyarlanabilir değişim noktası tespit modeli izlenilen dizgedeki olası DHR saldırılarının hesabını yapmaktadır. Kullanıcıların davranışsal örüntüleri arasındaki benzerlik skorları üstünde koşan bir öbekleme modeli kötü niyetlileri masumlardan ayırmakta kullanılır. İkinci olarak, izdüşülmüş örnekler üzerinde en yakın komşu sınıflandırma doğruluğunu iyileştiren izdüşüm yönlerini bulan uzaklık tabanlı bir geniş kenar paylı ayrımcı gerey ayrıştırması (GKAGA) öneriyoruz. Siber güvenlik dizgesini benzetilmiş SIP haberleşme ortamında deniyoruz. Hem saldırı hem saldırgan tespiti bileşenleri yazındaki bazı rakipler ile karşılaştırılmaktadır. Gerey ayrıştırma, kıt veri durumunda, imge ve görüntü geri çağırma sorununa uygulanmakta ve başarımı diğer ayrıştırma yöntemleri ile karşılaştırılmaktadır. Her iki uygulama için deneysel sonuçlar rapor edilir. Önerilen metotların rakiplerinden daha yüksek doğruluk oranı sergiledikleri gösterilmektedir.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF SYMBOLS

c	Constant used in thresholding
$\mathbf{C}$	Label vector
C	Margin value
d	Number of dimensions in the input space
dg_q	Degree of q^{th} active user
$\mathbf{D}$	Degree matrix
$D(o_i, o_j)$	Distance between the objects o_i and o_j
$D_{ld}(\mathbf{A}, \mathbf{B})$	Logarithmic determinant divergence (LogDet) between two matrices
e	Number of dimension in the reduced space
$f(\mathbf{M}\|\mathbf{x}_n : \mathbf{x}_{n-k-1})$	Loss function defined on $\mathbf{M}$ given vectors from $\mathbf{x}_{n-k-1}$ to $\mathbf{x}_n$
$\mathbf{I}$	Identity matrix
I_n	Number of dimensions in mode-n
k	Time frame size or number of nearest neighbors
$\mathbf{K}$	Kernel matrix
K	Number of tensor objects in the data set
$K(o_i, o_j)$	Kernel function of the objects o_i and o_j
$\mathbf{L}$	Laplacian or projection matrix
L	Associated loss function
M	Number of instances in the data set
$\mathbf{M}$	Mahalanobis metric
N	Order of tensor $\mathcal{X}$
$Ne_p(o)$	Set of p-nearest neighbors of object o
P	Number of projections
P_r	Number of SIP messaging activities of r^{th} active user (u_r)
$\mathbf{S}_+$	Positive semi-definite metric space
$\mathbf{S}_{++}$	Positive definite metric space
t	Timestamp
tr($\mathbf{A}$)	Trace of a matrix

u_r	Active user with the index value r in an observation interval
T_{p_q,p_r}	Alignment score of two sequences with lengths p_q and p_r
$\boldsymbol{u}^{(n)}$	Projection unit vector in mode-n
$\mathbf{U}^{(n)}$	Transformation / projection / factor matrix in mode-n
$\boldsymbol{v}$	Message type count vector
$\mathbf{w}$	Message type count vector with the timestamp
$\boldsymbol{x}, \boldsymbol{x}^{(n)}$	Vectors in $\mathbb{R}^{I_n}$
$\mathcal{X}$	N-way tensor in $\mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$
$\hat{\mathcal{X}}$	Reconstructed N-way tensor in $\mathbb{R}^{I_1 \times I_2 \times \ldots I_N}$
$\mathbf{X}$	Matrix in $\mathbb{R}^{I_1 \times I_2}$
$\mathbf{X}_i^{(n)}$	Mode-n matricization of i^{th} tensor $\mathcal{X}^{(i)}$
α	Confidence level
β	Penalty coefficient for LogDet distance to identity matrix in DCPM or weight of tensor similarity in LMTD
γ	Heat kernel distance component decay parameter in DCPM or weight of tensor dissimilarity in LMTD or regularization for within-class scatter in R-UMLDA
Δ	Observation interval
ϵ_{th}	Threshold value
ζ	Within-class scatter rate in TR1DA
$\kappa(o_i, o_j)$	Kernel function of the objects o_i and o_j
λ	Penalty coefficient for LogDet distance of two sequential metric matrices
λ_i	i^{th} eigenvalue of a matrix or normalization weight
Λ	Diagonal matrix with eigenvalues or normalization weights on the diagonal
μ	Mean vector or weight of reconstruction fidelity in LMTD
ξ	Slack variable
ρ	Heat kernel time component decay rate
Σ	Covariance Matrix
$\chi^2_{\alpha,d}$	Chi-square test with d dimensions and α confidence level

ψ_i	i^{th} eigenvector of a matrix
Ψ	Matrix of concatenated eigenvectors
Ω	State space
$\times_{(n)}$	Mode-n matrix product in $\mathbb{R}^{I_n}$
$\bar{\times}_{(n)}$	Mode-n vector product in $\mathbb{R}^{I_n}$
$\circ$	Outer product
$\otimes$	Kronecker product for matrices and Tensor product for vectors
$\odot$	Khatri-Rao product
$*$	Hadamard product

LIST OF ACRONYMS/ABBREVIATIONS

2D	Two Dimensional
3D	Three Dimensional
3G	Third Generation of Cellular Mobile Communications
5G	Fifth Generation of Cellular Mobile Communications
APK	Android Application Packages
ARIMA	Autoregressive Integrated Moving Average
CANDECOMP	Canonical Decomposition
CP	CANDECOMP + PARAFAC
CPU	Central Processing Unit
DCPM	Distance-based Change Point Method
DNS	Domain Name Server
DoS	Denial of Service Attack
DDoS	Distributed Denial of Service Attack
EFSM	Extended Finite State Machine
EMP	Elementary Multilinear Projection
GTDA	General Tensor Discriminant Analysis
HMM	Hidden Markov Model
HOOI	Higher Order Orthogonal Iteration
HOSVD	Higher Order Singular Value Decomposition
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
IDS	Intrusion Detection System
IP	Internet Protocol
KL	Kullback–Leibler Divergence
k-nn	k-Nearest Neighbor
LDA	Linear Discriminant Analysis
LPP	Locality Preserving Projections
LMCA	Large Margin Component Analysis

LMNN	Large Margin Nearest Neighbor
LMTD-F	Large Margin Tensor Decomposition - Full
LMTD-C	Large Margin Tensor Decomposition - Core
LogDet	Logarithmic Determinant Divergence
MAC	Media Access Control
mAP	mean Average Precision
MDA	Multilinear Discriminant Analysis
MMS	Multimedia Messaging Service
P2P	Peer-to-Peer Communication
PARAFAC	Parallel Factor Analysis
PBX	Private Branch Exchange
PC	Personal Computer
PCA	Principal Component Analysis
PSD	Positive Semi-Definite
R-UMLDA	Regularized Uncorrelated Multilinear Discriminant Analysis
SIP	Session Initiation Protocol
SMS	Short Message Service
SVD	Singular Value Decomposition
T2T	Tensor-to-Tensor
T2V	Tensor-to-Vector
TCP	Transmission Control Protocol
TR1DA	Discriminant Tensor Rank-1 Decomposition
VoIP	Voice over Internet Protocol

1. INTRODUCTION

In the age of communication, the security and monitoring of the communication traffic is a major concern. The high volume and speed of traffic demand a fast computing model to inspect the packages. It should be easy enough to compute, fast and accurate enough to detect the traffic type. The monitoring system should be autonomous and intelligent to make decisions rather than applying some fixed rules or thresholds. Similarly, storage and retrieval of multi-dimensional structures are another challenge at hand. In case of multi-dimensional structures (tensors), if the data is scarce, a good feature extractor can improve the performance of retrieval from an archive of objects. The tensor decomposition technique should extract a set of discriminative features to retrieve the most similar objects.

Distributed Denial of Service (DDoS) attacks are among the most encountered cyber criminal activities in communication networks that can result in considerable financial and prestige losses for the corporations or governmental organizations. Therefore, autonomous detection of a DDoS attack and identification of its sources is essential for taking counter-measures. For the multi-dimensional structures, in which cases we have scarce data, a discriminative tensor decomposition can capture the correlation between multi-dimensions and it can act as a feature extractor and a pre-processor before a second level processing such as classification and clustering. It can also improve the storage, memory and running times requirements.

In order to address these two separate but closely related challenges, we propose two methods as solutions: A novel distance based change-point detection model applied to a cyber-security system for DDoS detection and attacker discrimination, with an application to SIP networks, and a novel tensor decomposition applied to object retrieval for image and video files.

Distributed Denial of Service (DDoS) attacks are one of the major cyber threats on communication networks. DDoS attacks occur very frequently because they are

fairly simple and cheap to initiate while their broad impact on users and service providers can potentially be severe. Such an attack incapacitates the victim server and renders it unable to provide services at all or at desired quality of service levels to its subscribers. With the cost-effective deployment of cloud systems, DDoS attacks might affect the overall availability of the services by targeting more than one server [1]. They can even be a tool for political struggle on a grander scale; a case in point is the set of DDoS attacks to Turkey's domain name servers by hacktivist groups in December 2015 [2]. As a more radical case, they can be exerted over smart power transmission grids, with potentially more catastrophic consequences [3]. Therefore automatic detection of DDoS attacks and identification of malicious users are crucial in protecting the network entities and for non-degraded service continuity.

Telephone service providers follow the trend of changing their circuit-switched networks to packet-switched ones in view of the cost-effectiveness and maturity of the Voice-over-IP (VoIP) technology. The most popular protocol for control signaling between communicating parties in VoIP is currently the Session Initiation Protocol (SIP) [4]. SIP is based on a simple, HTTP-like text-based request-response transaction model. It provides basic signaling functionalities required for registering clients, checking their presence and online availability, exchanging their communication capabilities, and overall managing the sessions. With the deployment of 5G, VoIP is expected to be one of the major instruments for the multimedia communication. The wide deployment of VoIP networks and the key importance of telephone networks have made the security issues of SIP servers extremely important.

VoIP networks are under a variety of cyber threats and the intensity of attacks seems only to be growing [5]. The attacks can be motivated by potential financial benefits, such as pilfering call charges or causing data leakage masqueraded as a stealth threat. Conversely, it may be part of a plan to cause financial losses to the service providers via heavy service disruption [6]. In addition to the session layer attacks, telecommunication networks are also susceptible to a plethora of other threats below the session layer [7]. Since these are discussed in detail elsewhere, in this dissertation, we focus solely on SIP-specific threats.

Tensors are multi-dimensional arrays, also called N-way arrays, which represent complex structures with higher dimensional relationships such as multi-channel signals, chemical compounds, graphs etc. They have their uses in many application areas (E.g. chemometrics, neuroscience, signal processing and computer vision). With the advancement of data storage and acquisition technologies, and the extensive use of IoT and mobile devices, massive amount of data is collected. Tensor decomposition is a promising tool to handle and process this overwhelmingly increasing amount of stored data.

The tensor decomposition plays two crucial roles; first in understanding the intrinsic subspaces of multiway data, in discovering implicit relationships between features, and second, in reducing the massive tensor data volumes. More explicitly, decompositions that exploit correlations inherent in the tensor data and that extract descriptive features are useful for various inference tasks such as detection, classification, and regression. Additionally, tensor decomposition can be instrumental in alleviating the curse of dimensionality inherent in tensor data and in facilitating training of signal processing and machine learning algorithms.

Distance functions have been extensively studied and the importance of the judicious selection of a distance function in machine learning has been emphasized in the literature. Distance learning, i.e., finding an optimal distance function under given constraints, is a well-understood problem for vectorial data. These methods try either to find projection matrices that minimize the Euclidean distance in the projected space [8–10] or to design a parametrized family of Mahalanobis metrics [11–13] satisfying certain constraints. A metric matrix over the vectorized tensors has been introduced in [14]. Two recent surveys in distance metric learning are given in [15] and [16].

In this dissertation, firstly we propose an anomaly detection model based on Mahalanobis distances to detect attacks and attackers in SIP-communication. Secondly, we propose a new method for generative-discriminative tensor decomposition that incorporates distance metric learning. We apply it to the image and video retrieval problems as case studies.

1.1. Related Work

SIP DoS and DDoS attacks typically exploit vulnerabilities in the SIP protocol. Signature-based attacks utilize properties of the SIP grammar, and can be detected by pattern matching between ongoing traffic and the set of signatures. In other words, this type of attack can be determined or even prevented by inspecting the steps that the attacker must follow through. The non-signature based threats, e.g., behavior-based attacks such as DDoS, are harder to detect. SIP threats can be roughly categorized into 4 groups [7]:

- Service Abuse Threats: These attacks include commercial abuse of services to gain some financial benefit such as toll fraud or billing avoidance.
- Eavesdropping, Interception and Modification Threats: These attacks concentrate on illegally intervening to the call with the goal of capturing sensitive information.
- Social Threats: These attacks use protocol shortcomings, misconfigurations or bugs of SIP server implementation and use these weaknesses to misrepresent the identity of malicious parties to the subscribers.
- (Distributed) Denial of Service ((D)DoS): These attacks focus on the SIP server to prevent it from giving service to the subscribers or to cause significant degradation in the quality of network services. An attacker can achieve this by flooding the server with SIP messages and depleting the network and server resources, such as CPU, memory, bandwidth. In the DoS attack, only one machine is involved to mount the attack on the SIP server. If the attacks are simultaneously performed by many machines, possibly coordinated, the attack becomes a DDoS attack. The botnet attack, where the attack is staged by many zombie machines that are controlled by a master node, is a well-known instance of DDoS.

There is a large variety of possible DDoS attacks, such as Domain Name Server (DNS) attack and fuzzing attack [17, 18]. The DNS flooding attack wastes the bandwidth resources by injecting fake addresses, tying up the call during address resolution,

and causing unnecessary messaging traffic between DNS and SIP server. The fuzzing attack, on the other hand, wastes CPU time by forcing it to parse invalid SIP messages. DDoS attacks in SIP networks can be grouped into four classes: SIP message payload tampering, SIP message flow tampering, SIP message flooding, and finally exploiting SIP vulnerabilities, e.g., for toll fraud [19].

Many methods have been proposed to detect and prevent DDoS attacks in VoIP networks. For example, for the SIP message flooding varieties, an extended finite state machines (EFSM) can be designed for SIP transactions in order to monitor transaction anomalies [20]. Selected network traffic variables are tracked and if an undefined transaction occurs or any traffic variable count exceeds a pre-determined threshold, a preventive action is triggered. A full protocol stack intrusion detection and prevention system for VoIP systems is proposed in [21]. This is a table-based system that collects, correlates and tuples data from different protocols on the communication stack, e.g., MAC addresses, IP addresses, subscriber IDs, packet timestamps. The decisions, such as dropping packets, are given by certain rules applied over these tuples.

In [22], the packets are labeled with respect to their transmission control protocol (TCP) flags. An alarm is raised if the packet counts in a time window deviates from the distribution fitted for the normal traffic. In an alternate research, a naive Bayesian classifier has been constructed as a DDoS detector based on network traffic variables. In [23, 24], a Bayesian change point model that detects traffic surges or dips, which possibly correspond to DDoS attacks is proposed. The model is a hierarchical hidden Markov model that links the features extracted from SIP network traffic and server load to latent variables. One set of these variables tracks the hidden dynamics of the system and the others serve as change point indicators. The output of the model is the posterior probability of a change indication, which is calculated at fixed time intervals.

As for SIP message payload tampering variety, an N-gram technique has been considered to detect the fuzzing attacks exploiting malformed SIP messages. In this case, based on a corpus of SIP messages, which contains both valid and malformed messages, 4-grams, i.e., sequential 4-byte blocks in SIP messages, are extracted. The

4-grams which exceed a given frequency threshold are designated as significant features and their occurrence count vectors are used as features to train classifiers [25]. An experimental study of applying 5 different machine learning models to detect DDoS attacks in SIP-deployed networks have been conducted [26]. The authors have implemented a simulation environment in order to train and evaluate the performance of the models. The classifiers are trained with pre-generated training data collected from SIP message headers, which contain both attacks and normal traffic. The models are required to be re-trained whenever the network or service operating conditions are changed. The trained classifiers are evaluated in terms of accuracy and time overhead required to run them online for each message. A recent research proposes using an autoregressive integrated moving average (ARIMA) time series model to classify the normal traffic, DoS and DDoS attacks [27] for IP networks. The number of packets and the number of IP sources are tracked for each time unit and their ratios are stored. The local Lyapunov exponents are calculated for these ratios and these values are compared with a threshold to discriminate malicious from non-malicious traffic type.

A statistical anomaly detection model, to which our method has resemblances, was proposed in [28, 29]. This method detects significant deviations in the 3G mobile network traffic patterns based on a variant of Kullback-Liebler (KL) divergence between two empirical distributions. The collected data samples for each observed feature within time window are fitted into respective univariate histograms. Then, these empirical distributions are compared with reference distributions of the observed features based on the proposed divergence metric. If the distance of any of the inspected feature distributions to that of its corresponding reference exceeds an empirically set threshold, then an alarm is raised to declare a detected anomaly. A human expert gives the final decision about the detected anomaly as to whether it is an attack or not.

The spread of intelligent mobile devices has resulted in a new facet of mobile botnets. The distributed characteristics of the mobile network (capability to change IP addresses frequently) and huge number of easily-hacked zombie devices by malwares make it hard to prevent the DDoS attacks with conventional PC-centric solutions. Besides using the Internet for command propagation, the bot master can coordinate the

zombies in some exceptional ways such as Bluetooth communication or SMS/MMS messaging. Three different command and control architectures (coordination of zombies by the master) to start a mobile botnet DDoS attack are discussed in [30]. A recent study uses machine learning techniques to discriminate applications that are malwares used in mobile botnets. The manifest files of Android Application Packages (APK) are processed to extract features. After some pre-processing steps, the selected features are used in training classifiers to detect the malwares [31].

A detailed survey on historical evolution of Botnets is provided in [32]. A detailed review of network intrusion systems which are capable of detecting DDoS attacks and the specific methods used for detection can be found in [33].

Analysis of time series for classification, prediction, change and outlier detection has been active research topics for decades with particular focus on financial markets [34]. Among the plethora of methods proposed one can mention: i) methods that map the time series into a new feature space, such as spectral entropy, autocorrelation etc. [35]; ii) kernel methods for time-series classification with emphasis on sequence alignment [36–38]; iii) clustering time series with a combined distance function satisfying the triangle similarity, which is the cosine value between two vector, and dynamic time warping distance [39]; iv) approaches fitting the data to a number of possible models, such as a hidden Markov model with dynamic time warping, or an autoregressive moving average model with dynamic time warping, and clustering the data based on model instance with the best fit [40, 41]; v) singular spectral analysis where data is embedded, the embedding matrix decomposed and reconstructed into trend, noise and oscillatory components.

Metrics, which are functions to calculate distances between two entities in a set, can be used to detect anomalies in the network traffic and in [42] two such information metrics have been proposed for DDoS attacks. Similarly, a DDoS detector which uses the Tsallis entropy has been proposed [43]. The Mahalanobis distance, based on inverse covariance matrix, has been previously used in the detection of abnormal callers (outliers) by inspecting their SIP message flows [44]. In this dissertation, however, we

use an adaptively online trained variety of the Mahalanobis distance for a time series. We use the time series of Mahalanobis distances accompanying the input time series to detect DDoS attacks as well as to identify the malicious user from their messaging behavior analysis.

One of the first intrusion detection system (IDS) system architectures that uses behavioral analysis to detect DDoS attacks and the malicious attackers was proposed in [45]. The attacking entities aiming for a distributed DoS attack are characterized by a common messaging pattern. However, this cannot be represented by a rule-based system. We propose a system that consists of three components: A sniffer to capture the packets, a preprocessor to extract informative features from the packets and a classifier to detect the anomalies in the traffic.

Tensor decomposition has been an increasingly active research topic in recent decades and related literature is already quite rich in algorithms and application cases. A few highlights related to discriminative tensor decompositions and their distance functions are as follows. In [46], a measure for tensor sparsity, called Kronecker-basis-representation is used to compute individual sparse representations that minimize the reconstruction error. In this work a loss function is defined that balances the trade-off between sparsity and reconstruction error and a set of rank-1 tensors that minimize this loss function is obtained. Finding a set of mutually uncorrelated rank-1 tensors with unit vectors in each way (tensor-to-vector projections, each called an elementary multilinear projection: EMP) is presented in [47, 48]. Here, the input tensors are represented as P-dimensional feature vectors that store dot product results between the input tensors and the EMPs, where P is the number of EMPs. Multilinear discriminant analysis (MDA) is a version of linear discriminant analysis (LDA) for tensors that aims to find mutual discriminative projection matrices [49]. A method proposed in [50] extracts a mutual set of projection matrices which preserves the neighborhood relationships of the input space in the projected space. In a similar vein, [51] uses the neighborhood relationships to find discriminative projections from the so-called local tensor descriptor representation of the data. In [52], a face verification system has been designed that uses multilinear whitened principal component analysis to enhance

multilinear PCA and tensor exponential discriminant analysis. General tensor discriminant analysis (GTDA) [53], is one of the earliest tensor-to-tensor projection methods. This method uses the weighted difference of between-scatter and within-scatter matrices in each mode to project a tensor into a discriminative feature space. Another tensor subspace analysis that uses heat kernels to embed the sub-manifold structures inherent in the input space into a feature space is presented in [54]. The method of tensor locally linear discriminative analysis, which can be regarded as a hybrid of local LDA and locality preserving projections (LPP) for tensors, which also uses heat kernels to embed similarities into the projection matrices is introduced in [39]. A low-rank tensor approximation to learn a dictionary for a tensor is proposed in [55]. A loss function is defined over the norm of the projection matrices and of the reconstruction error, and a zero-norm constraint is used to enforce sparsity in dictionaries of predetermined modes. A convex tensor decomposition that uses the norm of singular values as the regularization term finds the set of optimal lower rank tensors to approximate the input tensor is elaborated in [56]. Recent detailed reviews of the tensor factorization methods and their applications can be found in [57–59] and [60].

1.2. The Objective and the Contributions of This Dissertation

In this dissertation, we mainly focus on how to *approximate distances* between the high and dimensional structures and how to apply them into some solutions. We use distances either in the objective function or in the constraints and we find an optimal solution for the problem at hand, which can be local or global depending on the formulation. We propose two novel distance learning models and apply them to two real-life problems.

As our first contribution, we introduce a novel real-time online distance based change-point detector which is applied to an intrusion detection and prevention system for communication networks, particularly for networks with Session Initiation Protocol (SIP) traffic. The proposed system both detects the presence of an attack and identifies the attackers. The system focuses on the DDoS attacks that flood and suffocate a server with excessive amount of requests. One clue for the occurrence of a DDoS attack is

a marked change in the messaging traffic patterns in the network. To this effect, we develop a change detection algorithm which monitors the network traffic intensities at the server side. Significant changes in the characteristics of messaging flows are interpreted as the onset or offset of a potential DDoS attack. We assume tacitly that in a DDoS attack, the attackers are always acting in a coordinated manner.

A novel aspect of the proposed change-point detection method is that it relies on the adaptive tracking of Mahalanobis distances between successive state vectors as a way to monitor abnormal changes in messaging traffic. This enables the monitor to adapt itself to the normal traffic regime and/or to the diurnal or seasonal variations while at the same time remaining sensitive to abnormal changes. The second novelty of our system is that the algorithm beside detecting the occurrence of an attack, can also pinpoint the set of attackers. In other words, under certain realistic assumptions, it can discriminate between messaging patterns of the attackers and those of the non-malicious, i.e., normal users. Similarly, the attacker identification model runs in an unsupervised mode and it is independent of underlying attack model except for the assumption of attacker coordination. Performance results of the algorithm are studied under extensive network traffic and attack traffic simulations.

As our second contribution, we propose a new method that combines distance learning with tensor decomposition and we apply it to the object retrieval problem. A novel tensor decomposition, we call it Discriminative Tensor Decomposition with Large Margin (shortly, LMTD), that uses the distances and pairwise relationships between tensor objects to find a discriminative tensor-to-tensor mapping, is introduced. The pairwise relationships are enforced as distance constraints and the resulting tensor mapping scheme achieves a higher classification performance. We actually present two different versions of the large-margin tensor decomposition algorithm under on neighborhood constraints, one working over the reconstructed tensors, and the other the core tensors. The proposed method straddles generative and discriminative methods in that reconstruction error is also taken into the account. In other words, while the projection matrices are designed to be discriminative, the reconstructed tensor is also forced to be as similar as possible to the input tensor. In this sense our method enables

a trade-off between classification accuracy and reconstruction error. Thus, for example, if discrimination is of prime importance, the corresponding penalty function weight is set to a higher value; if denoising and/or good fidelity shape reconstruction of the detected objects then the weight of the reconstruction error term is boosted. LMTD can also be viewed as a feature extractor. We show that problems at hand, such as data retrieval within a scarce archive, can be solved better over the core tensors instead of the raw input tensors, i.e., with higher precision and better classification accuracy.

This thesis is organized as follows. In Chapter 2, we provide our mathetical notation and background used throughout the thesis in details. Chapter 3 introduces the novel adaptive distance based change-point model, which basically tracks the changes in Mahalanobis distances between the sampled feature vectors and raises an alarm in case of a detected change. Chapter 4 presents the discriminative tensor decomposition with large margin that searches for a set of global projection matrices embedding the similarity (neighborhood) constraints between the multi-dimensional structures into the core tensors. A collection of experimental applications and their results are given in Chapter 5. Finally, in Chapter 6 we conclude this thesis and share the learnings for future studies.

2. MATHEMATICAL BACKGROUND AND DISTANCE FUNCTIONS

This chapter introduces the mutual notation and mathematical background used throughout the dissertation.

2.1. Mathematical Notation

A tensor is a multi-dimensional array and the order (or the ways) of a tensor is its number of dimensions. In this respect, vectors and matrices are 1-way and 2-ways tensors. The basic component of a tensor is called a fiber, which is obtained by fixing all but one of its indices. An unfolding of a tensor at mode-n is represented by a matrix whose columns are constituted by its fibers at the nth mode.

Let $\boldsymbol{x} \in \mathbb{R}^{I_n}$ represent an I_n-dimensional vector, $\mathbf{X}$ a matrix in $\mathbb{R}^{I_n \times I_{n'}}$, and $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$ an N-ways tensor, respectively. The unfolding of a tensor $\mathcal{X}$ in mode-n is represented as the matrix $\mathbf{X^{(n)}} \in \mathbb{R}^{I_n \times (\prod_{k \neq n}^{N} I_k)}$

The inner (dot) product of two conformable tensors is analogous to the inner product of matrices:

$$\langle \mathcal{X}, \mathcal{Y} \rangle = \sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_n=1}^{I_N} x_{i_1 i_2 \ldots i_n} y_{i_1 i_2 \ldots i_n} \tag{2.1}$$

where $x_{i_1 i_2 \ldots i_n}$ represents elements of the N-way tensor. The Frobenius norm can be used in calculating the norm of a tensor as follows:

$$||\mathcal{X}||_F = \sqrt{\sum_{i_1=1}^{I_1} \sum_{i_2=1}^{I_2} \cdots \sum_{i_n=1}^{I_N} x^2_{i_1 i_2 \ldots i_n}} \tag{2.2}$$

An N-way tensor is called a rank-1 tensor if it can be represented as an outer product of N vectors, as follows:

$$\begin{aligned} \mathcal{X} &= \boldsymbol{x}^{(1)} \circ \boldsymbol{x}^{(2)} \circ \ldots \boldsymbol{x}^{(N)} \\ x_{i_1 i_2 \ldots i_n} &= x_{i_1}^{(1)} x_{i_2}^{(2)} \ldots x_{i_n}^{(N)} \end{aligned} \tag{2.3}$$

where $\boldsymbol{x}^{(n)}$ is a vector in $\mathbb{R}^{I_n}$ dimensions and $x_{i_1, i_2 \ldots i_n}$ denotes one of the elements of the tensor $\mathcal{X}$. Note that an elementary multilinear projection (EMP), which reduces each way of the tensor to a scalar is a rank-1 tensor where unit vectors, $||\boldsymbol{x}^{(n)}|| = 1$, are used.

A tensor can be multiplied with a matrix at mode-n as follows:

$$\mathcal{Y} = \mathcal{X} \times_n \mathbf{U}^{(n)} \iff \mathbf{Y}^{(n)} = \mathbf{U}^{(n)} \mathbf{X}^{(n)} \tag{2.4}$$

The Kronecker product of two matrices, $\mathbf{X} \in \mathbb{R}^{I \times J}$ and $\mathbf{Y} \in \mathbb{R}^{K \times L}$, denoted as $\mathbf{X} \otimes \mathbf{Y}$ where $\otimes$ is the Kronecker product operator, is defined as a matrix with size $(IK) \times (JL)$:

$$\begin{aligned} \mathbf{X} \otimes \mathbf{Y} &= \begin{pmatrix} x_{11}\mathbf{Y} & x_{12}\mathbf{Y} & \ldots & x_{1J}\mathbf{Y} \\ x_{21}\mathbf{Y} & x_{22}\mathbf{Y} & \ldots & x_{2J}\mathbf{Y} \\ \vdots & \vdots & \ddots & \vdots \\ x_{J1}\mathbf{Y} & x_{J2}\mathbf{Y} & \ldots & x_{IJ}\mathbf{Y} \end{pmatrix} \\ &= \begin{pmatrix} \boldsymbol{x}_1 \otimes \boldsymbol{y}_1 & \boldsymbol{x}_1 \otimes \boldsymbol{y}_2 & \ldots & \boldsymbol{x}_J \otimes \boldsymbol{y}_{L-1} & \boldsymbol{x}_J \otimes \boldsymbol{y}_L \end{pmatrix} \end{aligned} \tag{2.5}$$

The Khatri-Rao product is defined over matrices that have the same number of columns, $\mathbf{X} \in \mathbb{R}^{I \times K}$ and $\mathbf{Y} \in \mathbb{R}^{J \times K}$. The resulting matrix has size $(IJ) \times K$ and is obtained as:

$$\mathbf{X} \odot \mathbf{Y} = \begin{pmatrix} \boldsymbol{x}_1 \otimes \boldsymbol{y}_1 & \boldsymbol{x}_2 \otimes \boldsymbol{y}_2 & \dots & \boldsymbol{x}_K \otimes \boldsymbol{y}_K \end{pmatrix} \tag{2.6}$$

The elementwise matrix product of two equal-sized matrices is called the Hadamard product (also called Schur product or entrywise product) and defined as:

$$\mathbf{X} * \mathbf{Y} = \begin{pmatrix} x_{11}y_{11} & x_{12}y_{12} & \dots & x_{1J}y_{1J} \\ x_{21}y_{21} & x_{22}y_{22} & \dots & x_{2J}y_{2J} \\ \vdots & \vdots & \ddots & \vdots \\ x_{J1}y_{J1} & x_{J2}y_{J2} & \dots & x_{IJ}y_{IJ} \end{pmatrix} \tag{2.7}$$

2.2. Distance Functions

Design and selection of distance functions, alternatively called metrics, are of paramount importance in inference and signal modeling problems such as k-nn classification, k-means clustering, kernel smoothing etc. The performance of the algorithm depends significantly on how well the metric captures similarities and hidden underlying associations between data instances. The distance function should be conceived so that objects from the same class, i.e., with features from the same region of the feature subspace should be mapped close to each other while objects of different classes in the data set should be as distant as possible.

Given two I_n-dimensional feature vectors $\boldsymbol{x}_i, \boldsymbol{x}_j \in \mathbb{R}^{I_n}$, the squared Euclidean distance is:

$$D_E(\boldsymbol{x}_i, \boldsymbol{x}_j) = \|\boldsymbol{x}_i - \boldsymbol{x}_j\|_2^2 = (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top (\boldsymbol{x}_i - \boldsymbol{x}_j) \tag{2.8}$$

Let's assume we have a transformation $\boldsymbol{v}_i = \mathbf{L}\boldsymbol{x}_i$, where $\mathbf{L}$ is an $I_{n'} \times I_n$ transformation matrix, then we have:

$$\begin{aligned} D_E(\boldsymbol{v}_i, \boldsymbol{v}_j) &= \|\boldsymbol{v}_i - \boldsymbol{v}_j\|_2^2 \\ &= (\mathbf{L}\boldsymbol{x}_i - \mathbf{L}\boldsymbol{x}_j)^\top (\mathbf{L}\boldsymbol{x}_i - \mathbf{L}\boldsymbol{x}_j) \\ &= (\mathbf{L}(\boldsymbol{x}_i - \boldsymbol{x}_j))^\top \mathbf{L}(\boldsymbol{x}_i - \boldsymbol{x}_j) \\ &= (\boldsymbol{x}_i - \boldsymbol{x}_j)^\top \mathbf{L}^\top \mathbf{L}(\boldsymbol{x}_i - \boldsymbol{x}_j) \\ &= D_{\mathbf{L}}(\boldsymbol{x}_i, \boldsymbol{x}_j) \end{aligned} \tag{2.9}$$

Equation 2.9 shows that if the transformation matrix, $\mathbf{L}$, is given, then the squared Euclidean distance between the feature vectors, $(\boldsymbol{v}_i, \boldsymbol{v}_j)$, can be calculated in terms of input vectors, $(\boldsymbol{x}_i, \boldsymbol{x}_j)$.

A useful property is that the squared Frobenius norm of the difference of any two N-way tensors can be computed as the squared Frobenius norm of the difference of the matricized forms of the tensors, or as the squared Euclidean norm of the difference of their vectorized forms, i.e.:

$$\begin{aligned} ||\mathcal{X}_i - \mathcal{X}_j||_F^2 &= ||\mathbf{X}_i^{(n)} - \mathbf{X}_j^{(n)}||_F^2 \\ &= \mathrm{tr}\Big((\mathbf{X}_i^{(n)} - \mathbf{X}_j^{(n)})^\top (\mathbf{X}_i^{(n)} - \mathbf{X}_j^{(n)})\Big) \\ &= ||vec(\mathcal{X}_i) - vec(\mathcal{X}_j)||_2^2 \end{aligned} \tag{2.10}$$

where n can be any mode and $\mathrm{tr}(\bullet)$ is the trace operator.

2.3. Large Margin Distance Models

Our proposed methods use concepts from the large margin algorithms [61], [9], hence a brief outline of these methods is in order. The large margin nearest neighbor algorithm (LMNN) [61] was proposed to learn the optimal distance metric from data in order to enhance the classification performance of the k-nn method. It defines a

semi-definite programming problem over the squared Mahalanobis distances of in-class and impostor sets. Here an impostor is defined as a neighborhood case with "other" class label, and a in-class is a neighborhood case with the correct label, that is, the same class label as that of the data instance. The algorithm minimizes distances of the in-class neighbors while distances to impostors are penalized if they are within a margin. These neighbors must be kept at a safe distance away from the in-class neighbors. This is formulated as a convex programming problem, hence it possesses a unique solution.

Consider the k neighbors of each data instance i in the training set which may be in-class tensors or impostors. Also consider a positive semi-definite matrix, $\mathbf{M}$, where $\mathbf{M} = \mathbf{L}^\top\mathbf{L}$ that induces Mahalanobis distance metric. Henceforth, with some abuse of terminology, we call M the Mahalanobis matrix. This weighting matrix is learned in solving the optimization problem,

$$\begin{aligned}
\min_{\mathbf{M}} \quad & (1-\mu)\sum_{i,j\rightsquigarrow i}(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_j)+\mu\sum_{i,j\rightsquigarrow i,l}(1-y_{il})\xi_{ijl} \\
\text{s.t.} \quad & (\boldsymbol{x}_i-\boldsymbol{x}_l)^\top\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_l)-(\boldsymbol{x}_i-\boldsymbol{x}_j)^\top\mathbf{M}(\boldsymbol{x}_i-\boldsymbol{x}_j)\geq 1-\xi_{ijl} \\
& \xi_{ijl}\geq 0 \\
& \mathbf{M}\succeq 0
\end{aligned} \tag{2.11}$$

where $j \rightsquigarrow i$ (j leads to i) means the j^{th} object, o_j, is one of the in-class k-nearest neighbors of the i^{th} object, o_i in the training set, that is ($o_j \in Ne_k(o_i)$). The slack variable, ξ_{ijl}, is the penalty we pay when an impostor is closer to the instance than an in-class case, and $y_{il} = 1$ if $y_i = y_l$, which are the labels of o_i and o_l, and $y_{il} = 0$ otherwise.

Large Margin Component Analysis (LMCA) [9] is a variant of LMNN and finds a lower dimensional rectangular projection matrix, $\mathbf{L}$, instead of a square Mahalanobis matrix, $\mathbf{M}$. Both methods share the same objective function but since LMCA defines the squared distance in terms of the projection matrix, this is no longer a convex optimization problem and LMCA can only converge to a local optimum.

3. ADAPTIVE DISTANCE BASED CHANGE-POINT DETECTION

We first introduce the notation specific to the communication control, e.g., SIP, messaging. Time is discrete, represented by the instants $t = i\Delta$ at which user behavior data is collected and then processed to output a feature vector. Δ is an observation interval, e.g., 1 second long, within which user messaging activities are monitored. A messaging activity observed at the server side is the arrival of one of the SIP messages (invite, bye, 200 etc.) from a user or the transmission of such a message to a user.

At the end of this interval, the r^{th} user's activity is denoted by the d-dimensional vector $\mathbf{v}_r$, where d is the number of different SIP request or response message types taken into consideration. The vector $\mathbf{v}$ is an integer vector whose components correspond to the number of times each one of the d message types has occurred within the i^{th} time frame ($(i-1)\Delta < t < i\Delta$). Not all users are active in each observation interval. An active user, for example the r^{th} one, is a registered user that has sent and/or received at least one SIP message within the given observation interval, and it is indicated by $u_r, r = 1, \ldots, |U|$, where $|U|$ is the cardinal of this set.

Next, let's look into the details of the user's count vectors. A count vector results from the sum of individual messaging activities of an active user. The r^{th} active user is assumed to run $P_r > 0$ messaging activities within the observation interval. Each messaging activity is represented $\mathbf{v}_r^p, p = 1, \ldots, P_r$, which is a unit vector with one component being 1, and the rest 0. Let's call this as a message indicator vector, because it indicates which one of the d-messages has occurred. Then $\mathbf{v}_r = \sum_{p=1}^{P_r} \mathbf{v}_r^p$, $\mathbf{v}_r$ is simply the count vector of messages sent by the r^{th} user, as shown in Figure 3.1.

Finally, let us introduce the d-dimensional count vector, x, called the state vector, that represents the collective activities of all $|U|$ active users within a time frame. The state vector, which is the total message count vector from all users at the server side

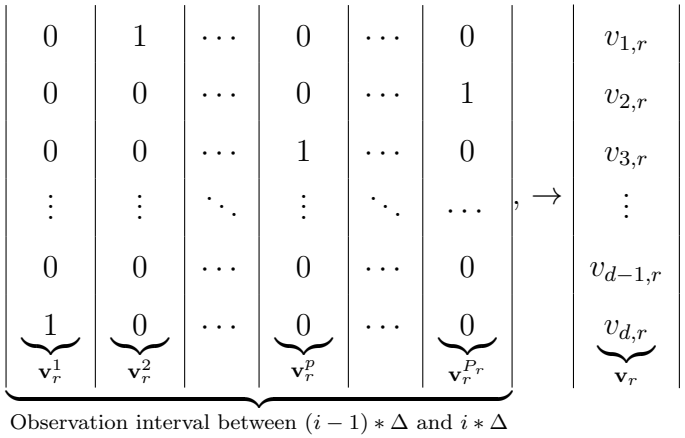


Figure 3.1. The r^{th} user count vector resulting from the accumulation of message indicator vectors ($\mathbf{v}_r = \sum_{p=1}^{P_r} \mathbf{v}_r^p$) in an observation interval.

is simply the sum of the active user count vectors, $\mathbf{x} = \sum_{r=1}^{|U|} \mathbf{v}_r$, and this is illustrated in Figure 3.2.

We have so far omitted any specific index to denote the time frames to avoid notational clutter. However, we will use the notation $\mathbf{x}_i, \mathbf{x}_j \in \mathbb{R}^d$ to denote server state vectors at the i^{th} and j^{th} observation intervals. These feature vectors or server state vectors can be used to monitor the traffic regime changes in a network.

Let $\mathbf{M}$ be a $d \times d$ positive (semi) definite matrix ($\mathbf{M} \in \mathbf{S}_+$ or $\mathbf{M} \in \mathbf{S}_{++}$). $D_{\mathbf{M}}(\mathbf{x}_i, \mathbf{x}_j)$ is the distance between the feature vectors $\mathbf{x}_i$ and $\mathbf{x}_j$ calculated over metric matrix $\mathbf{M}$. $f(\mathbf{M}|\mathbf{x}_n : \mathbf{x}_{n-k-1})$ is a function of $\mathbf{M}$ defined over the time window of length k tracked between feature vectors from $\mathbf{x}_{n-k-1}$ to $\mathbf{x}_n$: From the time index $n-k-1$ to time index n. $D_{ld}(\mathbf{A}, \mathbf{B})$ is a function defined over any two same dimension matrices, $\mathbf{A}$ and $\mathbf{B}$.

Notice that up to this point we have neglected the stamp information, that is, the actual time instances $t_r^1, ..., t_r^{P_r}$ within a generic Δ-long time frame, at which the P_r messaging activities, say, of the r^{th} user, are occuring. We can incorporate this information by augmenting the dimensionality of the message indicator vector, $\mathbf{v}_r^p$, by one, as follows: $(\mathbf{w}_r^p)^\top = ((\mathbf{v}_r^p)^\top, t_r^p)$. Thus, $\mathbf{w}_r^p$ is the timestamp-enriched version of

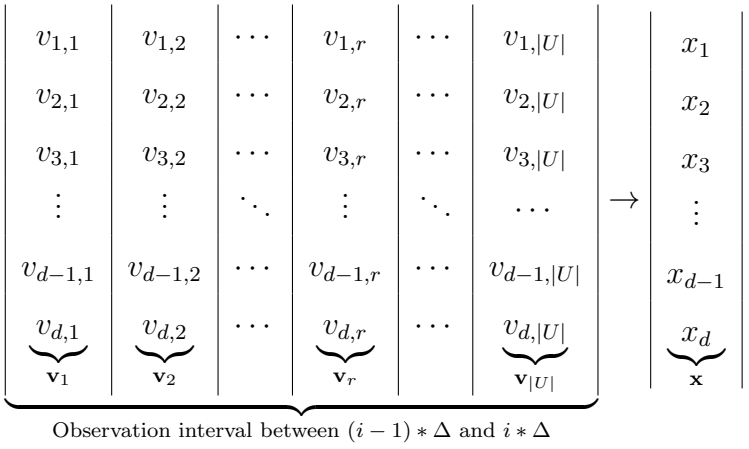


Figure 3.2. The server state vector is the sum of user count vectors ($\mathbf{x} = \sum_{r=1}^{|U|} \mathbf{v}_r$).

$$\mathbf{w}_r^p = \begin{vmatrix} \mathbf{v}_r^p \\ t_r^p \end{vmatrix}$$

Figure 3.3. The timestamped user message vector is the concatenation of user unit message vector and the time it is sent ($\mathbf{w}_r^p \in \mathbb{R}^{d+1}$).

the message indicator vector $\mathbf{v}_r^p$. Notice that $\mathbf{w}_r^p \in \mathbb{R}^{d+1}$ consists of the concatenation of message indicator vector $\mathbf{v}_r^p$ and the time instance at which the message occurs, t_r^p, as given in Figure 3.3.

Given the definitions above, any user u_r, can be mapped to a time series, which can be represented as one of these two matrices: $\mathbf{V}_r = [\mathbf{v}_r^1|\mathbf{v}_r^2|\ldots|\mathbf{v}_r^{P_r}]$ or $\mathbf{W}_r = [\mathbf{w}_r^1|\mathbf{w}_r^2|\ldots|\mathbf{w}_r^{P_r}]$. The kernel function that measures the similarity of any two users pair (u_q, u_r), is represented as $K(u_q, u_r)$. $\kappa(\mathbf{w}_r^{p_r}, \mathbf{w}_q^{p_q})$ is defined as the heat kernel to calculate the similarity between timestamped message vectors of any two users in the same interval: p_r^{th} message of r^{th} user and p_q^{th} message of q^{th} user. Using the user pair kernel functions, for that time interval, we can calculate the kernel matrix $\mathbf{K}$ of size $|U| \times |U|$.

3.1. Adaptive Distance Based Change-Point Detection Estimator

Feature instances extracted from adjacent intervals within the correlation length of a stationary process tend to have high statistical similarity. On the other hand, features originating from different generative processes or from different sections of a non-stationary process can be expected to have large pairwise distances. Based on this premise, a significant change in the distances between consecutive feature vectors in a time series can be interpreted as an indicator of a change in the data generating process. The Hidden Markov Model (HMM) can capture these regime changes as a switching variable from one generator to another in the hidden layer. In the context of communication networks, such an abrupt change in feature vectors corresponding to traffic intensity patterns and/or of server resource utilization rates can be conjectured to signal a DDoS attack. A Distance based Change-Point Method (DCPM), as used in our work, first tracks the distances between sequential feature vectors and then computes the statistics of these distances to decide for a change or not. Judicious choice of a distance function can prove critical in the performance of machine learning algorithms. To this effect, one can use one of the well-known distance functions or attempt to learn a distance function specific to the problem at hand. In this work, we have opted to use a learning scheme for the Mahalanobis distance.

3.1.1. Distance Based Change-Point Model

The distance-based change detection is achieved by inspecting sum of distances over a sliding window, called moving distance, where distances between the current feature vector and its immediate predecessors in a time-frame of size k are summed. The result of the sliding window sum is compared with a threshold value, ϵ_{th}, and an alarm is raised for the potential occurrence of a regime change. This step is followed by the malicious user discrimination algorithm, as detailed in Section 3.2. The main novelty of this method is that we learn the weight matrix $\mathbf{M}$ (called the Mahalanobis metric from now on) under a loss function so that the detection algorithm is adapted to inlier variations and trends in the traffic intensity to avoid false alarms. The inlier variations can be due to diurnal or week-day based changes or to short-lived sporadic

flurry of call activities.

The moving distance over a k-sized time frame can be defined as a function of the symmetric positive definite matrix, $\mathbf{M} \in \mathbf{S}_{++}$, as follows:

$$f(\mathbf{M}|\mathbf{x}_n : \mathbf{x}_{n-k-1}) = \sum_{j=n-k-1}^{n-1} (\mathbf{x}_n - \mathbf{x}_j)^\top \mathbf{M} (\mathbf{x}_n - \mathbf{x}_j) \tag{3.1}$$

If the moving distance computed using the current Mahalanobis metric is above the threshold, $f(\mathbf{M}_{n-1}|\mathbf{x}_n : \mathbf{x}_{n-k-1}) > \epsilon_{th}$, then an alarm is raised. The Mahalanobis metric is updated periodically at each time interval under the loss function given below:

$$\min_{\mathbf{M} \in \mathbf{S}_{++}} \quad f(\mathbf{M}|\mathbf{x}_n : \mathbf{x}_{n-k-1}) + \lambda D_{ld}(\mathbf{M}, \mathbf{M}_{n-1}) + \beta D_{ld}(\mathbf{M}, \mathbf{I}) \tag{3.2}$$

In Equation 3.2, the second and the third terms, $\lambda D_{ld}(\mathbf{M}, \mathbf{M}_{n-1})$ and $\beta D_{ld}(\mathbf{M}, \mathbf{I})$, respectively, are regularization functions based on the logarithmic determinant divergence (LogDet) [12]. LogDet function is a pseudo-metric that measures the distance between two matrices and is defined in Equation 3.3. Detailed information about the LogDet function can be found in the Appendix section. The former regularizer imposes the updated matrix to be as similar as possible to its predecessor. The latter one forces it to be as close as possible to the identity matrix to prevent it from converging to an irrelevant matrix and at the same time to induce sparsity. Thus, their relative weights can be gauged to trade-off the update rate of the Mahalanobis metric and the aging of the effect of the past measurements. The four parameters to be set are the sliding window size k (time frame size), the two regularization cost weights, λ and β, and the parameter α for thresholding. At the start of the algorithm, $\mathbf{M}_0$ is initialized as the identity matrix, $\mathbf{M}_0 = \mathbf{I}$. Since the LogDet is a convex function of $\mathbf{M}$, we are guaranteed to find the optimal positive definite matrix, that minimizes the criterion in Equation 3.3.

Require k, λ, β and α (for ϵ_{th}).
Initialize $\mathbf{M}_0$ (default $\mathbf{I}$).
repeat
Inspect the SIP traffic in the time window of size k, and compute the count vector.
if $f(\mathbf{M}_{n-1}|\mathbf{x}_n : \mathbf{x}_{n-k-1}) > \epsilon_{th}$ **then**
Raise alarm.
Run the malicious user detection algorithm given in Figure 3.8.
end if
Evaluate $\mathbf{M}^*$.
Set $\mathbf{M}_{n-1} = \mathbf{M}^*$.
until the traffic ends

Figure 3.4. Adaptive Online Distance Based Change-Point Detection Algorithm.

$$D_{ld}(\mathbf{M}, \mathbf{M}_{t-1}) = \mathrm{tr}(\mathbf{M}\mathbf{M}_{t-1}^{-1}) - \log\det(\mathbf{M}\mathbf{M}_{t-1}^{-1}) - d \tag{3.3}$$

where tr(•) is the trace function for the matrices.

The optimal Mahalanobis metric ($\mathbf{M}^*$) can be found by taking the derivative of Equation 3.2 and setting it to zero.

$$\mathbf{M}^* = \left(\frac{\lambda}{\lambda+\beta}\mathbf{M}_{n-1}^{-1} + \frac{\beta}{\lambda+\beta}\mathbf{I} + \frac{1}{\lambda+\beta}\sum_{j=n-k-1}^{n-1}(\mathbf{x}_n - \mathbf{x}_j)(\mathbf{x}_n - \mathbf{x}_j)^\top\right)^{-1} \tag{3.4}$$

This Mahalanobis metric update is repeated at each time index. The change detection algorithm is given in Figure 3.4.

3.1.2. Thresholding of the Moving Distances

The characteristics of the moving average of distances depend on the traffic volume intensity, the dimension of the feature vector, the size of the time frame etc., and hence it becomes critical to set a threshold value judiciously to detect regime anomalies or abrupt changes. In this dissertation we test comparatively two different threshold functions.

Experimental evidence has shown that we can approximate the distribution of the moving sum of distances as a Chi-squared distribution. It is then assumed that Mahalanobis distances are obtained from a Gaussian distribution such that $\mu = \mathbf{x}_n$ in the immediate past observation interval, and $\mathbf{\Sigma} = \mathbf{M}^{-1}$. If $\mathbf{y}$, which is the set of observations in the current sliding window, is a d-dimensional random vector drawn from a Gaussian distribution with a mean vector μ and a d-rank covariance matrix $\mathbf{\Sigma}$, then $z = (\mathbf{y} - \mathbf{x}_n)^\top \mathbf{M}(\mathbf{y} - \mathbf{x}_n) = (\mathbf{y} - \mu)^\top \mathbf{\Sigma}^{-1}(\mathbf{y} - \mu)$ becomes Chi-Squared distributed with d-degrees of freedom.

Let z_i denote one of k independent, identically distributed random variables that follow a chi-square distribution such as $z_1 \sim \chi^2_{\alpha,d_1}$, $z_2 \sim \chi^2_{\alpha,d_2}$, ..., $z_k \sim \chi^2_{\alpha,d_k}$. Due to the additive property of independent chi-squared variables, the sum of the random variables follows a chi-square distribution with $d_1 + d_2 + \ldots + d_k$ degrees of freedom. That is,

$$\begin{aligned} Z &= z_1 + z_2 + \ldots + z_k \\ Z &\sim \chi^2_{\alpha,d_1+d_2+\ldots+d_k} \end{aligned} \tag{3.5}$$

Thus, the threshold of our anomaly detection model becomes $\epsilon_{th} = \chi^2_{\alpha,k*d}$. The α parameter is the probability of accepting a chance fluctuation as an anomaly. In other words, in the absence of an attack the score of the moving average of distances, denoted by Z above, has a probability less than α to exceed the threshold ϵ_{th}. The converse

event of Z exceeding the threshold can be accepted as an anomaly with probability $1 - \alpha$. The value of α depends on the requirements of the system and it is typically set by an human expert to some such value as $\alpha \in \{0.1, 0.05, 0.02, 0.01\}$. This is a statistical approach that is based on the sum of observed distances.

An alternate, empirically found constant threshold, which is a function of two system parameters is given below:

$$\epsilon_{th} = c\ k \left(\frac{d}{2}\right)^2 \tag{3.6}$$

and is found to work equally well. This fixed threshold value only depends on the time frame size (k), the number of dimensions (d) and a constant c. As a plausible argument for the fact that the constant thresholding function works equally well, we observe that the same parameters (k and d) are also inherent in the χ^2 thresholding. Notice also that there is some liberty in adjusting this threshold by setting the constant c according to the requirements of the deployed system. A case in point could be to make the constant indexed by time periods c_n, e.g., to account for seasonal trends.

More importantly, even though the threshold is set to a constant, the system is still an adaptive model due to the adaptation inherent in the updates of the Mahalanobis metric. At each observation interval, the Mahalanobis metric is updated to accommodate the new distances between the observations. Therefore whenever the threshold is exceeded, it means that the quadratic smoother could not smooth out the new measurement digressions, and therefore it is very likely to be an anomaly.

3.2. Malicious User Discrimination

If a detected anomaly is in fact a DDoS attack, the next task is to identify the set of malicious users that are presumably coordinating to mount a distributed attack. For this analysis, each subscriber's behavior history in the observation interval is represented as a time-series, as given in Figure 3.1. We process the time series using a similarity functions so that the subscribers with similar behavior patterns are clustered

into the same group. We have proposed and evaluated two different attacker discrimination methods. The first one is based on a global time series alignment kernel that makes use of both epoch differences and feature distances between message sequences. The second one uses the user message count vectors at the end of periodic observation intervals, i.e., the information on message time instants are ignored. The pairwise similarity of any two users is calculated using their count vectors.

3.2.1. Sequence Alignment Kernel

We consider the ensemble of the timestamped messages sent by a user within a time frame of k units, say $(n-k-1), \ldots, (n-1)$, as message sequences. Each user's sequence can have a different number of messaging events, each event occuring at a different time instant. In other words, a user's message sequence or time series corresponds to the ensemble of messages sent by a registered terminal within the designated observation interval, each event being characterized by the type of SIP message and its timestamp. Our goal is to estimate the similarity of messaging activities of the users via a kernel-based scheme. For this purpose, the message sequences must be aligned without pair repetition. The similarity between two sequences of possibly different lengths, i.e, number of messaging events, can be determined as the sum of similarities of all their feasible alignments. Thus two sequences are more similar as a pair if their messaging types, e.g., invite or bye, and their occurrences in time resemble each other.

Let us assume the user time series, i.e., timestamped message sequences, $(\mathbf{W}_q, \mathbf{W}_r)$ of the user pair (u_q, u_r), $\mathbf{W}_q = [\mathbf{w}_q^1 | \mathbf{w}_q^2 | \mathbf{w}_q^3]$ and $\mathbf{W}_r = [\mathbf{w}_r^1 | \mathbf{w}_r^2]$ with three and two messaging events, respectively. Figure 3.5 shows an example of all possible alignments for these two sequences. In this specific example, there are 5 possible alignments, as follows:

- $(\mathbf{w}_q^1, \mathbf{w}_r^1), (\mathbf{w}_q^1, \mathbf{w}_r^2), (\mathbf{w}_q^2, \mathbf{w}_r^2), (\mathbf{w}_q^3, \mathbf{w}_r^2)$
- $(\mathbf{w}_q^1, \mathbf{w}_r^1), (\mathbf{w}_q^2, \mathbf{w}_r^2), (\mathbf{w}_q^3, \mathbf{w}_r^2)$
- $(\mathbf{w}_q^1, \mathbf{w}_r^1), (\mathbf{w}_q^2, \mathbf{w}_r^1), (\mathbf{w}_q^2, \mathbf{w}_r^2), (\mathbf{w}_q^3, \mathbf{w}_r^2)$
- $(\mathbf{w}_q^1, \mathbf{w}_r^1), (\mathbf{w}_q^2, \mathbf{w}_r^1), (\mathbf{w}_q^3, \mathbf{w}_r^2)$

- $(\mathbf{w}_q^1, \mathbf{w}_r^1), (\mathbf{w}_q^2, \mathbf{w}_r^1), (\mathbf{w}_q^3, \mathbf{w}_r^1), (\mathbf{w}_q^3, \mathbf{w}_r^2)$

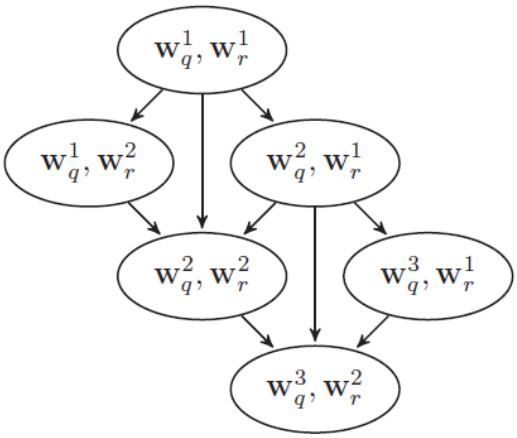


Figure 3.5. All possible alignments for $\mathbf{W}_q = [\mathbf{w}_q^1|\mathbf{w}_q^2|\mathbf{w}_q^3]$ and $\mathbf{W}_r = [\mathbf{w}_r^1|\mathbf{w}_r^2]$.

A global alignment kernel has been proposed in [62], which uses dynamic programming to compute the similarity of all possible alignments of two sequences. We use a variation of this algorithm, where we employ a pairwise heat kernel that is based on the Mahalanobis distance and differences of time stamps.

Global Sequence Alignment Kernel: Given the two message sequences such that $\mathbf{W}_q = [\mathbf{w}_q^1|\mathbf{w}_q^2|\dots|\mathbf{w}_q^{P_q}]$ and $\mathbf{W}_r = [\mathbf{w}_r^1|\mathbf{w}_r^2|\dots|\mathbf{w}_r^{P_r}]$ for the user pair (u_q, u_r) in a state space Ω, we set the doubly-indexed series T_{p_q,p_r} as $T_{p_q,0} = 0$ for $p_q = 1, ..., P_q$, $T_{0,p_r} = 0$ for $p_r = 1, ..., P_r$, and $T_{0,0} = 1$. We also assume that there is a function to measure the similarity between the p_q^{th} signaling event of user u_q and the p_r^{th} signaling event of other user u_r, $\kappa(\mathbf{w}_q^{p_q}, \mathbf{w}_r^{p_r})$. Computing recursively $(p_q, p_r) \in \{1, ..., P_q\} \times \{1, ..., P_r\}$, for the terms, one has:

$$T_{p_q,p_r} = (T_{p_q,p_r-1} + T_{p_q-1,p_r-1} + T_{p_q-1,p_r})\kappa(\mathbf{w}_q^{p_q}, \mathbf{w}_r^{p_r}) \tag{3.7}$$

Finally, the unnormalized similarity between two users (u_q, u_r) is measured when the recursion has considered all possible alignments, that is:

$$K_{unnormed}(u_q, u_r) = T_{P_q, P_r} \tag{3.8}$$

After that the kernel matrix for all user pairs has been obtained, we unit-diagonal normalize the $|U| \times |U|$ kernel matrix, where $|U|$ is the number of active users in the system, in order to eliminate any scaling issues:

$$\begin{aligned} K(u_q, u_r) &= \frac{K_{unnormed}(u_q, u_r)}{\sqrt{K_{unnormed}(u_q, u_q)}\sqrt{K_{unnormed}(u_r, u_r)}}, q, r = 1, ..., |U| \\ K(u_q, u_r) &\rightarrow [0, 1] \end{aligned} \tag{3.9}$$

We will call this kernel as the time series kernel.

Pairwise Heat Kernel: Each user in a time window can be represented in terms of her ordered timestamped message sequence. Recall that user sequences can have differing lengths and can consist of different types of messages.

A kernel function (pairwise heat function) for any two timestamped vectors, $(\mathbf{w}_q^{p_q})^\top = ((\mathbf{v}_q^{p_q})^\top, t_q^{p_q})$ and $(\mathbf{w}_r^{p_r})^\top = ((\mathbf{v}_r^{p_r})^\top, t_r^{p_r})$ is evaluated as:

$$\begin{aligned} \kappa(\mathbf{w}_q^{p_q}, \mathbf{w}_r^{p_r}) &= \exp(-\gamma D_{\mathbf{M}}(\mathbf{v}_q^{p_q}, \mathbf{v}_r^{p_r}) - \rho|t_q^{p_q} - t_r^{p_r}|) \\ D_{\mathbf{M}}(\mathbf{v}_q^{p_q}, \mathbf{v}_r^{p_r}) &= (\mathbf{v}_q^{p_q} - \mathbf{v}_r^{p_r})^\top \mathbf{M} (\mathbf{v}_q^{p_q} - \mathbf{v}_r^{p_r}) \end{aligned} \tag{3.10}$$

where $\mathbf{M}$ is the Mahalanobis metric evaluated at that observation interval as in Equation 3.4. Note that $\kappa(\mathbf{w}_q^{p_q}, \mathbf{w}_r^{p_r}) = 1$ iff $\mathbf{v}_q^{p_q} = \mathbf{v}_r^{p_r}$ and $t_q^{p_q} = t_r^{p_r}$. The coefficients γ and ρ determine the weights of message type distance and timing distance, respectively. In this study we have assumed $\gamma = \rho = 1$.

3.2.2. User Distance Kernel

A kernel matrix of pairwise user-to-user similarities can be created based on their Mahalanobis distances. User pairs would have high similarity (close to 1) if their Mahalanobis distance is close to 0; conversely, if the pair similarity is small (close to 0), then their distance is large. The Mahalanobis distance kernel can be regarded as a variant of Gaussian kernel.

Any two users, u_q and u_r, can be compared based on their messaging count vectors $\mathbf{v}_q, \mathbf{v}_r \in \mathbb{R}^d$, as follows:

$$K(u_q, u_r) \;=\; \exp(-(\mathbf{v}_q - \mathbf{v}_r)^\top \mathbf{M}(\mathbf{v}_q - \mathbf{v}_r)) \tag{3.11}$$

We will call this kernel simply as distance kernel. $K(u_q, u_r)$ is 1 iff $\mathbf{v}_q = \mathbf{v}_r$ Note that this feature vector does not take into account the occurrence timing of the messages, but it averages the messaging traffic in that interval. We would like to point out again the difference between the two ways of measuring user behavior differences. In Equation 3.11, we consider the messaging events integrated over the observation interval, which is represented by the d-dimensional count vector of messaging events according to their types. In Equations 3.9 and 3.10, we calculate the difference of user behaviors by comparing and measuring distances, messaging event by messaging event, as they occur during the observation interval.

3.2.3. Spectral Clustering

A matrix of pairwise user-to-user similarities is created from the users' messages as in Equations 3.9 or 3.11. The kernel matrix, $\mathbf{K}$, then corresponds to a fully connected weighted adjacency graph, where the users are the vertices and the similarities are the edge costs. The adjacency matrix is expected to consists of two sub-graphs: One representing the malicious users characterized by similar behavior patterns and the other representing the non-malicious users with random-like behavior patterns. In

order to partition this graph into these two sub-graphs, we have used the normalized Laplacian spectral clustering algorithm. Such algorithms are conceived to find graph partitioning solutions in clustering problems. In the literature there are various spectral clustering algorithms. We have preferred to use normalized Laplacian spectral clustering because we want not only to have the similar nodes to be closely projected to each other, but also to have the dissimilar nodes to be projected far from each other. The normalized spectral methods satisfy both of these criteria, as discussed in [63].

The degree of q^{th} active user in the kernel matrix, which is the sum of all the weight entries related the q^{th} active user, at a given time frame is evaluated as:

$$dg_q = \sum_{r=1}^{|U|} \mathbf{K}_{q,r} \tag{3.12}$$

where $\mathbf{K}_{q,r} = \mathbf{K}(u_q, u_r)$.

The degree matrix $\mathbf{D}$ is a diagonal matrix whose diagonal elements contain the degree values, $dg_1, dg_2, \ldots, dg_{|U|}$. The Laplacian matrix, $\mathbf{L}$, is evaluated as in Equation 3.13 and the spectral clustering algorithm is given in Figure 3.6.

$$\mathbf{L} = \mathbf{D} - \mathbf{K} \tag{3.13}$$

where $\mathbf{K}$ is the $|U| \times |U|$ is kernel matrix whose entries, $\mathbf{K}(u_q, u_r)$, are calculated as in Equations 3.9 or 3.11.

3.2.4. Automatic Identification of Malicious Users Cluster

The malicious users are conjectured to be characterized by repetitive and correlated behaviors, while the rest of users are characterized by uncoordinated and diverse behaviors. Once the two clusters are obtained, then the final task would be that of distinguishing the attacker set.

Require $\mathbf{K} \in \mathbb{R}^{|U| \times |U|}$.

Evaluate $\mathbf{D}$ and $\mathbf{L}$.

Compute the first two eigenvectors, ψ_1 and ψ_2, of the two smallest eigenvalues $0 = \lambda_1 < \lambda_2$ for the generalized eigenproblem $\mathbf{L}\psi = \Lambda\mathbf{D}\psi$, where Λ is the diagonal matrix of eigenvalues $\lambda_1, \ldots, \lambda_{|U|}$.

Matricize ψ_1 and ψ_2 vectors to obtain $\mathbf{\Psi} \in \mathbb{R}^{|U| \times 2}$. Use the rows of $\mathbf{\Psi}$ as the new feature vectors in the mapped space, $\mathbf{y} \in \mathbb{R}^2$.

Apply 2-means clustering.

return the cluster label vector $\mathbf{C}$ from 2-means clustering.

Figure 3.6. Normalized Laplacian Spectral Clustering.

For each of the two clusters, we compute the sample covariance matrix of the user message sequence vectors in that cluster. Since the malicious user cluster is assumed to consist of similar messaging behaviors, such message vectors are expected to be more strongly aligned along a few particular axes. In fact, in the extreme case when all messages in the cluster are of the same type, the sample covariance matrix would be the $\mathbf{0}$ matrix. Therefore, we assign the cluster with significantly higher eigenvalue concentration to malicious users. This algorithm, based on the heuristics that malicious users must be somewhat coordinated to mount an attack, and therefore that the data vectors must concentrate along a few eigenvectors as given in Figure 3.7. Each cluster is assumed to contain at least two subscribers.

Putting all of these steps together, the algorithm to detect the attackers is summarized in Figure 3.8.

Require the cluster label vector **C**

Determine the two clusters, C_1 and C_2.

For the two clusters, evaluate the sample covariance matrix of the projected message vectors.

if a cluster has a covariance matrix = **0** **then**

Return this cluster.

else

Evaluate the eigenvalues of the cluster covariance matrices.

Return the cluster with the highest eigenvalue

end if

Figure 3.7. Cluster Selection Heuristics.

if Global Sequence Kernel is used **then**

Set the weight parameters γ and ρ of the pairwise heat kernel.

Evaluate the kernel matrix $\mathbf{K}$ such that $\forall\ (u_q, u_r) \in U \times U$, we have $\mathbf{K}_{q,r} = K(u_q, u_r)$, where we use the timestamped message sequences $\mathbf{W}_q$, $\mathbf{W}_r$ of the q^{th} and r^{th} users in the given time interval, respectively with the alignment kernel, and U is the set of active users.

Unit-diagonal normalize $\mathbf{K}_{unnormed}$ to obtain $\mathbf{K}$.

end if

if User Distance Kernel is used **then**

Evaluate the kernel matrix $\mathbf{K}$ such that $\forall\ (u_q, u_r) \in U \times U$, we have $\mathbf{K}_{q,r} = K(u_q, u_r)$, where we use the total message count vectors $\mathbf{v}_q$, $\mathbf{v}_r$ of the q^{th} and r^{th} users in the given time interval, respectively with the distance kernel, and U is the set of active users.

end if

Apply the normalized Laplacian spectral clustering algorithm over $\mathbf{K}$ such that # clusters = 2, as defined in Figure 3.6.

Use the cluster label vector **C** returned by the spectral clustering in cluster selection heuristics as defined in Figure 3.7.

return The selected cluster members as the set of attackers.

Figure 3.8. Attacker Detection.

4. DISCRIMINATIVE TENSOR DECOMPOSITION WITH LARGE MARGIN

Before introducing the proposed tensor decomposition, we provide the notation followed throughout this chapter and a brief mathematical background on tensor decomposition.

4.1. Tensor Decomposition and Distances Between Tensors

The most popular decomposition method consists of decomposing a tensor in a sum of rank-1 tensors. The CANDECOMP (canonical decomposition) and PARAFAC (parallel factor analyis) are the two methods, pioneered independently, that accomplish the same rank-1 factor decomposition of a tensor [57]. For example, the CP decomposition of a third-order tensor, $\mathcal{X} \in \mathbb{R}^{I \times J \times K}$ can be written as:

$$\mathcal{X} \;=\; \sum_{p=1}^{P} \boldsymbol{a}_p \circ \boldsymbol{b}_p \circ \boldsymbol{c}_p \tag{4.1}$$

where P is some positive integer, $\boldsymbol{a}_p \in \mathbb{R}^I$, $\boldsymbol{b}_p \in \mathbb{R}^J$, and $\boldsymbol{c}_p \in \mathbb{R}^K$ for $p = 1, \ldots, P$. Figure 4.1 illustrates the CP decomposition of a 3-way tensor. For an Nth-order tensor, $\mathcal{X} \in \mathbb{R}^{I_1 \times I_2 \times \ldots \times I_N}$, the decomposition expression becomes:

$$\begin{aligned} \mathcal{X} \;&\approx\; [\![\boldsymbol{\Lambda}; \mathbf{A}^{(1)}, \mathbf{A}^{(2)}, \ldots, \mathbf{A}^{(N)}]\!] \\ &\equiv\; \sum_{p=1}^{P} \lambda_p \boldsymbol{a}_p^{(1)} \circ \boldsymbol{a}_p^{(2)} \circ \ldots \circ \boldsymbol{a}_p^{(N)} \end{aligned} \tag{4.2}$$

where $\boldsymbol{a}_p^{(n)}$ is a unit normalized fiber, $\mathbf{A}^{(n)} \in \mathbb{R}^{I_n \times P}, \mathbf{A}^{(n)} = [\boldsymbol{a}_1^{(n)} | \boldsymbol{a}_2^{(n)} | \ldots | \boldsymbol{a}_P^{(n)}]$ and $\boldsymbol{\Lambda} \in \mathbb{R}^{P \times P}$ is a diagonal matrix that stores normalization weights. The rank of a tensor is defined as the minimum number of rank-1 tensors required to exactly reconstruct it in PARAFAC.

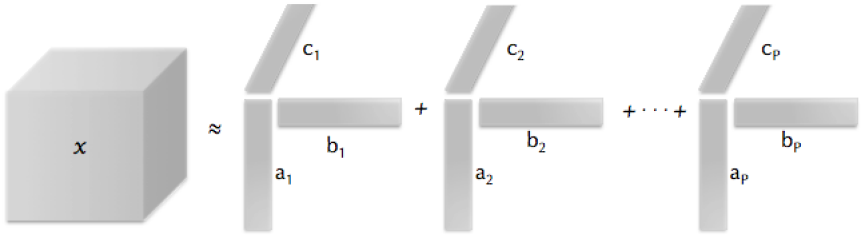


Figure 4.1. CP decomposition of a 3-way array.

The mode-n matricization in terms of the Khatri-Rao products of the remaining modes is given as:

$$\begin{aligned} \mathbf{X}^{(n)} &\approx \mathbf{A}^{(n)}\mathbf{\Lambda}\Big(\mathbf{A}^{(N)} \odot \mathbf{A}^{(N-1)} \odot \ldots \odot \\ &\quad \mathbf{A}^{(n+1)} \odot \mathbf{A}^{(n-1)} \odot \ldots \odot \mathbf{A}^{(1)}\Big)^{\top} \end{aligned} \tag{4.3}$$

An alternate popular tensor decomposition is the Tucker decomposition, which can be regarded as the extension of principal component analysis (PCA) to tensors. Its N-way generalization can be written as:

$$\begin{aligned} \mathcal{X} &= \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_N \mathbf{U}^{(N)} \\ &= [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!] \end{aligned} \tag{4.4}$$

or

$$x_{i_1 i_2 \ldots i_N} = \sum_{p_1=1}^{P_1} \ldots \sum_{p_N=1}^{P_N} g_{p_1 p_2 \ldots p_N} u^{(1)}_{i_1 p_1} \ldots u^{(N)}_{i_N p_N} \tag{4.5}$$

for $i_n = 1, \ldots, I_n, n = 1, \ldots, N$ and where $\mathcal{G}$ denotes the core tensor, $g_{p_1 p_2 \ldots p_N} \in \mathcal{G}$.

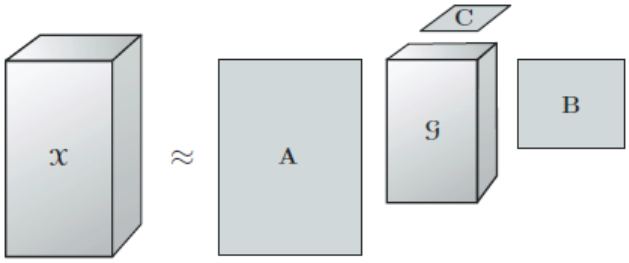


Figure 4.2. Truncated Tucker decomposition of a 3-way array.

The two best known versions of the Tucker decomposition are higher order orthogonal iteration (HOOI) and higher order singular value decomposition (HOSVD), both of which compute projection matrices with orthogonal columns. Since $\mathbf{U}^{(n)}$ matrices are orthonormal, for these two versions of the Tucker decomposition the following also holds:

$$\mathcal{G} = \mathcal{X} \times_1 \mathbf{U}^{(1)\top} \times_2 \mathbf{U}^{(2)\top} \ldots \times_N \mathbf{U}^{(N)\top} \tag{4.6}$$

It is possible to compress a tensor by representing it as a core tensor multiplied by a projection matrix in each mode, where the projection matrices have ranks $(P_1, P_2, \ldots, P_N)$ and $P_n \leq \text{rank}(\mathbf{X}^{(n)})$. Note that in the case of strict inequality in one or more of the modes, that is for a truncated Tucker decomposition, Equation 4.4 does not hold anymore, and we obtain an approximation to the tensor. Figure 4.2 illustrates the truncated Tucker decomposition of a 3-way tensor.

$$\begin{aligned} \hat{\mathcal{X}} &\approx \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_N \mathbf{U}^{(N)} \\ &\approx [\![\mathcal{G}; \mathbf{U}^{(1)}, \mathbf{U}^{(2)}, \ldots, \mathbf{U}^{(N)}]\!] \end{aligned} \tag{4.7}$$

The details of distance functions and metrics are already discussed in Subsection 2.2. Here we refocus on Frobenius norm. The squared Frobenius norm of the difference

of two matrices or tensors (we call it Frobenius distance from now on) is a tractable and popular metric in machine learning algorithms. In the sequel we will use Frobenius distance as a measure of goodness of approximation in tensor decompositions. More specifically, while any tensor can be decomposed in terms of a core tensor and its projection matrices, we will design tensor projection matrices to maximize classification accuracy in the projected space.

We can further manipulate the Frobenius distance in Equation 2.10 using their joint Tucker decompositions, as given in Equation 4.4:

$$\begin{aligned} ||\mathcal{X}_i - \mathcal{X}_j||_F^2 &= ||\mathcal{G}_i \times_1 \mathbf{U}^{(1)} \ldots \times_N \mathbf{U}^{(N)} - \\ &\quad \mathcal{G}_j \times_1 \mathbf{U}^{(1)} \ldots \times_N \mathbf{U}^{(N)}||_F^2 \end{aligned} \tag{4.8}$$

If we define $\mathcal{X}^{(-n)}$ as follows,

$$\begin{aligned} \mathcal{X} &= \mathcal{X}^{(-n)} \times_n \mathbf{U}^{(n)} \\ \mathcal{X}^{(-n)} &= \mathcal{G} \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_{n-1} \mathbf{U}^{(n-1)} \\ &\quad \times_{n+1} \mathbf{U}^{(n+1)} \ldots \times_N \mathbf{U}^{(N)} \end{aligned} \tag{4.9}$$

then, we can rewrite Equation 4.8:

$$\begin{aligned} ||\mathcal{X}_i - \mathcal{X}_j||_F^2 &= ||\mathcal{X}_i^{(-n)} \times_n \mathbf{U}^{(n)} - \mathcal{X}_j^{(-n)} \times_n \mathbf{U}^{(n)}||_F^2 \\ &= ||\mathbf{U}^{(n)}(\mathbf{X}_i^{(-n)} - \mathbf{X}_j^{(-n)})||_F^2 \\ &= \mathrm{tr}\Big((\mathbf{X}_i^{(-n)} - \mathbf{X}_j^{(-n)})^\top \mathbf{U}^{(n)\top} \mathbf{U}^{(n)} \\ &\quad (\mathbf{X}_i^{(-n)} - \mathbf{X}_j^{(-n)})\Big) \end{aligned} \tag{4.10}$$

Equation 4.10 implies that if we continue the Tucker decomposition with orthonormal matrices for all modes, than we will be able to compute the Frobenius distance of two tensors in terms of their core tensors and mutual projections matrices.

4.2. Discriminative Tensor Decomposition With Large Margin

We propose a new method to estimate the joint projection matrices that map a set of tensors into a more discriminative feature subspace, in the same vein as the LMNN and LMCA methods. The latter two methods [61], [9] actually focus on improving the k-nn classification accuracy by mapping the data into a new feature space. Our method differs from these two works in that it is a hybrid of the LMCA method and the Tucker decomposition. More specifically, it uses distances to project the multi-dimensional input data into a multi-dimensional feature space as in LMCA meanwhile preserving the orthogonality of the projection matrices and the low reconstruction error as in Tucker decomposition. In other words, we try to achieve simultaneously both good classification and high fidelity reconstruction of the data. To put into evidence the merit of the proposed method, we consider the case of tensor data retrieval with very limited training data. Collected data can be limited due to rare occurrence of events or to the cost/limitations of data acquisition. Thus a new tensor instance is queried when only very few sample tensors are available. The paucity of training data will hopefully be compensated for by the inherent noise filtering and stabilization in the subspace of core tensors. Pattern analysis and tensor retrieval could thus be more reliably conducted in the core tensors space.

In essence our methodology provides trade-offs between generality and locality, and more importantly between classification error and reconstruction error. To make this trade-off clear, we point out that locality is preserved in the sense that the nearest neighbors are forced to be close to each other in the projected space while generality is provided in the sense that the projection matrices are evaluated jointly over all the data set. We find projection matrices $\mathbf{U}^{(n)}, n = 1, ...N$ for every mode such that for each class each tensor is projected into a subspace where in-class tensor cores stand closer to each other while all other-classes tensor cores are forced to be as distant as possible from core instances of the target class. In addition, the projection subspace vies concomitantly to have as faithful a reconstruction of each tensor as possible from its core. In other words, the average reconstruction error of the tensors is minimized under the constraint of good class discrimination. Note that we do not execute Tucker

decomposition on individual tensors; instead we find N projection matrices, one for each mode, learned jointly by the training set tensors.

Although there are several distance function options for embedding pairwise similarities of tensor objects, in this study we have opted for the squared Frobenius norm since it is intuitive and easier to optimize.

We actually develop two versions of the method according to the evaluation criteria, and we call them the Core and the Full versions. The LMTD-C (Core) version finds the projection matrices that optimizes the classification performance relying on the core (reduced) tensors, while the LMTD-F (Full) version tries to find the projection matrices that optimize the performance using the reconstructed tensors. The LMTD-C focuses on improving discrimination over the core tensors while the reconstruction is its second concern. On the other hand, the LMTD-F targets aligning the tensors in the input spaces such that tensors could be better discriminated in the input space while the separation in the feature space is a concomitant benefit.

4.2.1. LMTD-C: Large Margin Tensor Decomposition - Core

The LMTD-C decomposition method is illustrated in Figure 4.3. As shown in the figure, the larger dimensional N-way tensors are reduced to lower dimensional N-way tensors via a set of joint projection matrices. Note that these smaller objects are not anymore the core tensors of a Tucker decomposition.

Thus each N-way core tensor $\mathcal{G}_i$ for a generic tensor $\mathcal{X}_i$ can be extracted as:

$$\begin{aligned} \mathcal{G}_i &= \mathcal{X}_i \times_1 (\mathbf{U}^{(1)})^\top \times_2 (\mathbf{U}^{(2)})^\top \times_3 \dots \\ &\quad \times_N (\mathbf{U}^{(N)})^\top \end{aligned} \tag{4.11}$$

where $\mathbf{U}^{(n)}, n = 1, \dots, N$ are the projection matrices per modes as obtained via the algorithm in Figure 4.4 (to be derived in the sequel) as in Equation 4.6. One should

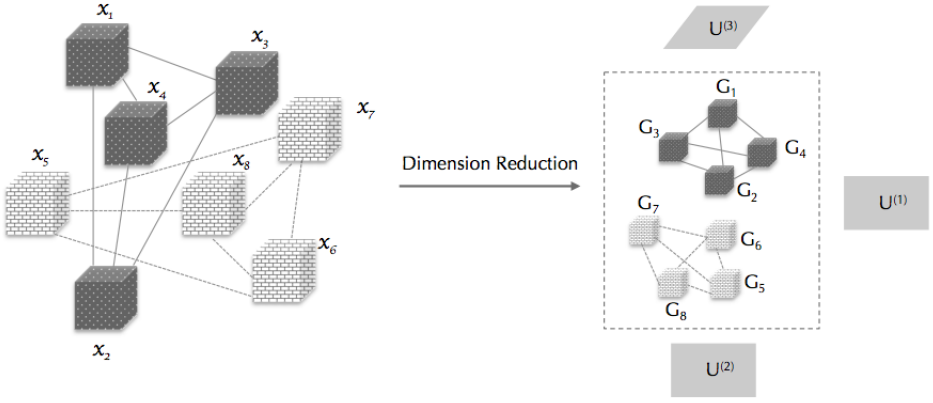


Figure 4.3. Large Margin Tensor Decomposition - Core version, illustrated for $N = 3$. Ties between tensor cubes are for illustrative purposes to show class memberships.

note that while $\mathbf{U}^\top\mathbf{U} = \mathbf{I}$, $\mathbf{U}\mathbf{U}^\top$ is not guaranteed to be $\mathbf{I}$.

The matricized version of this transformation for any one, say n, mode is:

$$\begin{aligned} \mathbf{G}_i^{(n)} &= (\mathbf{U}^{(n)})^\top \mathbf{X}_i^{(n)} \mathbf{H}^{(-n)} \\ \mathbf{H}^{(-n)} &= \Big((\mathbf{U}^{(N)})^\top \otimes (\mathbf{U}^{(N-1)})^\top \otimes \ldots \otimes (\mathbf{U}^{(n+1)})^\top \otimes \\ &\quad (\mathbf{U}^{(n-1)})^\top \otimes \ldots \otimes (\mathbf{U}^{(1)})^\top \Big)^\top \end{aligned} \tag{4.12}$$

Conversely, any tensor, $\mathcal{X}_i$, can be reconstructed approximately from its core and the joint projection matrices:

$$\hat{\mathcal{X}}_i \approx \mathcal{G}_i \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_N \mathbf{U}^{(N)} \tag{4.13}$$

If we substitute Equation 4.11 in Equation 4.13, we can express the approximate reconstruction $\hat{\mathcal{X}}_i$ of a tensor from its projections:

$$\begin{aligned}\hat{\mathcal{X}}_i &\approx \mathcal{X}_i \times_1 (\mathbf{U}^{(1)})^\top \times_2 (\mathbf{U}^{(2)})^\top \ldots \times_N (\mathbf{U}^{(N)})^\top \\ &\quad \times_1 \mathbf{U}^{(1)} \times_2 \mathbf{U}^{(2)} \ldots \times_N \mathbf{U}^{(N)} \\ &= \mathcal{X}_i \times_1 (\mathbf{U}^{(1)}(\mathbf{U}^{(1)})^\top) \times_2 (\mathbf{U}^{(2)}(\mathbf{U}^{(2)})^\top) \\ &\quad \times_3 \ldots \times_N (\mathbf{U}^{(N)}(\mathbf{U}^{(N)})^\top) \end{aligned} \tag{4.14}$$

We can see better the reconstruction of the tensor per mode if we matricize the transformation for some selected mode n:

$$\begin{aligned}\hat{\mathbf{X}}_i^{(n)} &\approx (\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top)\mathbf{W}_i^{(-n)} \\ \mathbf{W}_i^{(-n)} &= \mathbf{X}_i^{(n)} \Big((\mathbf{U}^{(N)}(\mathbf{U}^{(N)})^\top) \otimes \\ &\quad (\mathbf{U}^{(N-1)}(\mathbf{U}^{(N-1)})^\top) \otimes \\ &\quad \ldots \otimes (\mathbf{U}^{(n+1)}(\mathbf{U}^{(n+1)})^\top) \otimes \\ &\quad (\mathbf{U}^{(n-1)}(\mathbf{U}^{(n-1)})^\top) \otimes \\ &\quad \ldots \otimes (\mathbf{U}^{(1)}(\mathbf{U}^{(1)})^\top) \Big)^\top \end{aligned} \tag{4.15}$$

Given a set of K tensors with class labels the LMTD-C algorithm can be expressed as:

$$\begin{aligned}\min_{\mathbf{U}^{(1)},\ldots,\mathbf{U}^{(N)}} \quad & \mu \sum_{i=1}^{K} ||\mathcal{X}_i - \hat{\mathcal{X}}_i||_F^2 + \beta \sum_{i=1, j \rightsquigarrow i}^{K} ||\mathcal{G}_i - \mathcal{G}_j||_F^2 \\ & + \gamma \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \xi_{ijl} \\ \text{s.t} \quad & |\mathcal{G}_i - \mathcal{G}_l||_F^2 - ||\mathcal{G}_i - \mathcal{G}_j||_F^2 \geq C - \xi_{ijl} \\ & \xi_{ijl} \geq 0, \quad \forall i, \forall j, \forall l \\ & \mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top = \mathbf{I}, \forall n \end{aligned} \tag{4.16}$$

where C is an assigned margin value, which determines the safety region between an in-class and an impostor. Note that during the evaluation of the impostor distance constraints third row of Equation 4.16, each instance is compared within in its own k-nn neighborhood set, $o_l \in Ne_k(o_i)$.

It will be instructive the consider the three terms of the objective function separately. The first term with weight μ accounts for the reconstruction fidelity. Obviously the higher the value of the weight coefficient μ the lower the reconstruction error, but potentially at the detriment of classification accuracy. The second term corresponds to the sum of distances between core tensors and those of their targets. The weight coefficient β enforces the closeness of in-class core tensors for a more accurate k-nn classification. The third term penalizes impostors violating the margin condition, that is, tensors that appear closer to the target object than in-class neighbors. It consists of the sum of distances to the margin and its weight is adjusted with the coefficient γ.

If we had $\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top = \mathbf{I}_{I_n \times I_n}, \forall n$, we would have $\hat{\mathcal{X}}_i = \mathcal{X}_i$. But this might not be the case, since $\text{rank}(\mathbf{U}^{(n)}) \leq I_n$. Since $\mathbf{U}^{(n)}$ is a projection operation, we reduce the dimensions and the equality definitely does not hold ($\text{rank}(\mathbf{U}^{(n)}) < I_n$). Thus $\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top = \mathbf{I}$ spans the space of reconstruction errors and uncorrelated projection directions.

Let's manipulate the objective function and convert it to an unconstrained optimization problem. It will be more convenient to proceed with the n-mode expansion of the tensors.

$$
\begin{aligned}
L^{(C)} &= \mu \sum_{i=1}^{K} \mathrm{tr}\Big((\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})^\top (\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})\Big) + \\
&\quad \beta \sum_{i=1, j \rightsquigarrow i}^{K} \mathrm{tr}\Big((\mathbf{G}_i^{(n)} - \mathbf{G}_j^{(n)})^\top (\mathbf{G}_i^{(n)} - \mathbf{G}_j^{(n)})\Big) + \\
&\quad \gamma \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \Big[C + \\
&\quad \mathrm{tr}\Big((\mathbf{G}_i^{(n)} - \mathbf{G}_j^{(n)})^\top (\mathbf{G}_i^{(n)} - \mathbf{G}_j^{(n)})\Big) - \\
&\quad \mathrm{tr}\Big((\mathbf{G}_i^{(n)} - \mathbf{G}_l^{(n)})^\top (\mathbf{G}_i^{(n)} - \mathbf{G}_l^{(n)})\Big) \Big]_+
\end{aligned} \tag{4.17}
$$

where $[a]_+$ is the hinge loss, which yields a when $a > 0$ and is 0 otherwise. The superscript (C) in $L^{(C)}$ denotes the "Core" version of the LMTD algorithm.

Let's consider the individual terms of the loss function and define $\mathbf{\Upsilon}_{ij}^{(n)} = (\mathbf{G}_i^{(n)} - \mathbf{G}_j^{(n)})^\top (\mathbf{G}_i^{(n)} - \mathbf{G}_j^{(n)})$:

$$
\begin{aligned}
L_1 &= \sum_{i=1}^{K} \mathrm{tr}\Big((\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})^\top (\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})\Big) \\
L_2^{(C)} &= \sum_{i=1, j \rightsquigarrow i}^{K} \mathrm{tr}(\mathbf{\Upsilon}_{ij}^{(n)}) \\
L_3^{(C)} &= \sum_{i=1, j \rightsquigarrow i, l}^{M} (1 - y_{il}) \Big[C + \mathrm{tr}(\mathbf{\Upsilon}_{ij}^{(n)}) - \mathrm{tr}(\mathbf{\Upsilon}_{il}^{(n)}) \Big]_+ \\
L^{(C)} &= \mu L_1 + \beta L_2^{(C)} + \gamma L_3^{(C)}
\end{aligned} \tag{4.18}
$$

Note that the loss function is non-convex due to the lower rank projection matrices in multi-ways, hence it does not have a closed form solution and search algorithms may not always reach the global optimum. We search for an iterative solution, so that in each iteration we fix the projection matrices except for $\mathbf{U}^{(n)}$ and solve for it, $\forall n = 1, ..., N$. To this effect, we first use the gradient descent algorithm and then

we apply QR decomposition to project the solution back onto the unit sphere. The gradient for $\mathbf{U}^{(n)}$ can be computed as follows:

$$\frac{\partial L^{(C)}}{\partial \mathbf{U}^{(n)}} = \mu \frac{\partial L_1}{\partial \mathbf{U}^{(n)}} + \beta \frac{\partial L_2^{(C)}}{\partial \mathbf{U}^{(n)}} + \gamma \frac{\partial L_3^{(C)}}{\partial \mathbf{U}^{(n)}} \tag{4.19}$$

We define the following shorthand notations: $\mathbf{P}_{ij}^{(-n)} = \mathbf{X}_i^{(n)} \mathbf{H}^{(-n)} (\mathbf{H}^{(-n)})^\top (\mathbf{X}_j^{(n)})^\top$, where $\mathbf{H}^{(-n)}$ is defined in Equation 4.12, $\mathbf{T}^{(n)} = \mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top$, $\mathbf{V}_{ij}^{(-n)} = \mathbf{W}_i^{(-n)} (\mathbf{W}_j^{(-n)})^\top$, $\mathbf{\Phi}_{ij}^{(-n)} = \mathbf{P}_{ii}^{(-n)} - \mathbf{P}_{ji}^{(-n)} - \mathbf{P}_{ij}^{(-n)} + \mathbf{P}_{jj}^{(-n)}$ and $\mathbf{W}_i^{(-n)}$ is as in Equation 4.15. The expression for the gradient of the loss function vis-à-vis $\mathbf{U}^{(n)}$ becomes:

$$\begin{aligned}
\frac{\partial L^{(C)}}{\partial \mathbf{U}^{(n)}} = \; & 2\mu \sum_{i=1}^{K} \Big[- \mathbf{W}_i^{(-n)} (\mathbf{X}_i^{(n)})^\top - \mathbf{X}_i^{(n)} (\mathbf{W}_i^{(-n)})^\top + \\
& \mathbf{V}_{ii}^{(-n)} \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \mathbf{V}_{ii}^{(-n)} \Big] \mathbf{U}^{(n)} + \\
& 2\beta \sum_{i=1, j \rightsquigarrow i}^{K} \left[\mathbf{\Phi}_{ij}^{(-n)} \right] \mathbf{U}^{(n)} + \\
& 2\gamma \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \left[\mathbf{\Phi}_{ij}^{(-n)} - \mathbf{\Phi}_{il}^{(-n)} \right] \mathbf{U}^{(n)}
\end{aligned} \tag{4.20}$$

The LMTD-C algorithm is sketched in Figure 4.4. Note that the max norm ($\|\mathbf{U}\|_{\max} = \max_{ij} |U_{ij}|$) is used for preventing numerical instabilities and scaling the gradient such that its entries are guaranteed to be between -1 and 1.

The number of nearest neighbors and the margin C play critical roles in the number of impostors that we have to deal. The higher the value of k and/or of C, the more impostors there are. As the margin goes wider, the algorithm searches over a bigger hypersphere, hence potentially one can find more impostors. Similarly, the bigger the number k of nearest neighbors we search for, the higher the likelihood of encountering impostors. One has to check for impostors in every iteration since an update in the projection matrices may cause some impostors to vanish and new ones to emerge.

Require the set of tensor objects, $\{\mathcal{X}^{(m)}\}_{m=1}^{M}$, their class labels $c_m \in \{1, 2, \ldots, N_c\}$, the reduced dimensions $I'_1 \times I'_2 \times \ldots \times I'_N$, the number of neighbors (by default, $k = 3$), the weight parameters (by default, $\mu = 1, \beta = 1, \gamma = 1, \alpha = \prod_{n=1}^{N} I_n, C = \prod_{n=1}^{N} I'_n$) and the learning rate (by default, $\rho = 0.01$).

Determine the target neighbors in the input space and fix them.

Initialize $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$

for $t = 1$ to T_{max} **do**

  for $n = 1$ to N **do**

    Determine the impostors for each reduced instance, $\mathcal{G}^{(m)}$

    Calculate $\frac{\partial L^{(C)}}{\partial \mathbf{U}^{(n)}}$

    $\mathbf{U}^{(n)} \leftarrow \mathbf{U}^{(n)} - \rho\Big(\Big\| \frac{\partial L^{(C)}}{\partial \mathbf{U}^{(n)}} \Big\|_{\max} \Big)^{-1} \frac{\partial L^{(C)}}{\partial \mathbf{U}^{(n)}}$

    $\mathbf{Q}, \mathbf{R} = \text{QRDecomposition}(\mathbf{U}^{(n)})$

    $\mathbf{U}^{(n)} \leftarrow \mathbf{Q}$

  end for

end for

return $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$.

Figure 4.4. Algorithm of LMTD-C.

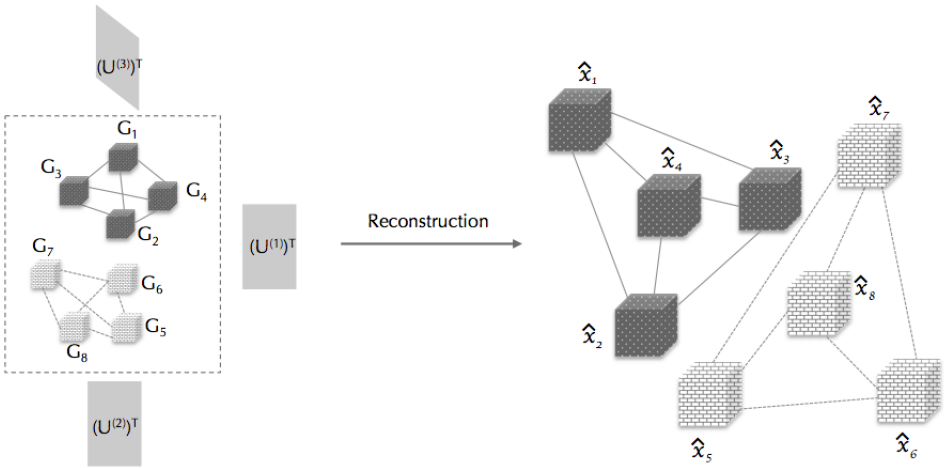


Figure 4.5. Tensors reconstructed to original dimensions from core tensors as in Figure 4.3 to be used in Large Margin Tensor Decomposition - Full.

Any rule can be used in the initialization of the projection matrices $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$; in fact, they can even be assigned randomly. But to have a consistent and sensible starting point, we have calculated the individual Tucker decompositions of tensors and averaged over the individual projection matrices. Thus the initial projection matrices are derived from the singular vectors of the averaged Tucker projection matrices obtained by singular value decomposition (SVD).

4.2.2. LMTD-F: Large Margin Tensor Decomposition - Full

An alternative adaptation of the LMTD approach emphasizes the similarity between in-class reconstructed tensors and their original versions in the k-nearest neighborhood and similarity the dissimilarity between reconstructed impostors and in-class neighbor, while paying tribute as well to the classification accuracy. LMTD-F distills tensors so that both reconstruction and discriminant qualities are directly guaranteed; LMTD-C distills tensors for good discrimination while similarity of reconstructed tensors is indirectly provided for based on the similarity of the core tensors. In essence, both are approaches to improve the accuracy of the k-nn classifier thanks to the large margin constraints embedded. The LMTD-F algorithm is visualized in Figure 4.5.

The formulation of LMTD-F is given below:

$$\begin{aligned}
\min_{\mathbf{U}^{(1)},\ldots,\mathbf{U}^{(N)}} \quad & \mu \sum_{i=1}^{K} ||\mathcal{X}_i - \hat{\mathcal{X}}_i||_F^2 + \beta \sum_{i=1, j \rightsquigarrow i}^{K} ||\hat{\mathcal{X}}_i - \hat{\mathcal{X}}_j||_F^2 \\
& + \gamma \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \xi_{ijl} \\
\text{s.t} \quad & ||\hat{\mathcal{X}}_i - \hat{\mathcal{X}}_l||_F^2 - ||\hat{\mathcal{X}}_i - \hat{\mathcal{X}}_j||_F^2 \geq C - \xi_{ijl} \\
& \xi_{ijl} \geq 0, \quad \forall i, \forall j, \forall l \\
& \mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top = \mathbf{I}, \forall n.
\end{aligned} \tag{4.21}$$

Similar to LMTD-C, let's manipulate the objective function and convert it to an unconstrained optimization problem, which is legitimate for any mode (any n).

$$\begin{aligned}
L^{(F)} \quad = \quad & \mu \sum_{i=1}^{K} \mathrm{tr}\Big((\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})^\top (\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)}) \Big) + \\
& \beta \sum_{i=1, j \rightsquigarrow i}^{K} \mathrm{tr}\Big((\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)})^\top (\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)}) \Big) + \\
& \gamma \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \Big[C + \\
& \mathrm{tr}\Big((\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)})^\top (\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)}) \Big) - \\
& \mathrm{tr}\Big((\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_l^{(n)})^\top (\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_l^{(n)}) \Big) \Big]_+
\end{aligned} \tag{4.22}$$

The three components of the LMTD-F objective function read as follows:

$$\begin{aligned} L_1 &= \sum_{i=1}^{K} \mathrm{tr}\Big((\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})^\top (\mathbf{X}_i^{(n)} - \hat{\mathbf{X}}_i^{(n)})\Big) \\ L_2^{(F)} &= \sum_{i=1, j \rightsquigarrow i}^{K} \mathrm{tr}\Big((\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)})^\top (\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)})\Big) \\ L_3^{(F)} &= \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \Big[C + \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)})^\top (\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_j^{(n)})\Big) - \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_l^{(n)})^\top (\hat{\mathbf{X}}_i^{(n)} - \hat{\mathbf{X}}_l^{(n)})\Big) \Big]_+ \end{aligned} \tag{4.23}$$

This loss function is similarly non-convex and a reasonable locally optimum solution can be obtained via gradient descent optimization. The gradient of the objective function is the sum of the gradients of the three loss components.

$$\frac{\partial L^{(F)}}{\partial \mathbf{U}^{(n)}} = \mu \frac{\partial L_1}{\partial \mathbf{U}^{(n)}} + \beta \frac{\partial L_2^{(F)}}{\partial \mathbf{U}^{(n)}} + \gamma \frac{\partial L_3^{(F)}}{\partial \mathbf{U}^{(n)}} \tag{4.24}$$

The complete gradient of the loss function is evaluated as follows, where $\mathbf{V}_{ij}^{(-n)}$, $\mathbf{W}_i^{(-n)}$ and $\mathbf{T}^{(n)}$ have been defined in LMTD-C and $\mathbf{\Psi}_{ij}^{(-n)} = \mathbf{V}_{ii}^{(-n)} - \mathbf{V}_{ji}^{(-n)} - \mathbf{V}_{ij}^{(-n)} + \mathbf{V}_{jj}^{(-n)}$:

$$\begin{aligned} \frac{\partial L^{(F)}}{\partial \mathbf{U}^{(n)}} &= 2\mu \sum_{i=1}^{K} \Big[- \mathbf{W}_i^{(-n)} (\mathbf{X}_i^{(n)})^\top - \mathbf{X}_i^{(n)} (\mathbf{W}_i^{(-n)})^\top + \\ &\quad \mathbf{V}_{ii}^{(-n)} \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \mathbf{V}_{ii}^{(-n)} \Big] \mathbf{U}^{(n)} + \\ &\quad 2\beta \sum_{i=1, j \rightsquigarrow i}^{K} \left[\left(\mathbf{\Psi}_{ij}^{(-n)} \right)^\top \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \left(\mathbf{\Psi}_{ij}^{(-n)} \right)^\top \right] \mathbf{U}^{(n)} + \\ &\quad 2\gamma \sum_{i=1, j \rightsquigarrow i, l}^{K} (1 - y_{il}) \left[\left(\left(\mathbf{\Psi}_{ij}^{(-n)} \right)^\top \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \left(\mathbf{\Psi}_{ij}^{(-n)} \right)^\top \right) \mathbf{U}^{(n)} - \right. \\ &\quad \left. \left(\left(\mathbf{\Psi}_{il}^{(-n)} \right)^\top \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \left(\mathbf{\Psi}_{il}^{(-n)} \right)^\top \right) \mathbf{U}^{(n)} \right]_+ \end{aligned} \tag{4.25}$$

Require the set of tensor objects, $\{\mathcal{X}^{(m)}\}_{m=1}^{M}$, their class labels $c_m \in \{1, 2, \ldots, N_c\}$, the reduced dimensions $I'_1 \times I'_2 \times \ldots \times I'_N$, the weight parameters (by default, $\mu = 1, \beta = 1, \gamma = 1, \alpha = \prod_{n=1}^{N} I_n, C = \prod_{n=1}^{N} I_n$) and the learning rate (by default, $\rho = 0.01$).

Determine the target neighbors in the input space and fix them.

Initialize $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$

for $t = 1$ to T_{max} **do**

  for $n = 1$ to N **do**

    Determine the impostors for each reconstructed instance, $\hat{\mathcal{X}}^{(m)}$

    Calculate $\frac{\partial L^{(F)}}{\partial \mathbf{U}^{(n)}}$

    $\mathbf{U}^{(n)} \leftarrow \mathbf{U}^{(n)} - \rho\left(\left\| \frac{\partial L^{(F)}}{\partial \mathbf{U}^{(n)}} \right\|_{\max} \right)^{-1} \frac{\partial L^{(F)}}{\partial \mathbf{U}^{(n)}}$

    $\mathbf{Q}, \mathbf{R} = \text{QRDecomposition}(\mathbf{U}^{(n)})$

    $\mathbf{U}^{(n)} \leftarrow \mathbf{Q}$

  end for

end for

return $\{\mathbf{U}^{(n)}\}_{n=1}^{N}$.

Figure 4.6. Algorithm of LMTD-F.

The full algorithm to apply LMTD-F is given in Figure 4.6.

5. APPLICATIONS, EXPERIMENTS AND RESULTS

We experiment the proposed methods with the simulated data for the cyber-security system and with image and video data for the tensor decomposition. Both of them are compared with their competitors from the literature. The experimentation and comparison setup of each method are given in separate sections.

5.1. DDoS Detection and Attacker Discrimination

This section focuses on experimentation of the proposed DDoS detection and attacker discrimination system. Besides the performance comparison with the competitors, different setups are experimented to determine the performance of the proposed system under different influences such as the effects of traffic intensity and the observation window length, or overlapping attacks.

5.1.1. Simulation Environment

As it is often reported in the literature, we have also found that obtaining and getting the permission to use VoIP server data sets proves to be very problematic, mostly due to privacy concerns of the subscribers and the commercial secrecy concerns of the telecommunication operators. Therefore, we have used simulated data sets to analyze the performance of the change point detection model, detailed in Section 3.1, and of the malicious user identification algorithm, given in Section 3.2. An Asterisk-based PBX software, named *Trixbox*, is deployed as the SIP server in a virtual machine [64]. To mimic the traffic on a SIP server, we have built a probabilistic SIP network simulation system, which initiates calls between a number of probabilistically chosen users in real-time [65, 66]. An application that creates the user agents is deployed on another virtual machine. We have used PJSIP open source library [67] and implemented it in Python language. Lastly NOVA V-Spy, a vulnerability scanning tool, is installed on a final virtual machine and is used to simulate DDoS attacks targeting the server [68]. An overview of the simulation environment is provided in Figure 5.1. The proposed

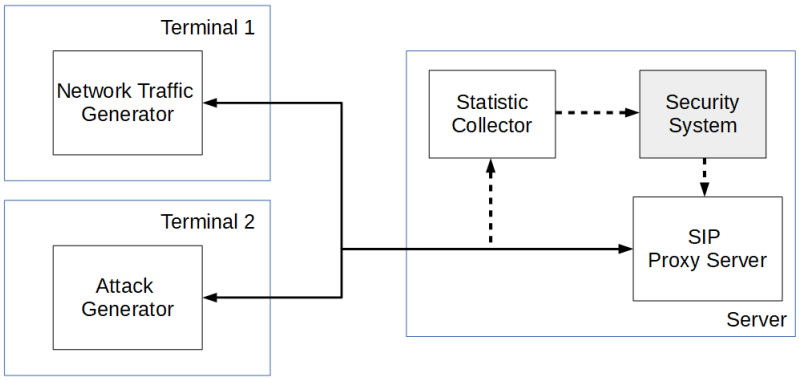


Figure 5.1. SIP Network Simulation Framework.

security system runs on the same machine with the SIP server, as represented with a gray box in Figure 5.1.

The traffic simulator, based on a probabilistic model, generates real-time SIP messaging traffic among registered subscribers [65]. The probabilistic model is basically a library that initiates all permitted actions of subscribers in generating real-life SIP messaging traffic through a SIP server. Instances of subscriber actions are: The potential callees and callers (the social network), how likely to call a certain contact (the phone book), how often to become active (registration frequency to the SIP server), how long to wait before the next call (the call frequency), how likely to make a call (the call probability), how likely to answer an incoming call (the response probability), and how long to talk on the phone (the call duration). The parameters provided to the simulator determine the behavior of probabilistic model and therefore statistically the actions of subscribers. The environmental parameters of the simulator are the total number of subscribers in the SIP server can serve and the number of social groups, where a social group is defined as the subset of subscribers that are more likely to interact with each other as compared to the rest. All subscribers are created as bots on the simulation machine and they all follow legitimate messaging rules of the protocol. An existing subscriber bot is active as long as its registration on the SIP server has not expired; therefore only active bots can interact with each other.

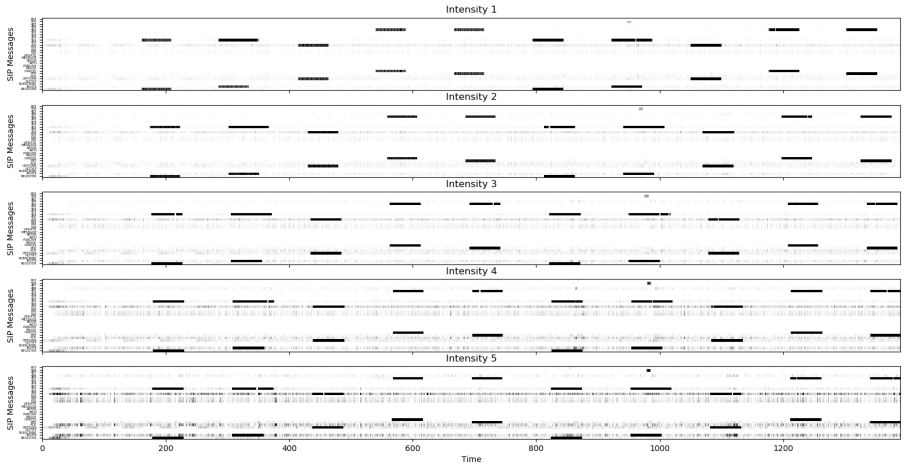


Figure 5.2. Illustration of traffic intensities generated by the simulator.

Data are collected by inspecting each SIP packet that arrives to or is sent by the server. Counts of 14 SIP request and 14 SIP response packets are recorded periodically for each time unit (which is assumed to be 1 second in our experiments) and the 28-dimensional vector, made up of packet counts per unit interval, constitutes the input data. The SIP message types, which are described in details in RFC 3261 [4], for which we record periodically the counts are as follows:

- Requests: Register, Invite, Subscribe, Notify, Options, Ack, Bye, Cancel, Prack, Publish, Info, Refer, Message, Update
- Responses: 100, 180, 183, 200, 400, 401, 403, 404, 405, 481, 486, 487, 500, 603

The experimental environment is controlled by two parameters: the intensity of the background traffic, that is, the normal-user traffic and the intensity of DDoS attack traffic. In our simulation system at any time there are 200 active registered subscribers. There are 5 levels of preset normal traffic intensity created collectively by the subscriber bots. The normal subscriber bots, on the average generate a total of 5, 10, 20, 40 and 80 call attempts among themselves (0.025 to 0.4 message/bot), in any observation (1 sec) interval. We grade these background traffic intensities as levels

from 1 to 5. Figure 5.2 exhibits these traffic intensities for a simulation setting. Note that the gray tones in the plots are proportional to the message counts so that the darker a region in the plot, the higher the number of that type messages observed in that interval. White represents intervals with no messages and intervals with a count higher than 200 messages are shown in pitch black.

During a DDoS attack, for a given setup, as long as not explicitly stated otherwise, 10 randomly selected users, that is 5 percent of the subscribers, play the role of attackers. During attacks, the attackers start sending messages more intensely to the SIP server. In the low-level attack setting, each attacker sends on the average 50 messages, while in a high-level attack, their rate becomes 100 messages per second. In each run, 10 DDoS attack sessions are simulated, consisting of attacks using the five types of messages (Invite, Register, Options, Cancel and Bye) and each carried out once with low intensity and once with high intensity. The runs are repeated ten folds, such that in each fold attacks occur in a different order and a different set registered subscribers are selected to act as attackers. In Figure 5.2 the darker regions correspond to attacks.

The experiments are executed in a 10-fold cross-validation setup. One dataset is used for determining the parameters of the distance change point model, and the remaining nine datasets are run with the estimated parameters. Recall that the distance-based change-point detector had three different parameters; we apply a grid search to find the best parameters ($k = \{5, 7, 9, 11\}$, $\lambda = \{1.0, 2.0, 4.0\}$, $\beta = \{1.0, 2.0, 4.0\}$ and an additional fourth one for χ^2 thresholding $\alpha = \{0.01, 0.02\}$). The default values are set for the parameters of the time series alignment kernel as ($\gamma = 1.0$ and $\rho = 1.0$). For the ARIMA model, we perform an exhaustive search to find its optimal parameters $p = \{1, 2, 3, 5, 10\}, d = \{0, 1, 2\}$ and $q = \{0, 1, 2, 3, 5, 10\}$.

Since we know the labels (the attack times and the identity of attackers) in the simulated data, we can evaluate the performance of the proposed system in terms of the F-measure. In the ideal case the F-measure would be 1, which can be obtained only when there are no falsely accused users (i.e., precision P = 1), all the attackers are

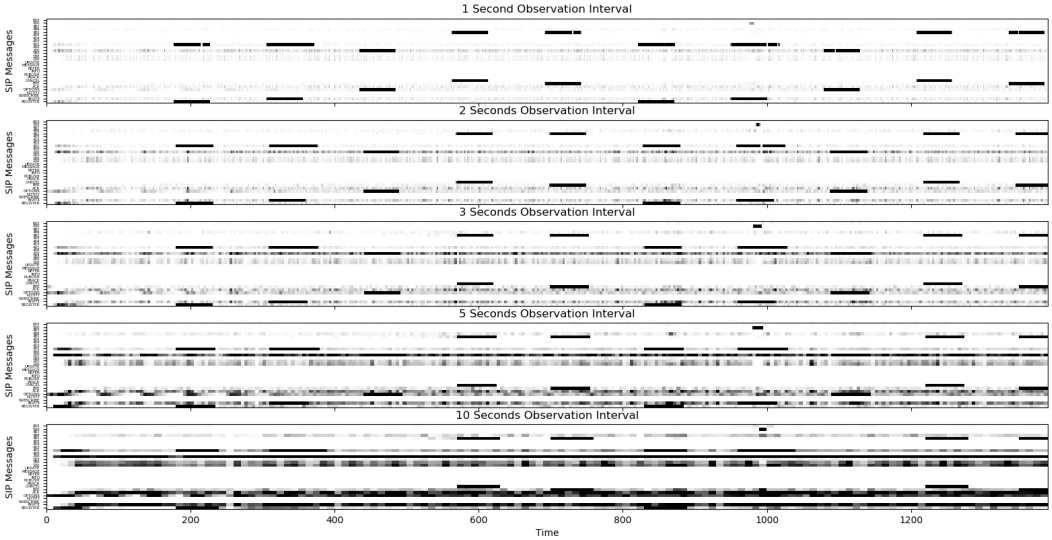


Figure 5.3. Illustration of traffic intensities as a function of observation interval. All traffic is generated at level 3.

identified correctly (i.e., recall R = 1), and all the change points are identified by the change point detector. The precision, recall and F-measure are evaluated as follows, for the case of malicious users:

$$\text{Precision (P)} = \frac{\#\text{ detected true malicious users}}{\#\text{ all detected malicious users}} \tag{5.1}$$

$$\text{Recall (R)} = \frac{\#\text{ detected true malicious users}}{\#\text{ all true malicious users}} \tag{5.2}$$

$$\text{F-measure (F)} = 2\frac{PR}{P+R} \tag{5.3}$$

For change point detection performance, we can replace the arguments of P and R with detected true change points, all detected change points, and all true change points.

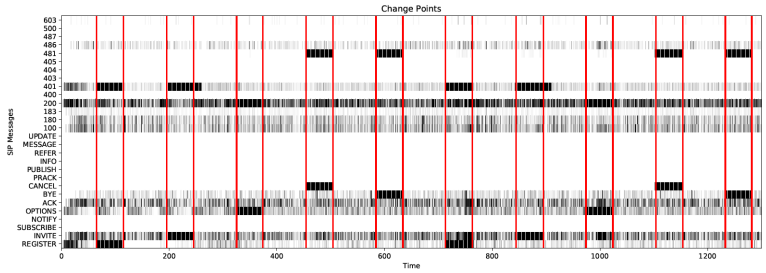
We have experimented with the duration of the observation interval. Figure 5.3 shows the effect of the length of the observation interval for the normal traffic setting. Not surprisingly, as the sampling interval increases, the messaging counts also go higher. A few words are in order to explain Figure 5.3: The abscissa represents real time in

seconds while an observation in taken every T seconds, $T = 1, 2, ..., 10$. We used in the graphic, the same gray tone is used to represent the observed count vector, hence the appearance of stretched bars. Furthermore, the longer the observation interval, the more the number of messages seen for any type, and consequently the plots become have darker areas.

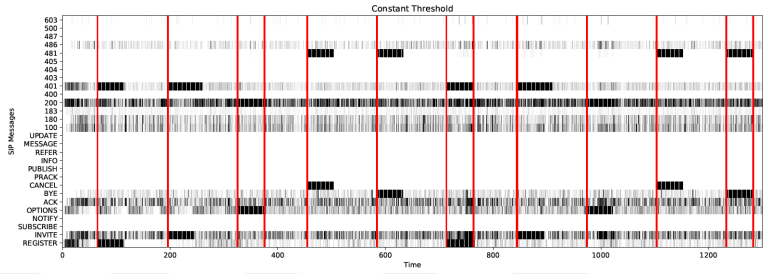
5.1.2. Comparison with a Competitor Algorithm

Figure 5.4 shows the performance of our algorithm in detecting the onset and offset instants of the DDoS attacks. This figure is illustrative in that for the each simulation traffic setting, the best parameters are chosen for the models found via grid search. These parameters used for performance comparisons are given in each table. The ordinate lists the 28 types of SIP messages, the abscissa shows the time in seconds, and the levels of gray show the intensity of messages. The red lines indicate the change point instants found by the algorithms. The experiments demonstrate that both proposed methods of thresholding are successful in detecting the onset of attacks, but they may sometimes fail to detect the offset. The possible reason is that an attack causes an abrupt change against the background of normal user traffic; however, after that the incoming message intensity subsides at the end of an attack, its aftershock effects linger on at the server side with server response messages. The ARIMA model often fails to raise alarms at the correct instances and it is also affected by the short-time small fluctuations in the counts. Therefore, it gives incorrect onset and offset indications.

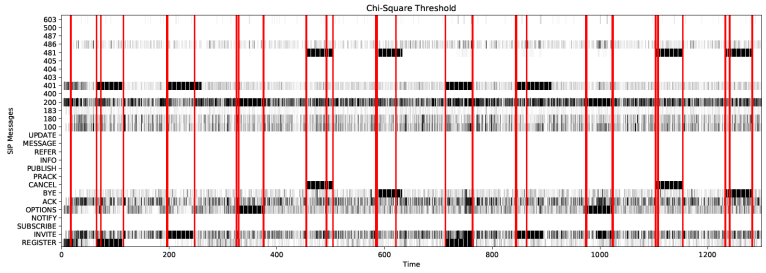
Attack Onset and Offset Determination: In our evaluations, we consider detecting the start and end instants of the attacks. Therefore, we measure the number of attacks for which the onset and offset are correctly detected as well the miss and false alarm probabilities (errors of the first type and of the second type). For the ARIMA model, we look at the start and end point of a contingent period which is detected as an anomaly. Since an attack engenders a different behavior in the network (anomaly), the anomaly detector should be able to detect the start and the end of an attack. During the comparison, we use the start and end times of continuous intervals detected as the



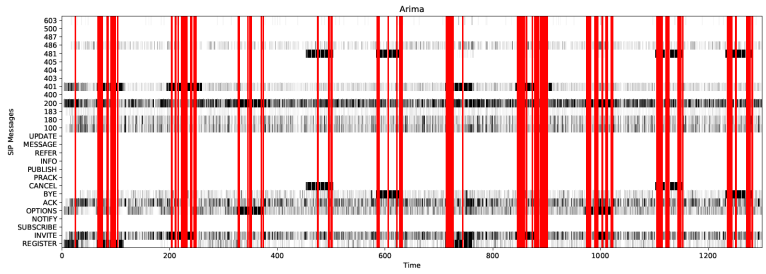
a) The ground-truth change points.



b) The change points detected by the constant thresholding DCPM ($k = 5, \lambda = 4.0, \beta = 4.0, c = 1$).



c) The change points detected by the χ^2 thresholding DCPM ($k = 11, \lambda = 4.0, \beta = 4.0, \alpha = 0.02$).



d) The alarms raised by the ARIMA model ($p = 2, d = 1, q = 1$).

Figure 5.4. The change points and alarms raised by the models. The first five attacks are low-level (50 messages per attacker), while the last five attacks are high-level (100 messages per attacker).

Table 5.1. The performance of the change-point detectors for normal traffic for 1 second (For constant thresholding $k = 5, \lambda = 1, \beta = 4, c = 1$, for χ^2 thresholding $k = 5, \lambda = 2, \beta = 2, \alpha = 0.02$, for ARIMA $p = 2, d = 1, q = 0$).

Change-Point Detector	Precision	Recall	F-Score
Constant-Thresholding DCPM	0.70 ± 0.04	0.88 ± 0.07	0.79 ± 0.04
χ^2 - Thresholding DCPM	0.81 ± 0.07	0.73 ± 0.10	0.77 ± 0.03
ARIMA	0.25 ± 0.11	0.15 ± 0.09	0.25 ± 0.04

change-points for a fair comparison.

The 10-fold cross-validation performance scores of the distance change-point detector and the ARIMA based DDoS detector are given in Table 5.1. Both thresholding functions are deployed. The onset and offset times of the attacks are known and they are compared with the change-point times returned by the models. For the ARIMA model, the change points are assumed to correspond to the time instances where the alarms are raised (onset) and the alarms are silenced (offset).

To assess the attack detection performance of the DCPM (Distance based Change-Point Method) algorithm vis-a-vis to an alternative method, we have run simulation experiments methods with a method from the literature, an ARIMA-based DDoS detector [27]. The rationale for the choice of this competitor algorithm is that it was the only model we could find in the literature operating in an online and unsupervised mode. At this stage we use only one of the thresholding methods, namely χ^2 thresholding as in Equation 3.5, since the two methods yield comparable results. Their comparative performance are given in Table 5.1 The proposed methods have higher performance scores than ARIMA. The main reason is that the ARIMA detector fails to behave consistently in the attack interval. It gives false onsets and offsets during an ongoing attack. The DCPM methods give comparably close scores, but it should be noted that the parameters should be set with respect to system characteristics such as tolerance to false alarms or traffic intensity.

Table 5.2. The performance of the change-point detectors for normal traffic intensity for different sampling rates (For constant thresholding $k = 5, \lambda = 1, \beta = 4, c = 1$, for χ^2 thresholding $k = 5, \lambda = 2, \beta = 2, \alpha = 0.02$).

Detector	Score	1 sec	2 secs	3 secs	5 secs	10 secs
Constant	Precision	0.70±0.04	0.72± 0.04	0.73±0.04	0.74±0.02	0.77±0.01
	Recall	0.88±0.07	0.92±0.04	0.97±0.02	0.98±0.01	0.99±0.01
	F-Score	0.79±0.04	0.81±0.04	0.83±0.02	0.84±0.01	0.87±0.01
χ^2	Precision	0.81±0.07	0.83±0.05	0.85±0.04	0.87±0.03	0.92±0.02
	Recall	0.73±0.10	0.75±0.04	0.76±0.03	0.80±0.04	0.84±0.02
	F-Score	0.77±0.04	0.79±0.04	0.81±0.05	0.84±0.02	0.88±0.01

5.1.3. Effect of the Observation Interval Length

Table 5.2 shows that increasing observation interval improves the accuracy of the system; in fact the F-score increases by 10 points when the interval is augmented from 1 to 10 seconds. The obvious reason is that the longer observation interval makes the attack traffic statistics increasingly more distinct from the background. However this improvement comes at the price of reduced time resolution, where the onset and offset instances of the attack become proportionally blurred.

5.1.4. Effect of Traffic Intensity

Table 5.3 shows the effect of traffic intensity over the performance of the change point detector. Even though the F-scores of the detector running with empirical and statistical thresholds are similar, their precision and recall scores differ. The χ^2 threshold detector has higher precision resulting in lesser false alarms, but it might miss more frequently an attack. On the other hand, the constant thresholding is more successful in detecting an attack but it results in more false alarms. Not surprisingly as the background traffic intensity increases, the detection performance decreases. Obviously the fluctuations in the normal traffic confound the attack traffic, which becomes less

Table 5.3. The performance of the change-point detectors for different traffic intensity levels for 1 second (For constant thresholding $k = 5, \lambda = 1, \beta = 4, c = 1$, for χ^2 thresholding $k = 5, \lambda = 2, \beta = 2, \alpha = 0.02$).

Detector	Score	level 1	level 2	level 3	level 4	level 5
Constant	Precision	0.77±0.03	0.73±0.05	0.70±0.04	0.69±0.06	0.68±0.08
	Recall	0.90±0.03	0.88±0.04	0.88±0.07	0.85±0.06	0.83±0.07
	F-Score	0.82±0.04	0.8±0.07	0.79±0.04	0.77±0.07	0.75±0.07
χ^2	Precision	0.88±0.06	0.85±0.04	0.81±0.07	0.81±0.08	0.79±0.06
	Recall	0.79±0.03	0.78±0.03	0.73±0.10	0.72±0.06	0.7±0.08
	F-Score	0.83±0.05	0.81±0.05	0.77±0.04	0.76±0.06	0.74±0.08

distinctive. Conversely, when the background traffic is low, the abrupt changes caused by the attacks are easier to detect. The optimal set of parameters should be sought for each traffic intensity.

5.1.5. Effect of Overlapping Attack Intervals

Figure 5.5 illustrates the flexibility of the proposed model. In this instance, the register attack is applied incrementally such that at events distanced in time by 80-90 seconds. A new set of 10 attackers starts an attack and at the same time the intensity of their attack is increased by additional steps of 5 messages. For example, a set of 10 attackers starts at the 175th second with 5 register messages per second, resulting in a total of 50 register messages per second; then a different set of 10 attackers starts at the 255th second with 10 register messages per second, resulting in a total of 100 messages per second etc. The final set of attackers sends 50 register messages per second per attacker. When we set $\lambda = \beta = 1$, for the fixed threshold model, the algorithm is able to detect the start and the end of the attacks when $c \leq 3$. For the chi-square thresholding, the algorithm is able to detect the onset and offsets when $\alpha > 0.005$.

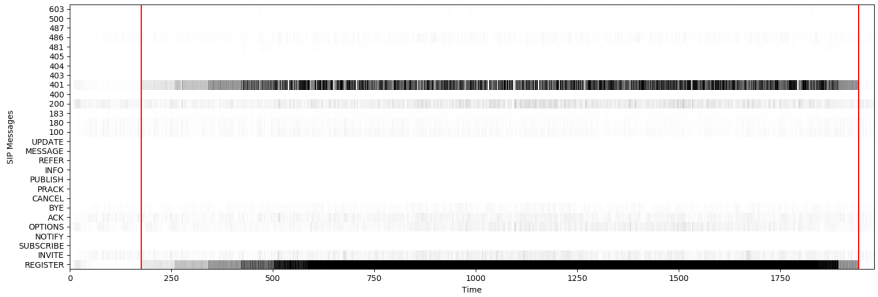


Figure 5.5. Register attacks increasing at incremental steps of 5.

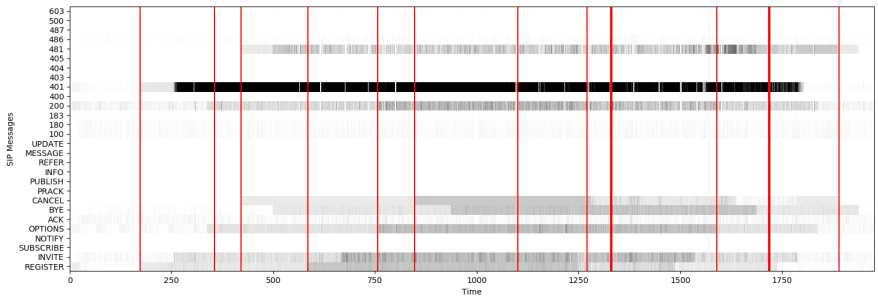


Figure 5.6. Overlapping mixed types of attack increasing at incremental steps of 5.

5.1.6. Detection Performance for Time Overlapped Attacks

Figure 5.6 shows the performance of the detector when the attacks are overlapping. The vertical bars in the figure indicate the detected onsets and offsets of the anomalous traffic when a fixed threshold is used $k = 5, \lambda = \beta = 1$ and $c = 1$. The χ^2 thresholding for $\alpha = 0.02$ shows very similar performance. Each attack type is executed twice with 10 different attackers each time. The first occurrences are with 5 messages per second and the second ones are with 10 messages per second. For example, the first cancel attack starts with 50 messages (5 cancel messages * 10 attackers) per second around 400th second and the second cancel attack starts with 100 messages per second around 850th second.

5.1.7. Effects of DCPM Parameters

The λ and β parameters provide the trade-off between aging and agility of the system. If the aging parameter λ is set to a value higher than β, the system is more resistive to the current change in the system and it is biased to keep its status quo. On the contrary, a higher β value means the system is unbiased to any change and its effects to the system will be eliminated sooner.

In the case of χ^2 thresholding, the α parameter determines the tolerance to false alarms. If it is set to a high value (e.g $\alpha = 0.1$), the algorithm is more likely to raise an alarm in case of an abrupt change even though it may not be caused by an attack. If it has a low value (e.g., $\alpha = 0.01$), then the number of false alarm decreases. The c parameter plays a role similar to the α parameter of χ^2 thresholding, in that c determines the tolerance for the false alarms. Setting it to low values (e.g. $c = 0.5$) may cause even a fluctuation of the normal traffic to raise an alarm. Conversely, for its high values (e.g., $c = 5$), the attacks might go undetected.

Figure 5.7 shows the Receiver Operating Characteristic (ROC) curve for the DCPMs for all other parameters fixed other than c and α, which are the constant threshold coefficient and the significance level, respectively. Both c and α decrease while we traverse along the curves.

5.1.8. Performance of Attacker Identification Methods

To assess attacker identification performance, we experiment the two proposed spectral clustering methods, the one based on the time series kernel and the other on the distance kernel. As a competitor method for attacker identification, we use the time series clustering method proposed in [39]. In the latter method, dynamic time warping distance is used for calculating the distances between time series having different lengths, and a one-nearest neighbor network is thus extracted. The performances of these three attacker identification methods are given in Table 5.4. Notice that the attacker identification methods are run in an unsupervised setting. This is a viable ap-

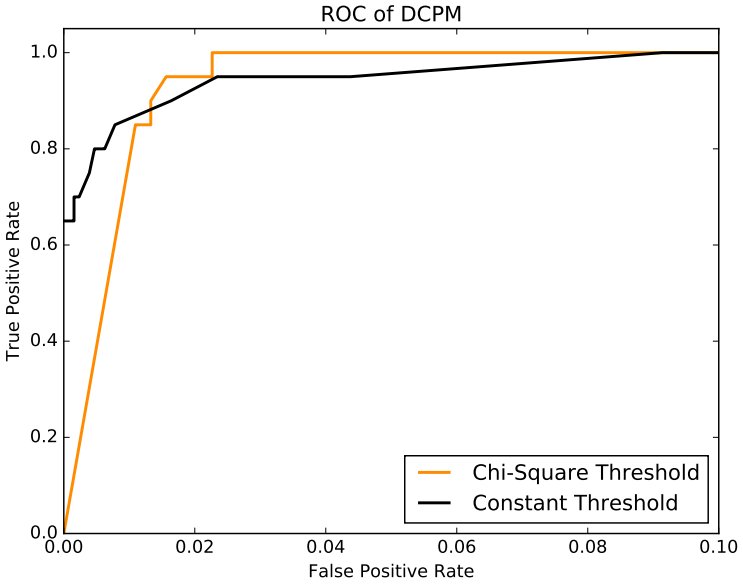


Figure 5.7. The Receiver Operating Characteristics curve of Distance based Change-Point Models as α goes from 0.1 to 0.01 by 0.01 decrease and c goes from 1.0 to 0.65 by 0.05 decrease.

proach since most of the time, in reality, labeled training data would not be available. This is due to either the changing characteristics of attack models, e.g. a zeroth day attack, or privacy and prestige concerns of the service providers.

Figure 5.8 shows the normalized kernel matrices, calculated according to the time series kernel and the distance kernel, respectively, as in Equations 3.9-3.10 and Equation 3.11. These matrices represent the messaging behavior similarity of the set of 200 users, as it is set in this experiment: In Figure 5.8 however, we plot the behavioral similarity of a subset of 25 users for clarity of illustration. 10 of the 200 users (though only the behaviour of only 25 - 10 = 15 of the normal users are plotted) mount a DDoS attack, as shown in Figures 5.8a and 5.8b. The users with similar behavior patterns have kernel values close to 1 (dark cells), while the uncorrelated users have kernel values close to 0 (white cells). Note that for the similarity values between the attackers (the attacker-attacker cells), there are some gray shades implying a modest similarity. But the attacker-normal user cells are almost all completely white (close to

0) because malicious and innocent users have totally different behaviors. In summary, the attackers are closer to each other than to the normal users.

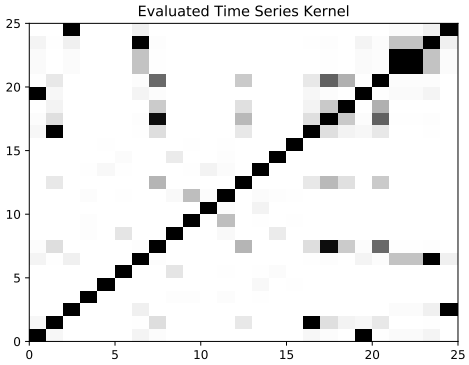
Figure 5.8c shows the attacker group labeled according to the 2-means clustering and malicious-user differentiation heuristics in Figure 3.7. The dark red cells correspond to attacker pairs; $\mathbf{K}_{q,r}$ is 1 (dark red) if (u_q, u_r) is an attacker pair. Both detectors correctly segregate the 10 malicious users from the remaining 15 innocent users.

Figure 5.9 shows how the same subscribers in Figure 5.8 are mapped by the spectral clustering. Note that in the plots, the number of points are less than 25. The reason is that some users overlap in the reduced 2-dimensional space.

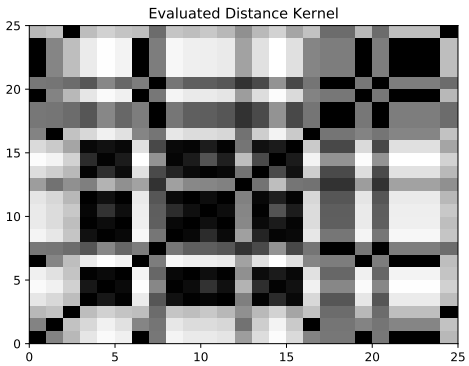
The performance of the attacker identification algorithms are given in Table 5.4. The identities of the attackers and of the normal users are known since we are using simulation data. The labels of the attackers returned by the models are compared with true labels of the users, and performance scores are computed accordingly. The three attacker identifiers, two of which are the models proposed in this work (distance and time series kernels) and the third one is the time series clustering model [39], are comparatively evaluated. The time series kernel algorithm yields the best performance scores. The time series clustering model [39] has the worst performance. The most likely reason for the lower performance of the latter is that this model uses Euclidean distance, hence does not use any data-driven weighting, for the calculation of dynamic time warping distance. The experimental results, in Table 5.4, show that the two proposed methods have almost the same F-score values.

5.1.9. Time Comparison of Attacker Identification Methods

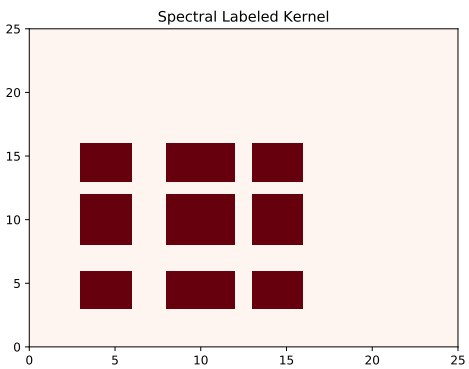
The time series kernel approach used in attacker detection is more accurate than the distance kernel, but requires longer run times. The run-times of times-series clustering model are between the other two models, but it has significantly lower accuracy. Similar conclusions can be drawn for other traffic intensities and observation intervals.



a) The evaluated time series kernel matrix ($\gamma = \rho = 1$).

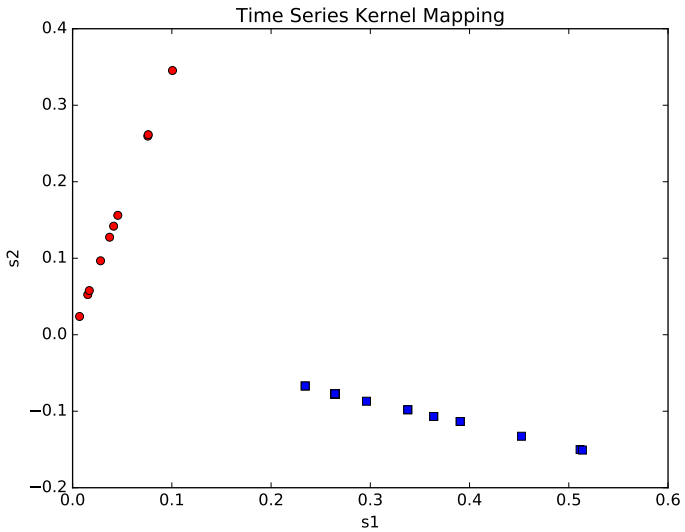


b) The evaluated distance kernel matrix.

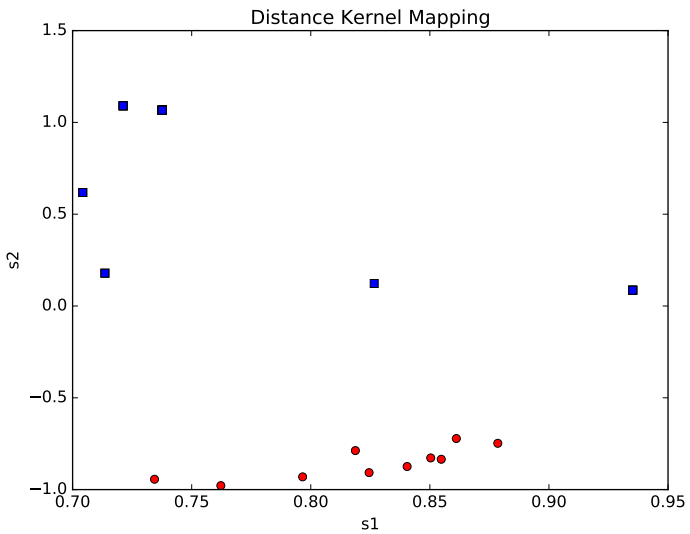


c) Both kernels result in the same spectral label matrix.

Figure 5.8. The difference between kernels in discriminating the malicious users. Note that the plots show only 25 of the 200 users for clarity purposes. In the bottom figure, the dark red cells correspond to the attacker pairs.



a) The time series alignment kernel mapping.



b) The distance kernel mapping.

Figure 5.9. The mapping of the two kernel matrices in the projected 2D space $(s1, s2)$ after spectral clustering. The blue squares are the attackers and the red circles are the innocent users for the level 4 intensity traffic with the observation interval of 1 second.

Table 5.4. The performance of different attacker identifiers. Constant and χ^2 stand for DCPMs run with constant and χ^2 thresholding, respectively. Distance, Time Series and Clustering represent the distance kernel, the time-series kernel and the time-series clustering, respectively, for the sampling interval of 1 second and level 4 traffic and where the DCPM parameters are the same with Table 5.1.

Model	Precision	Recall	F-Score
Constant - Distance	0.64 ± 0.13	0.5 ± 0.08	0.55 ± 0.1
Constant - Time Series	0.68 ± 0.1	0.51 ± 0.1	0.57 ± 0.1
Constant - Clustering	0.49 ± 0.12	0.37 ± 0.07	0.4 ± 0.08
χ^2 - Distance	0.72 ± 0.07	0.57 ± 0.04	0.62 ± 0.05
χ^2 - Time Series	0.77 ± 0.05	0.60 ± 0.06	0.66 ± 0.06
χ^2 - Clustering	0.5 ± 0.19	0.36 ± 0.13	0.4 ± 0.15

The average running times of the attacker identification methods are given in Table 5.5, where $\gamma = \rho = 1$ for the time series kernel, the order, which is the number of nearest neighbors to process during cluster member candidate selection, is set to 1 for time series clustering. The observation interval is taken as 1 second. The distance kernel does not have any parameters to be set. For each method, the number of clusters to be found is set to 2. This table shows that as the traffic rate goes high, the running times of the identification methods also increase. Note that the running time of the time series method is much higher than that of the other two models. The reason is that the computational load of pairwise similarities in the time-series kernel increases proportionally to the number of active users and the number of messages sent by the users.

Table 5.6 shows the run times of the models with respect to the observation interval. The longer the interval, the longer the models take to identify the objects. The running time of time series kernel increases exponentially since the kernel is evaluated by the pairwise similarity calculation of each messages sent by the subscribers. The longer the interval, the more messages the subscribers send. Also there are more

Table 5.5. The processing times of attacker identification methods in seconds, for the observation interval of 1 second with setting in Table 5.1.

Intensity	Distance Kernel	Time Series	Clustering
Level 1	0.11±0.07	19.81±26.46	0.91±1.03
Level 2	0.11±0.07	21.20±26.7	0.93±1.05
Level 3	0.12±0.07	22.02±28.5	0.94±1.12
Level 4	0.12±0.08	26.89±29.72	0.96±1.45
Level 5	0.12±0.08	29.4±37.21	1.03±1.94

Table 5.6. The processing times of attacker identification methods in seconds, for the different observation intervals.

Sampling	Distance Kernel	Time Series	Clustering
1 second	0.12±0.07	22.02±28.5	0.94±1.12
2 seconds	0.16±0.12	126.33±144.30	5.06±7.84
3 seconds	0.16±0.08	326.55±446.37	20.38±31.66
5 seconds	0.24±0.13	-	72.00±95.23
10 seconds	0.76±0.45	-	162.30±286.62

number of active subscribers. The running time of time series kernel is not evaluated for 5 and 10 seconds interval since it takes so much time.

5.2. Image and Video Retrieval

In order to assess the performance of the proposed discriminative tensor decomposition methods, we have tested our two proposed methods and compared them with 5 other tensor decomposition methods. For all methods we have run experiments on publicly available four popular image and one video data sets. Performance comparisons are obtained with competitor tensor decomposition/factorization methods in the current literature. The training samples in each data set consist of a randomly se-

lected subset of instances while the remaining ones being used for testing purposes. Experiments are repeated 20 fold for all data sets and performance results are averaged over the folds. The k-nn classifier and mean Average Precision (mAP) are used in all experiments.

5.2.1. Feature Extraction Algorithms

The features used for the competitor methods in the literature are detailed in the sequel. The baseline classifier uses simply raw tensor data, hence does not execute any feature extraction. As for the proposed method, recall that LMTD-C uses the core tensor for features. On the other hand, the LMTD-F algorithm uses the back-projected tensor from the core space, that is, the tensors reconstructed into their original dimensions. Overall seven different methods take place in the performance competition, and five of the seven tensor decomposition methods result in lower dimensional features.

k-nn (k-Nearest Neighbor): The k-nn classifier is applied directly to the input tensors without any preprocessing such as dimension reduction and subspace projection. This method forms the base classifier against which the improvements, if any, provided by the decomposition methods will be assessed.

LDA (Linear Discriminant Analysis): The LDA is applied to the vectorized input tensors. The vectorized tensors are reduced to $C-1$ dimensional feature vectors where C is the number of classes.

PCA (Principal Component Analysis): The PCA is applied to the vectorized input tensors to reduce their dimensions to $C-1$.

MDA (Multilinear Discriminant Analysis): An extended version of the linear discriminant analysis (LDA), proposed in [49], is applied to tensors. In this method a sequence of eigendecompositions are executed iteratively to find directions in each way to shrink the intraclass scatter while widening the interclass scatter, much as in the spirit of LDA. This algorithm consists of two loops: The outer loop determines the number

of times the set of eigendecomposition problems are solved to generate the projection matrices, e.g., until convergence, that is, until the change between two successive iterations become smaller then a margin. The inner loop solves the LDA problem in each way. In any n-th iteration of the inner loop, the tensors are first multiplied with all the projection matrices except $\mathbf{U}^{(n)}$, then LDA is solved over the n-mode fibers of core tensors to find $\mathbf{U}^{(*n)}$. Note that MDA does not guarantee convergence, thus a stopping criterion needs to be applied. Our experimental study shows however that, mostly, the algorithm converges within less than 20 iterations. Nevertheless a maximum number of 500 iterations is set to mitigate the cases of no convergence and the solution set which has the smallest total change of Frobenius distances for the projection matrices between two sequential iterations is assumed as the best solution.

TR1DA (Discriminant Tensor Rank-1 Decomposition): TR1DA [47], is a tensor-to-vector mapping method. This algorithm decomposes higher-order tensors into elementary multilinear projections (EMPs), which consist of a unit vector in each way. It finds a collection of discriminating EMPs to map an input tensor into a feature vector such that iteratively each EMP reduces the total reconstruction error. The set of P EMPs transforms the tensor into a P-dimensional feature vector. Similar to MDA, in any iteration of the inner loop, the error tensors, defined as the difference between the reconstructed tensors and the input tensors, are first multiplied with all the unit projection vectors except $\mathbf{u}_p^{(n)}$, then a LDA-like eigendecomposition problem is solved over the projected n-mode fibers to find $\mathbf{u}_p^{(*n)}$. As in the MDA method, the iterations continue until the convergence of the EMPs or the maximum number of iterations. In effect, TR1DA is a feature extractor for the solution of classification problems.

R-UMLDA (Regularized Uncorrelated Multilinear Discriminant Analysis): R-UMLDA [48] can be regarded as an updated version of TR1DA. Unlike TR1DA, in each outer iteration it focuses on finding uncorrelated EMPs so that ways the unit vectors in EMPs belonging to one of the N ways are orthonormal to each other. When a new EMP is derived from the error tensors, the unit vectora are added to the current solution set only after their direction are made uncorrelated to unit vectors in other EMPs.

Table 5.7. The details of data sets used in the experiments.

Data Set	Classes	Training	Test	Dimensions
FERET	80	400	845	$32 \times 32 \times 16$
ETH80	8	40	40	$32 \times 32 \times 41$
KTH Human Act.	6	30	2361	$32 \times 24 \times 24$
Cambridge Gest.	9	45	855	$32 \times 24 \times 32$

LMTD-F (Large Margin Tensor Decomposition - Full): LMTD-F focuses on minimizing the reconstruction error while trying to maximize the k-nn accuracy over the reconstructed instances. With appropriate weight assignments, LMTD-F can sway from a reconstruction optimization problem to a classification accuracy maximization problem.

LMTD-C (Large Margin Tensor Decomposition - Core): LMTD-C, a variation of the above LMTD-F algorithm but working with core tensors, pursues similar reconstruction error, k-nn accuracy trade-offs.

5.2.2. Data Sets

The above tensor decomposition methods are tested on a number of publicly available data sets whose details are given in their corresponding subsections. Whenever parameters had to be set, as in the case of, e.g., Gabor bank filtering for data preprocessing or deciding for a parameter value (e.g. resizing the tensors) we have selected values either to be compatible with the literature or according to the requirements of the algorithms.

The details of the data sets (except USF Gait) are summarized in Table 5.7 while those of the USF Gait data set is given in Table 5.8 shows. As mentioned before, it consists of 4 different data sets recorded with different probes. The one with the highest number of instances is used for the training, while the others are used for testing.

Table 5.8. The details of USF Gait data set.

Sets ($32 \times 22 \times 10$)	Training	Probe A	Probe B	Probe C
Classes / Size	71 / 731	71 / 727	41 / 423	42 / 420

Figure 5.10. An example gait from USF Gait data set.

USF Gait: USF (The University of South Florida) Gait challenge data set V1.7 contains 452 sequences from 74 individuals. The subjects are recorded walking in elliptical paths with two different viewpoints, with two different shoe types (A and B) and over two different surface types (grass and concrete) [69, 70]. A subset containing only over the grass recordings are used in our experiments. The training data consists of gaits filmed from the right viewpoint with shoe type A. The remaining gaits forming the testing sets are collected under three probes: Probe A (shoe type A, left viewpoint), B (shoe type B, right viewpoint) and C (shoe type B, left viewpoint). The original data set used in [70], which are the binary masked gait silhouette sequences already naturally in 3D, consist of $128 \times 88 \times 20$ tensors: 20 video frames of 128×88 pixels. We use the version resized to $32 \times 22 \times 10$ in [48], which is publicly available in the authors' webpage [71]. A tensor unfolded along the time index is shown in Figure 5.10 as an example.

KTH Human Action: KTH (Kungliga Tekniska Högskolan - KTH Royal Institute of Technology) human action database contains 600 videos [72]. There are six action classes: boxing, hand clapping, hand waving, jogging, running, and walking. Every class consists of 100 videos performed by 25 individuals under four different scenarios, which are outdoors, outdoors with scale variation, outdoors with different clothes, and indoors. The original video frames have 160×120 resolution and video sequences differ

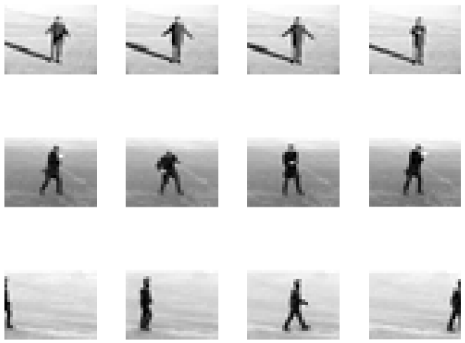


Figure 5.11. Some example frames from different actions in KTH data set.

in the number of frames. The actions are repeated multiple times by each subject under each condition yielding 2391 video sequences, providing almost 400 repeats for every action. The sequences are size normalized to 32×24 pixel frames and time normalized by picking 24 frames from the middle of sequences. Therefore, each action is packed into a tensor of size $32 \times 24 \times 24$ where the third dimension represents the time axis.

FERET: The FERET is a well-known face recognition database used in [73]. The original data set contains more than 14K images collected from 1199 people. In our study, we have used a subset of 1145 images belonging to 80 individuals. The selection has been made so that there are at least 10 images per person. We have 4 images per person in the training stage in each fold, and overall 320 images were used for training while the remaining 825 images were kept for testing purposes. The face images are aligned, cropped and normalized to 32×32 pixels with 256 gray levels per pixel, which is publicly available at [71]. We tensorize the image using a bank of Gabor filters, as in [49]. We use the orientations (θ) of $0, 45, 90$ and 135 degrees and wavelengths (λ) of $2, 4, 8$ and 16 pixels. An example image and its Gabor filtered versions are given in Figure 5.12.

ETH80: The multi-view image database ETH80 [74] includes object images from eight categories. There are ten objects from each category and every object is captured from

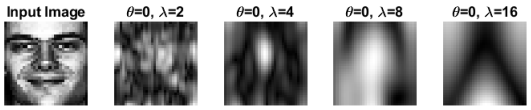


Figure 5.12. An image in the Feret database and its gabor filters with $\theta = 0$ degrees and $\lambda = \{2, 4, 8, 16\}$.

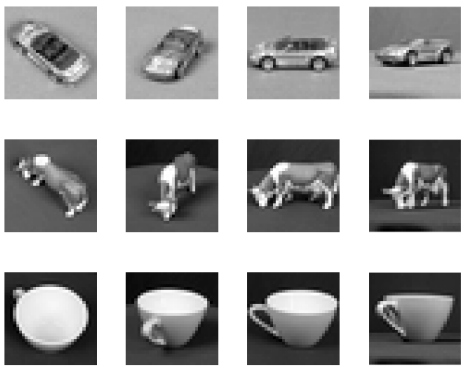


Figure 5.13. Some of the frontal slices from objects in ETH80 data set.

41 different viewpoints. The original images are of 128×128 pixels and for analysis purposes they are normalized to 32×32 size and 256 gray levels. Thus, each object is represented as a $32 \times 32 \times 41$ tensor, and we get 80 such kind of tensor objects per pose. In other words, the 3rd dimension of the tensor represents the view angle, while the other two correspond to pixel coordinates. In each fold, the data set is splitted into two halves, 5 tensors for training and 5 tensors for test for each object.

Cambridge Gestures: The Cambridge gesture database is made up of 900 varying length image sequences with nine different hand gesture types performed by two subjects [75], and recorded under five different illuminations. Frames have 320×240 pixels and each video clip consists of 100 image frames. The sequences are pre-processed and transformed into $32 \times 24 \times 32$ tensors with 256 grayscale levels. The sequences are length normalized by picking 32 consecutive frames from the middle of the sequences.

Figure 5.14. Some example frames from different actions in Cambridge Gestures data set.

The video clip database is thus reduced to 900 tensor objects where 40 tensors per class are used for training and the remaining 60 tensors for testing.

5.2.3. Experimental Setups

We address the scarce data problem in the experiments, that is retrieval and classification problems when there is a very limited quantity of training data. The retrieval to the query is based on the smallest Frobenius distance between matrices or between tensors depending on the data representation used. The performance is measured in terms of both classification (k-nn) accuracy, whether the query will be classified correctly within its nearest neighborhood, and mean average precision (mAP), whether the returned objects reflect the relevance in response to a query.

In all experiments, we assume that we have only 5 ($k = 5$) instances for training from each class. In each of the 20 folds, 5 instances are selected randomly and all the remaining database instances are used for testing. For each dataset and for each method we find the corresponding optimum parameters, which are yielding the highest accuracy result using a grid search. The maximum number of iterations is set to 500 for each one of iterative feature selection algorithms.

The early stopping threshold for TR1DA and R-UMLDA is set to $\epsilon = 0.01$. Thus these algorithms stop iterating if the Frobenius distance of each vector's update in every iteration falls below the threshold. The ζ parameters is searched for $\{1, 10, 100\}$ for TR1DA. For R-UMLDA, we search over $\gamma = \{0.001, 0.01, 0.1\}$. For LMTD-F and LMTD-C, during training the number of targets for each instance, which is also represented by k in the LMTD formulations, is 4, since it is the maximum possible number of neighbors in our experimental setup. We set $\beta = \gamma = 1$ and we select $\mu = \{0, 1/64, 1/8, 1\}$. The margin parameter, C, search set depends on the characteristics of the data set, roughly such as $C = \{\sum_{n=1}^{N=3} cI'_n, \sum_{n=1}^{N=3} cI_n, \prod_{n=1}^{N=3} cI'_n, \prod_{n=1}^{N=3} cI_n\}$ for different c values. If there is a draw in the class votes when a test instance is classified, it is assigned to the class of the closest tying up neighbor.

The tensor dimensions quoted in Tables 5.8 and 5.7 represent the raw data dimensions given as input to the feature extraction and classification algorithms listed in Section 5.2.1, as Feature Extraction Algorithms. Recall that in some methods as LDA, the tensor is reduced into a feature vector and in other methods as MDA, the tensor is reduced to a smaller sized feature tensor. The feature dimensions used in each case, as used in the algorithms irrespective of the input data size are listed in Table 5.9. The Tensor-to-Vector (T2V) group contains LDA, PCA, TR1DA and R-UMLDA methods while the Tensor-to-Tensor (T2T) group consists of MDA, LMTD-F and LMTD-C methods. Since TR1DA and R-UMLDA find EMPs and the tensor object is reduced to a coefficient vector of these EMPs, these methods take place in the T2V group. The feature dimensions listed Table 5.9 are determined by one of the following criteria: 1) #classes $-$ 1 (as in LDA in ETH80, and KTH Human Action); 2) The smallest dimension in a way (USF Gait); 3) A common divisor for all dimensions (Cambridge Gestures) or 4) A common scaling in all dimensions (0.25 in Ferets).

5.2.4. Performance Comparisons of the Methods

To have a fair comparison of the decomposition methods, the number of parameters to be estimated in the projection matrices must be taken into the account. Thus when the input tensor data is reduced into an d-dimensional feature, the num-

Table 5.9. The reduced dimensions for each data set.

Data Set	Tensor-to-Vector	Tensor-to-Tensor
FERET	8	$8 \times 8 \times 4$
ETH80	7	$7 \times 7 \times 7$
KTH Human Act.	5	$5 \times 5 \times 5$
Cambridge Gest.	8	$8 \times 8 \times 8$
USF Gait	10	$10 \times 10 \times 10$

ber of free parameters list as follows: for PCA and LDA, $d \cdot \prod_{n=1}^{N} I_n$ parameters; for TR1DA and R-UMLDA, $d \cdot (\sum_{n=1}^{N} I_n)$ parameters; for MDA, LMTD-F and LMTD-C, $d \cdot (\sum_{n=1}^{N} I_n)$ parameters when the latter two are reduced to $(d \times d \ldots \times d)$ N-way tensors. For example, for the KTH Human Action data set, PCA and LDA require $5 \times 32 \times \times 24 \times 24 = 92160$ parameters, while the others require $5 \times (32 + 24 + 24) = 400$.

Tables 5.10 and 5.11 show the 5-nn classification results for different feature extractors over the five datasets while Tables 5.12 and 5.13 list their mean average precision (mAP). The classification scores show how accurately a new tensor test instance is labeled with respect to its closest neighbors, while the mAP scores show how relevant/similar the returned instances are for a query. These performance figures and their standard deviations result from averaging the scores over 20 folds. Finally the simulations are run under the optimal parameters for each data set and each algorithm. The proposed algorithms outperform their six competitors TR1DA, R-UMLDA, k-NN, LDA, PCA, MDA over the FERET, ETH80, KTH and Cambridge data sets, but are slightly inferior in the case of USF dataset.

FERET: Both of the LMTD varieties outperform their competitors with LMTD-C slightly better in both classification accuracy and mAP score. They have the same best parameter set, $\mu = 1$ and $C = 2$. An interesting outcome for this data set is that though LDA has a much lower accuracy rate than LTMD methods, it has a mAP value close theirs. It means that LDA also maps the targets closer than their input

space versions, but it fails to map them close enough to improve the accuracy. For the TR1DA, ζ is 10 and for the R-UMLDA γ is 0.01.

ETH80: The LTMD-F and LMTD-C have the parameter of $\mu = 0.0$, $C = 32$ and $\mu = 1/8$, $C = 32$, respectively. For the TR1DA, ζ is 1 and for the R-UMLDA γ is 0.01. The LMTD-F has the highest accuracy and mAP.

KTH: In KTH data set, the methods almost perform the same, but LMTD-F and LMTD-C are only marginally better than TR1DA in terms of mAP and accuracy. The parameters of the LMTD methods are the same $\mu = 1/64$ and $C = 16$. For the TR1DA ζ is 1 and for the R-UMLDA γ is 0.01.

Cambridge Gestures: The LMTD-F and LMTD-C perform very close to each other both in terms of accuracy and mAP, but otherwise obtain higher scores than their competitors. Their optimal parameters are set to $\mu = 1/64$ and $C = 12$, while $\zeta = 10$ is for TR1DA and $\gamma = 0.001$ for R-UMLDA.

USF Gait: In USF Gait data set, the two LMTD methods surprisingly do not achieve the highest performance, though the performance gap with the nearest competitor is not major. R-UMLDA has the highest mAP scores for all the three test sets and also the highest accuracy on two of the test sets. The best parameter settings for LMTD methods are $\mu = 1/8$ and $C = 8$, while for the TR1DA ζ is 1 and for the R-UMLDA γ is 0.01.

5.2.5. Sensitivity Analysis over Feature Dimensionality

We investigate the sensitivity of the performance of the tensor decomposition to the feature dimension D. Fixing the training set size, i.e., the number of instances ($k = 5$) and using the best parameter settings for each algorithm, we have evaluated accuracy and mAP scores for different reduced dimensions D over the ETH80 dataset. Note that for LDA and PCA the dimensions vary from 1 to 7, and for the tensor decomposition methods the dimensions vary from $1 \times 1 \times 1$ to $7 \times 7 \times 7$, since 7 is the

Table 5.10. The classification accuracy over the data sets for $k = 5$.

	5-nn	LDA	PCA	MDA	TR1DA	R-UMLDA	LMTD-F	LMTD-C
FERET	35.65 ± 1.67	48.66 ± 1.78	14.69 ± 1.23	61.97 ± 2.27	37.54 ± 1.94	28.93 ± 2.64	64.09 ± 2.20	**66.07 ± 2.62**
ETH80	70.38 ± 4.49	73.25 ± 5.6	67.5 ± 4.03	82.5 ± 4.03	63.63 ± 7	79.63 ± 6.19	**84.13 ± 5.02**	83.88 ± 5.27
KTH	25.16 ± 1.92	29.23 ± 1.59	25.78 ± 1.67	19.55 ± 2.25	30.93 ± 1.75	30.54 ± 2.89	31.30 ± 2.16	**31.59 ± 1.82**
Cambridge	18.09 ± 2.07	25.44 ± 2.28	16.56 ± 1.6	19.49 ± 4.75	33.36 ± 4.11	30.46 ± 4.26	**34.58 ± 4.16**	34.55 ± 4.18

Table 5.11. The classification accuracy over USF Gait data set for $k = 5$.

	5-nn	LDA	PCA	MDA	TR1DA	R-UMLDA	LMTD-F	LMTD-C
Probe A	29.88 ± 1.6	27.27 ± 6.27	17.52 ± 1.45	**37.28 ± 2.98**	33.67 ± 2.54	35.89 ± 2.20	33.97 ± 2.25	34.88 ± 1.79
Probe B	20.34 ± 1.94	19.31 ± 4.44	11.94 ± 1.69	22.97 ± 2.29	24.50 ± 2.45	**30.04 ± 2.68**	23.85 ± 2.26	24.74 ± 2.33
Probe C	11.75 ± 1.63	13.48 ± 3.10	7.68 ± 1.42	15.23 ± 2.79	15.71 ± 2.24	**17.37 ± 2.62**	14.58 ± 1.74	15.29 ± 2.05

Table 5.12. The mAP scores over the data sets where each class has 5 instances.

	5-nn	LDA	PCA	MDA	TR1DA	R-UMLDA	LMTD-F	LMTD-C
FERET	0.22 ± 0.01	0.42 ± 0.02	0.10 ± 0.01	0.44 ± 0.01	0.28 ± 0.01	0.21 ± 0.02	0.45 ± 0.01	**0.47 ± 0.02**
ETH80	0.67 ± 0.02	0.67 ± 0.03	0.65 ± 0.02	0.81 ± 0.03	0.65 ± 0.02	0.78 ± 0.04	**0.84 ± 0.03**	0.83 ± 0.03
KTH	0.30 ± 0.01	0.35 ± 0.02	0.30 ± 0.01	0.33 ± 0.02	0.41 ± 0.02	0.39 ± 0.04	0.41 ± 0.01	**0.41 ± 0.01**
Cambridge	0.23 ± 0.01	0.28 ± 0.02	0.21 ± 0.01	0.28 ± 0.03	0.36 ± 0.04	0.33 ± 0.03	0.38 ± 0.04	**0.38 ± 0.04**

Table 5.13. The mAP scores over USF Gait data set where each class has 5 instances.

	5-nn	LDA	PCA	MDA	TR1DA	R-UMLDA	LMTD-F	LMTD-C
Probe A	0.22 ± 0.01	0.24 ± 0.05	0.15 ± 0.01	0.28 ± 0.02	0.28 ± 0.02	**0.30 ± 0.02**	0.26 ± 0.01	0.27 ± 0.01
Probe B	0.17 ± 0.01	0.19 ± 0.03	0.12 ± 0.01	0.19 ± 0.02	0.22 ± 0.02	**0.27 ± 0.02**	0.20 ± 0.01	0.21 ± 0.01
Probe C	0.12 ± 0.01	0.15 ± 0.02	0.09 ± 0.01	0.14 ± 0.01	0.16 ± 0.01	**0.18 ± 0.02**	0.14 ± 0.01	0.15 ± 0.01

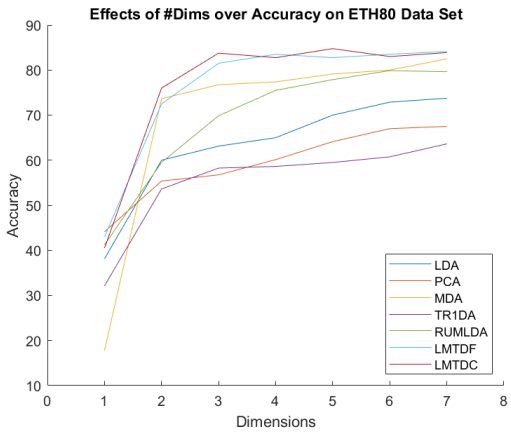


Figure 5.15. Effects of dimension number over accuracy on ETH80 data set.

maximum number of reduced dimensions for LDA.

Figure 5.15 shows the accuracy results with various feature dimensions. Note that both LTMD methods already get the highest accuracy results with only 2 dimensions. Almost all methods reach a saturation plateau beyond $D = 3$.

Figure 5.16 shows the mean average precision results over the dimensions. Interestingly, MDA has almost the same mAP scores with LMTD methods though its has lower accuracy. Similar to the accuracy plot, after $d = 3$, any increase in the dimensions does not affect the mAP scores significantly.

5.2.6. Sensitivity Analysis over Training Set Size

We investigate the effects of the number of instances provided per class over the performance scores in the KTH data set. We use the optimized parameter settings, the best ones obtained when feature dimensionality was set at $D = 5$, for each method.

Figure 5.17 shows that the accuracy is improving slowly with increasing number of instances. LDA seems to benefit the most with increasing number of instances. Figure 5.18 shows that mAP decreases with the increasing training set size. The performance

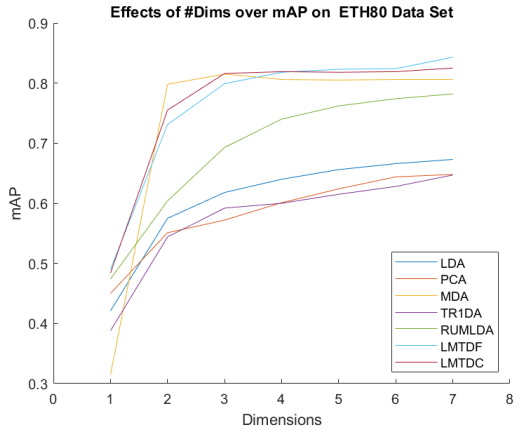


Figure 5.16. Effects of dimension number over mean average precision on ETH80 data set.

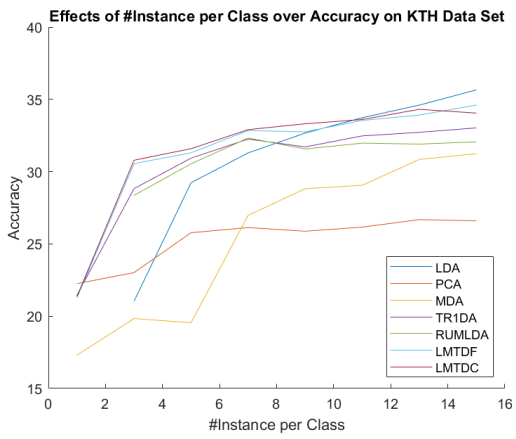


Figure 5.17. Effects of # Instances per Class over accuracy on KTH data set.

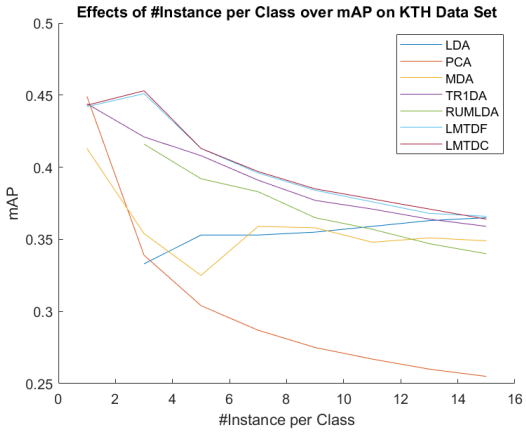


Figure 5.18. Effects of # Instances per Class over mean average precision on KTH data set.

of the methods seem to converge to the similar values.

6. CONCLUSION

In this concluding chapter, we summarize the contributions of this dissertation and give some leads for possible future directions. We present solutions for two main challenges addressed in this thesis: Attack detection and object retrieval. We compare them with some competitors in the literature. We also point out how to improve the proposed solutions.

As our first contribution, we have focused on the detection of DDoS attacks in SIP networks and on the identification of users coordinated in an attack. An adaptive cyber security monitor is developed consisting of two basic components: a change-point detector to alert the system of an ongoing attack and an identifier for the malicious user set.

The proposed change-point model tracks the Mahalanobis distance between the messaging counts in successive observation intervals. The rationale is that a marked (dis)similarity of sequential message count vectors can uncover abrupt changes in the traffic pattern. High dissimilarity instances, i.e., the Mahalanobis distance above a threshold, is labeled as a candidate attack. The threshold value is critical to differentiate DDoS attacks from random fluctuations of the traffic. The proposed DCPM is capable to adapt to the traffic variations due to the online estimation of the Mahalanobis metric and, consequently yields significantly better performance as compared to the literature results.

Identification of DDoS attackers is based on behavioral similarity in messaging sequences. Based on the premise that attackers act in a coordinated way while normal users show a much less structured messaging pattern, two corresponding clusters are conceived. The user-to-user similarity is measured by kernelizing their messaging time series. In the time-series kernel function we explicitly use the timestamps of the messaging events; in the distance kernel, we collapse the messaging activities within an observation interval into a cumulative count vector. The behavioral clusters are

extracted using normalized Laplacian spectral clustering.

The performance of the proposed system is tested over a simulated SIP network environment, which simulates transactions of ordinary subscribers and attackers. Depending on the intensity of the normal network traffic, observation interval and attack magnitude, our F-scores are more than 0.70 for the distance-based change point models, which is much higher than that of the ARIMA model.

The effects of observation window length, background traffic intensity and parameter settings for the proposed DCPM methods are discussed in detail. The longer observation windows result in more accurate attack detectors, but they come with a price of reduced onset/offset resolution. As one should expect, the intensity of background traffic has a diminishing effect on the performance of the proposed methods: The more fluctuations the traffic has, the lower the F-scores are, which are still higher than 0.70. Also the parameters of the models should be calibrated to account for the seasonal changes.

The attacker identification algorithms are also compared in detail. The time-series kernel has higher F-scores but also has a considerable running time. Longer observation intervals form longer time-series and the running times increase almost exponentially. Similarly, the higher intensity of traffic causes an increase in the running time. Even though the distance kernel has lower accuracy values, its running time is almost unaffected by the observation window interval or the traffic intensity. The reason for it is that each user is represented as a vector and the number of the operations are not affected by the window interval or the intensity.

The proposed solution can be advanced in several ways. First, in addition to observed message traffic, one can use additional data sources, such as SIP server log registers or its resource usage, e.g., CPU load. Second, the distance based change-point model compares only the last observation interval with the immediately preceding k frames to detect changes in the traffic. This can be extended to consider the most current m frames and the k frames in its past. We conjecture that the comparison of two

group-of-frames might diminish false alarms, that is, changes detected which are not DDoS attacks. Thirdly, though the time-series kernel has slightly higher performance, it takes longer time to respond due to the cost of kernel matrix computation. The distance kernel is faster but it does not benefit from the occurrence time information of the messages. An hybrid kernel might provide a more accurate detector than the distance kernel and a faster detector than the sequence alignment kernel. Finally, so far we have considered the cost of false negatives and false positives to be equal. From the point of view of operators that deploy SIP servers, these two costs are not equal and this should be taken into consideration in setting the threshold for attacker detection. Delayed response to a DDoS attack and suffering degradation of service quality should be weighted against taking preventive action toward subscribers that may not all be malicious users.

DDoS attacks may look deceptively simple, but they have proved to be hard to prevent, and they will be one of the major cyber security concerns with the spread of Internet-of-Things (IoT) devices. The capabilities of IoT devices and their security vulnerabilities (e.g. weak passwords or no protection mechanisms at all) make them easy victims as zombies for botnet applications, such as Mirai [76]. The botnets are also evolving and subsequently adapting against deployed counter-measures. Thus, more research should be carried over particular in detection of attack sources to overcome the possible outages and network congestion on the horizon with the wide spread of IoT devices.

As our second contribution, a novel tensor decomposition method with two versions is introduced. They use the Frobenius distances to the target neighbors to capture the local manifold structure, while training the global projection matrices. The nearest neighbor classification is improved: The targets are projected closer to the instance while its impostors are swept away. The reduced version focuses on higher accuracy in the feature space, while the full version aims improved accuracy in the input space over the reconstructed tensors.

We have investigated their performance with respect to five other non-tensor methods, i.e., methods that rely on the matricized or vectorized version of the tensor data. The method vies to improve the k-nn classification and/or retrieval performance by designing transformation matrices to warp the feature space such that, for all classes, in-class samples fall closer to each other while becoming more distinguishable from other-class samples. The proposed LMTD-Core version focuses on good classification accuracy, while reconstruction quality is indirectly guaranteed through core tensor-to-core tensor similarity. The second proposed algorithm, LMTD-Full, targets faithful reconstruction as well as good discrimination. In all work, tensor similarity is based on Frobenius distance.

The essence of our method is to explore the performance of the subspace methods while maintaining the tensor form in order not to incur the information loss, if any, when the tensor structure is discarded in favor of the matricized or vectorized versions. We conjecture that information loss incurred disregarding tensorial structure will have less impact in data rich problems; on the contrary, when the training data set is very limited, that is, for the scarce data problem, information in the tensorial structure will play a more prominent role. The experimental results have proved our conjecture in that the proposed methods perform with higher accuracy and higher mAP scores in four of the five datasets. The reason is that LMTD methods are able to capture the correlation between the projection matrices. Thus, LMTD can capture more information in the core tensors.

The effects of the feature dimensions and of the training set size on the performance have been investigated. The experiments show, in fact that while the two LTMD methods have higher performance scores for all feature dimensions, as the number of instances per class increases the performance of the methods seem to converge.

The proposed models contain a trade-off between the nearest neighbor accuracy and the reconstruction error. The coefficients in the loss function should be assigned with respect to the requirements of the application. In case minimizing the reconstruction error is significant, then μ parameter should be set to a high value. If the accuracy

is the main concern, then β and γ parameters should be set accordingly.

The future directions for LMTD include their extensions to tensor completion and use of different distance functions other than Frobenius norm distance, for example the angles between the fibers. An experimental study on the effects of the noise over tensor decomposition methods can be considered. As an improvement, an iterative version of the large margin tensor decomposition can be studied. The proposed methods find projection matrices in such a way that many projection vectors in the same mode are found simultaneously. As an alternative, an iterative and additive version can calculate new projection vectors in each mode, then append them to the existing projection matrices.

REFERENCES

1. Raza, N., I. Rashid and F. A. Awan, "Security and Management Framework for an Organization Operating in Cloud Environment", *Annals of Telecommunications*, Vol. 72, No. 5, pp. 325–333, June 2017.

2. Bolton, D., *Anonymous 'Declares War' on Turkey, Claims Responsibility for Recent Massive Cyberattacks*, 2015, `http://www.independent.co.uk/life-style/gadgets-and-tech/news/anonymous-declares-war-on-turkey-opsis-russia-cyberattack-erdogan-a6784026.html`, accessed at June 2019.

3. Gupta, B. B. and T. Akhtar, "A Survey on Smart Power Grid: Frameworks, Tools, Security Issues, and Solutions", *Annals of Telecommunications*, Vol. 72, No. 9, pp. 517–549, September 2017.

4. Rosenberg, J., H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley and E. Schooler, *SIP: Session Initiation Protocol*, RFC 3261, June, `https://www.ietf.org/rfc/rfc3261.txt`, accessed at June 2019.

5. Cooney, M., *IBM Warns of Rising VoIP Cyber-Attacks*, 2016, `http://www.networkworld.com/article/3146095/security/ibm-warns-of-rising-voip-cyber-attacks.html`, accessed at June 2019.

6. Wilson, C., *DDoS Attacks Targeting Traditional Telecom Systems*, 2012, `https://asert.arbornetworks.com/ddos-attacks-targeting-traditional-telecom-systems/`, accessed at June 2019.

7. Keromytis, A., "A Comprehensive Survey of Voice over IP Security Research", *IEEE Communications Surveys & Tutorials*, Vol. 14, No. 2, pp. 514–537, April 2012.

8. Goldberger, J., G. E. Hinton, S. T. Roweis and R. R. Salakhutdinov, "Neighbourhood Components Analysis", L. K. Saul, Y. Weiss and L. Bottou (Editors), *Advances in Neural Information Processing Systems 17*, pp. 513–520, MIT Press, 2005.

9. Torresani, L. and K.-c. Lee, "Large Margin Component Analysis", B. Schölkopf, J. C. Platt and T. Hoffman (Editors), *Advances in Neural Information Processing Systems 19*, pp. 1385–1392, MIT Press, 2007.

10. Bunte, K., P. Schneider, B. Hammer, F.-M. Schleif, T. Villmann and M. Biehl, "Limited Rank Matrix Learning, Discriminative Dimension Reduction and Visualization", *Neural Networks*, Vol. 26, pp. 159–173, 2012.

11. Xing, E. P., M. I. Jordan, S. J. Russell and A. Y. Ng, "Distance Metric Learning with Application to Clustering with Side-Information", S. Becker, S. Thrun and K. Obermayer (Editors), *Advances in Neural Information Processing Systems 15*, pp. 521–528, MIT Press, 2003.

12. Davis, J. V., B. Kulis, P. Jain, S. Sra and I. S. Dhillon, "Information-Theoretic Metric Learning", *Proceedings of the 24th International Conference on Machine Learning*, pp. 209–216, New York, NY, USA, 2007.

13. kyun Noh, Y., B. tak Zhang and D. D. Lee, "Generative Local Metric Learning for Nearest Neighbor Classification", J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel and A. Culotta (Editors), *Advances in Neural Information Processing Systems 23*, pp. 1822–1830, Curran Associates, Inc., 2010.

14. Liu, Y., Y. Liu and K. C. C. Chan, "Tensor Distance Based Multilinear Locality-Preserved Maximum Information Embedding", *IEEE Transactions on Neural Networks*, Vol. 21, No. 11, pp. 1848–1854, November 2010.

15. Kulis, B., *Metric Learning: A Survey*, Now, 2013.

16. Wang, F. and J. Sun, "Survey on Distance Metric Learning and Dimensionality Reduction in Data Mining", *Data Mining and Knowledge Discovery*, Vol. 29, No. 2, pp. 534–564, March 2015.

17. Sisalem, D., J. Kuthan and S. Ehlert, "Denial of Service Attacks Targeting a SIP VoIP Infrastructure: Attack Scenarios and Prevention Mechanisms", *IEEE Network*, Vol. 20, No. 5, pp. 26–31, October 2006.

18. Chen, E. and M. Itoh, "Scalable Detection of SIP Fuzzing Attacks", *Second International Conference on Emerging Security Information, Systems and Technologies, SECURWARE'08*, pp. 114–119, August 2008.

19. Ehlert, S., D. Geneiatakis and T. Magedanz, "Survey of Network Security Systems to Counter SIP-Based Denial-of-Service Attacks", *Computers & Security*, Vol. 29, No. 2, pp. 225–243, March 2010.

20. Chen, Z. and R. Duan, "The Formal Analyse of DoS Attack to SIP Based on the SIP Extended Finite State Machines", *2010 International Conference on Computational Intelligence and Software Engineering (CiSE)*, pp. 1–4, December 2010.

21. Vrakas, N. and C. Lambrinoudakis, "An Intrusion Detection and Prevention System for IMS and VoIP Services", *International Journal of Information Security*, Vol. 12, No. 3, pp. 201–217, January 2013.

22. Vijayasarathy, R., S. V. Raghavan and B. Ravindran, "A System Approach to Network Modeling for DDoS Detection Using a Naive Bayesian Classifier", *The Third International Conference on Communication Systems and Networks (COMSNETS)*, pp. 1–10, IEEE, February 2011.

23. Yildiz, C., T. Y. Ceritli, B. Kurt, B. Sankur and A. T. Cemgil, "Attack Detection in VOIP Networks Using Bayesian Multiple Change-Point Models", *24th Conference on Signal Processing and its Applications (SIU)*, pp. 1301–1304, May 2016.

24. Yildiz, C., M. Semerci, T. Y. Ceritli, B. Kurt, B. Sankur and A. T. Cemgil, "Change Point Detection for Monitoring SIP Networks", *European Conference on Networks and Communications (EuCNC2016)*, June 2016.

25. Nassar, M., R. State and O. Festor, "A Framework for Monitoring SIP Enterprise Networks", *4th International Conference on Network and System Security (NSS)*, pp. 1–8, September 2010.

26. Tsiatsikas, Z., D. Geneiatakis, G. Kambourakis and S. Gritzalis, "Realtime DDoS Detection in SIP Ecosystems: Machine Learning Tools of the Trade", *10th International Conference on Network and System Security, NSS 2016*, pp. 126–139, Springer International Publishing, Cham, September 2016.

27. Nezhad, S. M. T., M. Nazari and E. A. Gharavol, "A Novel DoS and DDoS Attacks Detection Algorithm Using ARIMA Time Series Model and Chaotic System in Computer Networks", *IEEE Communications Letters*, Vol. 20, No. 4, pp. 700–703, January 2016.

28. D'Alconzo, A., A. Coluccia and P. Romirer-Maierhofer, "Distribution-Based Anomaly Detection in 3G Mobile Networks: from Theory to Practice", *International Journal of Network Management*, Vol. 20, No. 5, pp. 245–269, August 2010.

29. D'Alconzo, A., A. Coluccia, F. Ricciato and P. Romirer-Maierhofer, "A Distribution-Based Approach to Anomaly Detection and Application to 3G Mobile Traffic", *IEEE Global Telecommunications Conference 2009, GLOBECOM 2009*, pp. 1–8, November 2009.

30. Anagnostopoulos, M., G. Kambourakis and S. Gritzalis, "New Facets of Mobile Botnet: Architecture and Evaluation", *International Journal of Information Security*, Vol. 15, No. 5, pp. 455–473, October 2016.

31. Kirubavathi, G. and R. Anitha, "Structural Analysis and Detection of Android

Botnets Using Machine Learning Techniques", *International Journal of Information Security*, Vol. 17, No. 2, pp. 153–167, April 2018.

32. Silva, S. S., R. M. Silva, R. C. Pinto and R. M. Salles, "Botnets: A Survey", *Computer Networks*, Vol. 57, No. 2, pp. 378–403, February 2013.

33. García-Teodoro, P., J. Díaz-Verdejo, G. Maciá-Fernández and E. Vázquez, "Anomaly-Based Network Intrusion Detection: Techniques, Systems and Challenges", *Computers & Security*, Vol. 28, No. 1–2, pp. 18–28, February-March 2009.

34. Gupta, M., J. Gao, C. C. Aggarwal and J. Han, "Outlier Detection for Temporal Data: A Survey", *IEEE Transactions on Knowledge and Data Engineering*, Vol. 26, No. 9, pp. 2250–2267, September 2014.

35. Hyndman, R. J., E. Wang and N. Laptev, "Large-Scale Unusual Time Series Detection", *IEEE International Conference on Data Mining Workshop, ICDMW 2015*, pp. 1616–1619, November 2015.

36. Cuturi, M., "Fast Global Alignment Kernels", *Proceedings of the 28th International Conference on Machine Learning, ICML 2011*, pp. 929–936, June 2011.

37. Sivaramakrishnan, K. R., K. Karthik and C. Bhattacharyya, "Kernels for Large Margin Time-Series Classification", *International Joint Conference on Neural Networks, IJCNN 2007*, pp. 2746–2751, August 2007.

38. Chen, H., F. Tang, P. Tino and X. Yao, "Model-Based Kernel for Efficient Time Series Analysis", *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD'13, pp. 392–400, ACM, New York, NY, USA, August 2013.

39. Zhang, X., J. Liu, Y. Du and T. Lv, "A Novel Clustering Method on Time Series Data", *Expert Systems with Applications*, Vol. 38, No. 9, pp. 11891–11900, September 2011.

40. Oates, T., L. Firoiu and P. Cohen, "Clustering Time Series with Hidden Markov Models and Dynamic Time Warping", *Proceedings of the IJCAI-99 Workshop on Neural, Symbolic, and Reinforcement Learning Methods for Sequence Learning*, 1999.

41. Xiong, Y. and D.-Y. Yeung, "Mixtures of ARMA Models for Model-Based Time Series Clustering", *Proceedings of the IEEE International Conference on Data Mining*, March 2002.

42. Behal, S. and K. Kumar, "Detection of DDoS Attacks and Flash Events Using Novel Information Theory Metrics", *Computer Networks*, Vol. 116, pp. 96–110, April 2017.

43. Tellenbach, B., M. Burkhart, D. Schatzmann, D. Gugelmann and D. Sornette, "Accurate Network Anomaly Classification with Generalized Entropy Metrics", *Computer Networks*, Vol. 55, No. 15, pp. 3485–3502, October 2011.

44. Heo, J., E. Y. Chen, T. Kusumoto and M. Itoh, "Statistical SIP Traffic Modeling and Analysis System", *10th International Symposium on Communications and Information Technologies*, pp. 1223–1228, October 2010.

45. D'Antonio, S., M. Esposito, F. Oliviero, S. P. Romano and D. Salvi, "Behavioral Network Engineering: Making Intrusion Detection Become Autonomic", *Annales Des Télécommunications*, Vol. 61, No. 9, pp. 1136–1148, October 2006.

46. Xie, Q., Q. Zhao, D. Meng and Z. Xu, "Kronecker-Basis-Representation Based Tensor Sparsity and Its Applications to Tensor Recovery", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 40, No. 8, pp. 1888–1902, Aug 2018.

47. Wang, Y. and S. Gong, "Tensor Discriminant Analysis for View-based Object Recognition", *18th International Conference on Pattern Recognition, ICPR'06*, Vol. 3, pp. 33–36, 2006.

48. Lu, H., K. N. Plataniotis and A. N. Venetsanopoulos, "Uncorrelated Multilinear Discriminant Analysis With Regularization and Aggregation for Tensor Object Recognition", *IEEE Transactions on Neural Networks*, Vol. 20, No. 1, pp. 103–123, January 2009.

49. Yan, S., D. Xu, Q. Yang, L. Zhang, X. Tang and H.-J. Zhang, "Multilinear Discriminant Analysis for Face Recognition", *IEEE Transactions on Image Processing*, Vol. 16, No. 1, pp. 212–220, January 2007.

50. Li, X., M. K. Ng, G. Cong, Y. Ye and Q. Wu, "MR-NTD: Manifold Regularization Nonnegative Tucker Decomposition for Tensor Data Dimension Reduction and Representation", *IEEE Transactions on Neural Networks and Learning Systems*, Vol. 28, No. 8, pp. 1787–1800, August 2017.

51. Han, X.-H., Y.-W. Chen and X. Ruan, "Multilinear Supervised Neighborhood Embedding of a Local Descriptor Tensor for Scene/Object Recognition", *IEEE Transactions on Image Processing*, Vol. 21, No. 3, pp. 1314–1326, March 2012.

52. Ouamane, A., A. Chouchane, E. Boutellaa, M. Belahcene, S. Bourennane and A. Hadid, "Efficient Tensor-Based 2D+3D Face Verification", *IEEE Transactions on Information Forensics and Security*, Vol. 12, No. 11, pp. 2751–2762, November 2017.

53. Tao, D., X. Li, X. Wu and S. J. Maybank, "General Tensor Discriminant Analysis and Gabor Features for Gait Recognition", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 29, No. 10, pp. 1700–1715, October 2007.

54. He, X., D. Cai and P. Niyogi, "Tensor Subspace Analysis", Y. Weiss, B. Schölkopf and J. C. Platt (Editors), *Advances in Neural Information Processing Systems 18*, pp. 499–506, MIT Press, 2006.

55. Fu, Y., J. Gao, D. Tien, Z. Lin and X. Hong, "Tensor LRR and Sparse Coding-Based Subspace Clustering", *IEEE Transactions on Neural Networks and Learning*

Systems, Vol. 27, No. 10, pp. 2120–2133, October 2016.

56. Tomioka, R. and T. Suzuki, "Convex Tensor Decomposition via Structured Schatten Norm Regularization", C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani and K. Q. Weinberger (Editors), *Advances in Neural Information Processing Systems 26*, pp. 1331–1339, Curran Associates, Inc., 2013.

57. Kolda, T. G. and B. W. Bader, "Tensor Decompositions and Applications", *SIAM Review*, Vol. 51, No. 3, pp. 455–500, August 2009.

58. Lu, H., K. N. Plataniotis and A. N. Venetsanopoulos, "A Survey of Multilinear Subspace Learning for Tensor Data", *Pattern Recognition*, Vol. 44, No. 7, pp. 1540–1551, 2011.

59. Rabanser, S., O. Shchur and S. Günnemann, "Introduction to Tensor Decompositions and their Applications in Machine Learning", *ArXiv e-prints*, November 2017.

60. Sidiropoulos, N. D., L. D. Lathauwer, X. Fu, K. Huang, E. E. Papalexakis and C. Faloutsos, "Tensor Decomposition for Signal Processing and Machine Learning", *IEEE Transactions on Signal Processing*, Vol. 65, No. 13, pp. 3551–3582, July 2017.

61. Weinberger, K. Q. and L. K. Saul, "Distance Metric Learning for Large Margin Nearest Neighbor Classification", *Journal of Machine Learning Research*, Vol. 10, pp. 207–244, February 2009.

62. Cuturi, M., J. P. Vert, O. Birkenes and T. Matsui, "A Kernel For Time Series Based On Global Alignment", *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing 2007, ICASSP'07*, Vol. 2, pp. 413–416, April 2007.

63. Luxburg, U., "A Tutorial on Spectral Clustering", *Statistics and Computing*, Vol. 17, No. 4, pp. 395–416, August 2007.

64. Fonality, *Trixbox Business Phone Solutions*, 2016, `https://www.netfortris.com/trixbox`, accessed at June 2019.

65. Kurt, B., C. Yildiz, T. Y. Ceritli, M. Yamac, M. Semerci, B. Sankur and A. T. Cemgil, "A Probabilistic SIP Network Simulation System", *24th Conference on Signal Processing and its Applications (SIU)*, pp. 1049–1052, IEEE, June 2016.

66. Yildiz, C., B. Kurt, T. Y. Ceritli, A. T. Cemgil and B. Sankur, *BOUN-SIM API Reference*, Tech. rep., Department of Computer Engineering, Bogazici University, December 2016, `https://github.com/cagatayyildiz/boun-sim/`, accessed at June 2019.

67. Teluu, *PJSIP*, 2005, `http://www.pjsip.org/`, accessed at June 2019.

68. NETAS, *Nova V-SPY*, 2016, `http://novacybersecurity.com/products/nova_vspy`, accessed at June 2019.

69. Phillips, P. J., S. Sarkar, I. Robledo, P. Grother and K. Bowyer, "The Gait Identification Challenge Problem: Data Sets and Baseline Algorithm", *Object Recognition Supported by User Interaction for Service Robots*, Vol. 1, pp. 385–388, August 2002.

70. Lu, H., K. N. Plataniotis and A. N. Venetsanopoulos, "MPCA: Multilinear Principal Component Analysis of Tensor Objects", *IEEE Transactions on Neural Networks*, Vol. 19, No. 1, pp. 18–39, January 2008.

71. Lu, H., *Code & Data*, 2019, `http://www.dsp.utoronto.ca/~haiping/index.php?page=code`, accessed at June 2019.

72. Schuldt, C., I. Laptev and B. Caputo, "Recognizing Human Actions: A Local SVM Approach", *Proceedings of the Pattern Recognition, 17th International Conference on (ICPR'04)*, Vol. 3, pp. 32–36, IEEE Computer Society, Washington, DC, USA, 2004.

73. Phillips, P. J., H. Moon, S. A. Rizvi and P. J. Rauss, "The FERET Evaluation Methodology for Face-Recognition Algorithms", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, No. 10, pp. 1090–1104, October 2000.

74. Leibe, B. and B. Schiele, "Analyzing appearance and contour based methods for object categorization", *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. II–409, June 2003.

75. Kim, T.-K. and R. Cipolla, "Canonical Correlation Analysis of Video Volume Tensors for Action Categorization and Detection", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 8, pp. 1415–1428, August 2009.

76. Gamblin, J., *Mirai Source Code*, 2016, `https://github.com/jgamblin/Mirai-Source-Code`, accessed at June 2019.

APPENDIX A: DERIVATIONS OF LOGDET AND LMTD GRADIENT

A.1. Derivation of LogDet

The Kullback-Leibler (KL) divergence from distribution Q to distribution P, where $p(\mathbf{x})$ and $q(\mathbf{x})$ are their respective probability density functions, and $\mathbf{x} \in \mathbb{R}^d$, is calculated as:

$$D_{KL}(P \parallel Q) = \int p(\mathbf{x}) \log(\frac{p(\mathbf{x})}{q(\mathbf{x})}) d\mathbf{x} \tag{A.1}$$

$$= E_P\Big[\log(\frac{P}{Q}) \Big] \tag{A.2}$$

Assuming that both p and q are multivariate Gaussian distributions with mean vectors μ_p and μ_q and covariance matrices $\mathbf{\Sigma_p}$ and $\mathbf{\Sigma_q}$, respectively, one has:

$$p(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \det(\mathbf{\Sigma_p})^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{p})^\top \mathbf{\Sigma_p}^{-1}(\mathbf{x}-\mu_\mathbf{p}) \Big) \tag{A.3}$$

$$q(\mathbf{x}) = \frac{1}{(2\pi)^{d/2} \det(\mathbf{\Sigma_q})^{1/2}} \exp\Big(-\frac{1}{2}(\mathbf{x}-\mu_\mathbf{q})^\top \mathbf{\Sigma_q}^{-1}(\mathbf{x}-\mu_\mathbf{q}) \Big) \tag{A.4}$$

Using definitions A.3 and A.4 in Equation A.1, one obtains:

$$D_{KL}(P \parallel Q) = E_P\big[\log P - \log Q \big] \tag{A.5}$$

$$= \frac{1}{2} E_P\big[-\log\det \mathbf{\Sigma_p} - (\mathbf{x}-\mu_\mathbf{p})^\top \mathbf{\Sigma_p}^{-1}(\mathbf{x}-\mu_\mathbf{p}) + \log\det \mathbf{\Sigma_q} + (\mathbf{x}-\mu_\mathbf{q})^\top \mathbf{\Sigma_q}^{-1}(\mathbf{x}-\mu_\mathbf{q}) \big] \tag{A.6}$$

$$= \frac{1}{2}\log\frac{\det \mathbf{\Sigma_q}}{\det \mathbf{\Sigma_p}} + \frac{1}{2} E_P\big[-(\mathbf{x}-\mu_\mathbf{p})^\top \mathbf{\Sigma_p}^{-1}(\mathbf{x}-\mu_\mathbf{p}) + (\mathbf{x}-\mu_\mathbf{q})^\top \mathbf{\Sigma_q}^{-1}(\mathbf{x}-\mu_\mathbf{q}) \big] \tag{A.7}$$

$$= \frac{1}{2}\log\frac{\det \mathbf{\Sigma_q}}{\det \mathbf{\Sigma_p}} + \frac{1}{2} E_P\big[-\mathrm{tr}(\mathbf{\Sigma_p}^{-1}(\mathbf{x}-\mu_\mathbf{p})(\mathbf{x}-\mu_\mathbf{p})^\top) + \mathrm{tr}(\mathbf{\Sigma_q}^{-1}(\mathbf{x}-\mu_\mathbf{q})(\mathbf{x}-\mu_\mathbf{q})^\top) \big] \tag{A.8}$$

Here we use the identity $\mathbf{a}^\top \mathbf{B}\mathbf{c} = \mathrm{tr}(\mathbf{B}\mathbf{c}\mathbf{a}^\top)$ for any $\mathbf{a}, \mathbf{c} \in \mathbb{R}^d$ and $\mathbf{B} \in \mathbb{R}^{d\times d}$. Since both the trace and integration are linear operators, we can proceed as follows:

$$\begin{aligned} D_{KL}(P \parallel Q) &= \frac{1}{2}\log\frac{\det \boldsymbol{\Sigma_q}}{\det \boldsymbol{\Sigma_p}} - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_p}^{-1} E_P\big[(\mathbf{x}-\mu_\mathbf{p})(\mathbf{x}-\mu_\mathbf{p})^\top\big]) \\ &\quad + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_q}^{-1} E_P\big[(\mathbf{x}-\mu_\mathbf{q})(\mathbf{x}-\mu_\mathbf{q})^\top\big]) \qquad \text{(A.9)} \\ &= \frac{1}{2}\log\frac{\det \boldsymbol{\Sigma_q}}{\det \boldsymbol{\Sigma_p}} - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_p}^{-1}\boldsymbol{\Sigma_p}) \\ &\quad + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_q}^{-1} E_P\big[\mathbf{x}\mathbf{x}^\top - \mu_\mathbf{q}\mathbf{x}^\top - \mathbf{x}\mu_\mathbf{q}^\top + \mu_\mathbf{q}\mu_\mathbf{q}^\top\big]) \qquad \text{(A.10)} \\ &= -\frac{1}{2}\log\frac{\det \boldsymbol{\Sigma_p}}{\det \boldsymbol{\Sigma_q}} - \frac{1}{2}\mathrm{tr}(\mathbf{I}) \\ &\quad + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_q}^{-1} E_P\big[(\mathbf{x}\mathbf{x}^\top - \mu_\mathbf{q}\mathbf{x}^\top - \mathbf{x}\mu_\mathbf{q}^\top + \mu_\mathbf{q}\mu_\mathbf{q}^\top\big]) \qquad \text{(A.11)} \end{aligned}$$

Note that one has, $\frac{\det \boldsymbol{\Sigma_p}}{\det \boldsymbol{\Sigma_q}} = \det \boldsymbol{\Sigma_p}\boldsymbol{\Sigma_q}^{-1}$, $\mathrm{tr}(\mathbf{I}) = d$ and $E_P\big[\mathbf{x}\mathbf{x}^\top\big] = \boldsymbol{\Sigma_p} + \mu_\mathbf{p}\mu_\mathbf{p}^\top$. The Kullback-Leibler divergence becomes:

$$\begin{aligned} D_{KL}(P \parallel Q) &= -\frac{1}{2}\log\det \boldsymbol{\Sigma_p}\boldsymbol{\Sigma_q}^{-1} - \frac{1}{2}d \\ &\quad + \frac{1}{2}\mathrm{tr}\Big(\boldsymbol{\Sigma_q}^{-1}(\boldsymbol{\Sigma_p} + \mu_\mathbf{p}\mu_\mathbf{p}^\top - \mu_\mathbf{q}\mu_\mathbf{p}^\top - \mu_\mathbf{p}\mu_\mathbf{q}^\top + \mu_\mathbf{q}\mu_\mathbf{q}^\top)\Big) \qquad \text{(A.12)} \\ &= -\frac{1}{2}\log\det \boldsymbol{\Sigma_p}\boldsymbol{\Sigma_q}^{-1} - \frac{1}{2}d + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_q}^{-1}\boldsymbol{\Sigma_p}) \\ &\quad + \frac{1}{2}\mathrm{tr}\Big(\boldsymbol{\Sigma_q}^{-1}(\mu_\mathbf{p}\mu_\mathbf{p}^\top - \mu_\mathbf{q}\mu_\mathbf{p}^\top - \mu_\mathbf{p}\mu_\mathbf{q}^\top + \mu_\mathbf{q}\mu_\mathbf{q}^\top)\Big) \qquad \text{(A.13)} \\ &= -\frac{1}{2}\log\det \boldsymbol{\Sigma_p}\boldsymbol{\Sigma_q}^{-1} - \frac{1}{2}d + \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma_q}^{-1}\boldsymbol{\Sigma_p}) \\ &\quad + \frac{1}{2}(\mu_\mathbf{p} - \mu_\mathbf{q})^\top \boldsymbol{\Sigma_q}^{-1}(\mu_\mathbf{p} - \mu_\mathbf{q}) \qquad \text{(A.14)} \end{aligned}$$

Now, let's assume that the means vectors of p and of q are identical, $\mu_{\mathbf{p}} = \mu_{\mathbf{q}}$, then we can conclude that:

$$
\begin{aligned}
D_{KL}(P \parallel Q) &= \frac{1}{2}\Big(\operatorname{tr}({\boldsymbol{\Sigma}_{\mathbf{q}}}^{-1}\boldsymbol{\Sigma}_{\mathbf{p}}) - \log\det \boldsymbol{\Sigma}_{\mathbf{p}}{\boldsymbol{\Sigma}_{\mathbf{q}}}^{-1} - d\Big) && \text{(A.15)}\\
&= \frac{1}{2}\Big(\operatorname{tr}(\boldsymbol{\Sigma}_{\mathbf{p}}{\boldsymbol{\Sigma}_{\mathbf{q}}}^{-1}) - \log\det \boldsymbol{\Sigma}_{\mathbf{p}}{\boldsymbol{\Sigma}_{\mathbf{q}}}^{-1} - d\Big) && \text{(A.16)}\\
&= \frac{1}{2}D_{ld}(\boldsymbol{\Sigma}_{\mathbf{p}}, \boldsymbol{\Sigma}_{\mathbf{q}}) && \text{(A.17)}
\end{aligned}
$$

A.2. Metric Functions

A metric function $D : S \times S \to [0, \infty)$ defined on a set X must hold three basic conditions, as below, for any $a, b, c \in S$:

(i) $D(a, b) \geq 0$ and $D(a, b) = 0$ iff $a = b$ (non-negativity and positive-definiteness)
(ii) $D(a, b) = D(b, a)$ (symmetry)
(iii) $D(a, c) \leq D(a, b) + D(b, c)$ (triangle inequality)

$D_{ld}(\boldsymbol{\Sigma}_{\mathbf{p}}, \boldsymbol{\Sigma}_{\mathbf{q}})$ is a pseudo-metric defined over positive-definite matrices, $\boldsymbol{\Sigma}_{\mathbf{p}}, \boldsymbol{\Sigma}_{\mathbf{q}} \in \mathbb{R}^{d\times d}$, since it only guarantees non-negativity, and the other two rules do not necessarily hold true.

A.3. Derivation of Gradients for LMTDs

Let's define the following shorthand notations: $\mathbf{T}^{(n)} = \mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top$ and $\mathbf{V}_{(ij)}^{(-n)} = \mathbf{W}_{(i)}^{(-n)}(\mathbf{W}_{(j)}^{(-n)})^\top$. Note that $\mathbf{T}^{(n)} = (\mathbf{T}^{(n)})^\top$ and $\mathbf{V}_{(ii)}^{(-n)} = (\mathbf{V}_{(ii)}^{(-n)})^\top$ since they are symmetric.

The loss function for LMTD-F is defined as follows:

$$
L^{(F)} = \mu L_1 + \beta L_2^{(F)} + \gamma L_3^{(F)}
$$

The first term, L_1, is responsible for the reconstruction error.

$$\begin{aligned} L_1 &= \sum_{i=1}^{M} \mathrm{tr}\Big((\mathbf{X}_{(i)}^{(n)} - \hat{\mathbf{X}}_{(i)}^{(n)})^\top (\mathbf{X}_{(i)}^{(n)} - \hat{\mathbf{X}}_{(i)}^{(n)})\Big) \\ &= \sum_{i=1}^{M} \mathrm{tr}\Big((\mathbf{X}_{(i)}^{(n)})^\top \mathbf{X}_{(i)}^{(n)}\Big) - \mathrm{tr}\Big((\mathbf{X}_{(i)}^{(n)})^\top \hat{\mathbf{X}}_{(i)}^{(n)}\Big) - \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^\top \mathbf{X}_{(i)}^{(n)}\Big) + \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^\top \hat{\mathbf{X}}_{(i)}^{(n)}\Big) \end{aligned} \tag{A.18}$$

Using the approximation in Equation 4.15, we rewrite the total reconstruction error and take its derivative with respect to any n-mode projection matrix, $\mathbf{U}^{(n)}$, as below:

$$\begin{aligned} \frac{\partial L_1}{\partial \mathbf{U}^{(n)}} &= \sum_{i=1}^{M} -\frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big((\mathbf{X}_{(i)}^{(n)})^\top \hat{\mathbf{X}}_{(i)}^{(n)}\Big) - \\ &\quad \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^\top \mathbf{X}_{(i)}^{(n)}\Big) + \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^\top \hat{\mathbf{X}}_{(i)}^{(n)}\Big) \end{aligned} \tag{A.19}$$

$$\begin{aligned} \frac{\partial L_1}{\partial \mathbf{U}^{(n)}} &= \sum_{i=1}^{M} -\frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big((\mathbf{X}_{(i)}^{(n)})^\top \mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)}\Big) - \\ &\quad \sum_{i=1}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big((\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)})^\top \mathbf{X}_{(i)}^{(n)}\Big) + \\ &\quad \sum_{i=1}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big((\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)})^\top (\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top) \mathbf{W}_{(i)}^{(-n)}\Big) \end{aligned} \tag{A.20}$$

$$\begin{aligned} \frac{\partial L_1}{\partial \mathbf{U}^{(n)}} &= \sum_{i=1}^{M} -\frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big(\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)} (\mathbf{X}_{(i)}^{(n)})^\top\Big) - \\ &\quad \sum_{i=1}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big(\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{X}_{(i)}^{(n)} (\mathbf{W}_{(i)}^{(-n)})^\top\Big) + \\ &\quad \sum_{i=1}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big(\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)} (\mathbf{W}_{(i)}^{(-n)})^\top\Big) \end{aligned} \tag{A.21}$$

$$\begin{aligned}\frac{\partial L_1}{\partial \mathbf{U}^{(n)}} &= -\sum_{i=1}^{M} \mathbf{W}_{(i)}^{(-n)} (\mathbf{X}_{(i)}^{(n)})^\top \mathbf{U}^{(n)} - \sum_{i=1}^{M} \mathbf{X}_{(i)}^{(n)} (\mathbf{W}_{(i)}^{(-n)})^\top \mathbf{U}^{(n)} \\ &\quad - \sum_{i=1}^{M} \mathbf{X}_{(i)}^{(n)} (\mathbf{W}_{(i)}^{(-n)})^\top \mathbf{U}^{(n)} - \sum_{i=1}^{M} \mathbf{W}_{(i)}^{(-n)} (\mathbf{X}_{(i)}^{(n)})^\top \mathbf{U}^{(n)} \\ &\quad + \sum_{i=1}^{M} 2\Big(\mathbf{V}_{(ii)}^{(-n)} \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \mathbf{V}_{(ii)}^{(-n)}\Big) \mathbf{U}^{(n)} \end{aligned} \tag{A.22}$$

$$\begin{aligned}\frac{\partial L_1}{\partial \mathbf{U}^{(n)}} &= 2 \sum_{i=1}^{M} \Big(- \mathbf{W}_{(i)}^{(-n)} (\mathbf{X}_{(i)}^{(n)})^\top - \mathbf{X}_{(i)}^{(n)} (\mathbf{W}_{(i)}^{(-n)})^\top \\ &\quad + \mathbf{V}_{(ii)}^{(-n)} \mathbf{T}^{(n)} + \mathbf{T}^{(n)} \mathbf{V}_{(ii)}^{(-n)}\Big) \mathbf{U}^{(n)} \end{aligned} \tag{A.23}$$

The second term, $L_2^{(F)}$, represents the total distance between each instance and its targets.

$$\begin{aligned} L_2^{(F)} &= \sum_{i=1, j \rightsquigarrow i}^{M} \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)} - \hat{\mathbf{X}}_{(j)}^{(n)})^\top (\hat{\mathbf{X}}_{(i)}^{(n)} - \hat{\mathbf{X}}_{(j)}^{(n)})\Big) \\ &= \sum_{i=1, j \rightsquigarrow i}^{M} \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^\top \hat{\mathbf{X}}_{(i)}^{(n)}\Big) - \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^\top \hat{\mathbf{X}}_{(j)}^{(n)}\Big) \\ &\quad - \mathrm{tr}\Big((\hat{\mathbf{X}}_{(j)}^{(n)})^\top \hat{\mathbf{X}}_{(i)}^{(n)}\Big) + \mathrm{tr}\Big((\hat{\mathbf{X}}_{(j)}^{(n)})^\top \hat{\mathbf{X}}_{(j)}^{(n)}\Big) \end{aligned} \tag{A.24}$$

Using the approximations and shorthand notations, we can calculate its gradient for any $\mathbf{U}^{(n)}$.

$$\begin{aligned}\frac{\partial L_2^{(F)}}{\partial \mathbf{U}^{(n)}} &= \frac{\partial}{\partial \mathbf{U}^{(n)}} \\ &\Bigg(\sum_{i=1,j\rightsquigarrow i}^{M} \mathrm{tr}\Big((\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)})^\top (\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top) \mathbf{W}_{(i)}^{(-n)} \Big) \\ &- \sum_{i=1,j\rightsquigarrow i}^{M} \mathrm{tr}\Big((\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \mathbf{W}_{(i)}^{(-n)})^\top (\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top) \mathbf{W}_{(j)}^{(-n)} \Big) \\ &- \sum_{i=1,j\rightsquigarrow i}^{M} \mathrm{tr}\Big((\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \mathbf{W}_{(j)}^{(-n)})^\top (\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top) \mathbf{W}_{(i)}^{(-n)} \Big) \\ &+ \sum_{i=1,j\rightsquigarrow i}^{M} \mathrm{tr}\Big((\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \mathbf{W}_{(j)}^{(-n)})^\top (\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top) \mathbf{W}_{(j)}^{(-n)} \Big) \Bigg) \end{aligned} \tag{A.25}$$

$$\begin{aligned}\frac{\partial L_2^{(F)}}{\partial \mathbf{U}^{(n)}} &= \sum_{i=1,j\rightsquigarrow i}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Bigg(\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \\ &\Big(\mathbf{W}_{(i)}^{(-n)}(\mathbf{W}_{(i)}^{(-n)})^\top - \mathbf{W}_{(j)}^{(-n)}(\mathbf{W}_{(i)}^{(-n)})^\top - \\ &\mathbf{W}_{(i)}^{(-n)}(\mathbf{W}_{(j)}^{(-n)})^\top + \mathbf{W}_{(j)}^{(-n)}(\mathbf{W}_{(j)}^{(-n)})^\top \Big) \Bigg) \end{aligned} \tag{A.26}$$

$$\begin{aligned}\frac{\partial L_2^{(F)}}{\partial \mathbf{U}^{(n)}} &= \sum_{i=1,j\rightsquigarrow i}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Bigg(\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \\ &\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big) \Bigg) \\ &= 2 \sum_{i=1,j\rightsquigarrow i}^{M} \Bigg(\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big)^\top \mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top + \\ &\mathbf{U}^{(n)}(\mathbf{U}^{(n)})^\top \Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big)^\top \Bigg) \mathbf{U}^{(n)} \end{aligned} \tag{A.27}$$

$$\frac{\partial L_2^{(F)}}{\partial \mathbf{U}^{(n)}} = 2\sum_{i=1,j\rightsquigarrow i}^{M}\Bigg(\Big(\mathbf{V}_{(ii)}^{(-n)}-\mathbf{V}_{(ji)}^{(-n)}-\mathbf{V}_{(ij)}^{(-n)}+\mathbf{V}_{(jj)}^{(-n)}\Big)^{\top}\mathbf{T}^{(n)} + \mathbf{T}^{(n)}\Big(\mathbf{V}_{(ii)}^{(-n)}-\mathbf{V}_{(ji)}^{(-n)}-\mathbf{V}_{(ij)}^{(-n)}+\mathbf{V}_{(jj)}^{(-n)}\Big)^{\top}\Bigg)\mathbf{U}^{(n)} \tag{A.28}$$

The third term, $L_3^{(F)}$, accounts for the penalty paid to the close distance impostors. The closer the impostors to the instance, the more probably to make a k-nn misclassification.

$$\begin{aligned} L_3^{(F)} &= \sum_{i=1,j\rightsquigarrow i,l}^{M}(1-y_{il})\Big[C + \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)}-\hat{\mathbf{X}}_{(j)}^{(n)})^{\top}(\hat{\mathbf{X}}_{(i)}^{(n)}-\hat{\mathbf{X}}_{(j)}^{(n)})\Big) \\ &\quad -\mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)}-\hat{\mathbf{X}}_{(l)}^{(n)})^{\top}(\hat{\mathbf{X}}_{(i)}^{(n)}-\hat{\mathbf{X}}_{(l)}^{(n)})\Big)\Big]_{+} \\ &= \sum_{i=1,j\rightsquigarrow i}^{M}(1-y_{il})\Big[C + \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^{\top}\hat{\mathbf{X}}_{(i)}^{(n)}\Big)-\mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^{\top}\hat{\mathbf{X}}_{(j)}^{(n)}\Big)- \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_{(j)}^{(n)})^{\top}\hat{\mathbf{X}}_{(i)}^{(n)}\Big)+\mathrm{tr}\Big((\hat{\mathbf{X}}_{(j)}^{(n)})^{\top}\hat{\mathbf{X}}_{(j)}^{(n)}\Big)- \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^{\top}\hat{\mathbf{X}}_{(i)}^{(n)}\Big)+\mathrm{tr}\Big((\hat{\mathbf{X}}_{(i)}^{(n)})^{\top}\hat{\mathbf{X}}_{(l)}^{(n)}\Big)+ \\ &\quad \mathrm{tr}\Big((\hat{\mathbf{X}}_{(l)}^{(n)})^{\top}\hat{\mathbf{X}}_{(i)}^{(n)}\Big)-\mathrm{tr}\Big((\hat{\mathbf{X}}_{(l)}^{(n)})^{\top}\hat{\mathbf{X}}_{(l)}^{(n)}\Big)\Big]_{+} \end{aligned} \tag{A.29}$$

Using the derivation of gradient of $L_2^{(F)}$, one can easily get:

$$\begin{aligned}
\frac{\partial L_3^{(F)}}{\partial \mathbf{U}^{(n)}} &= 2 \sum_{i=1, j \rightsquigarrow i, l}^{M} (1 - y_{il}) \\
&\Bigg[\Bigg(\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big)^{\top} \\
&\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^{\top} + \mathbf{U}^{(n)} (\mathbf{U}^{(n)})^{\top} \\
&\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big)^{\top} \Bigg) \mathbf{U}^{(n)} \\
&- \Bigg(\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(li)}^{(-n)} - \mathbf{V}_{(il)}^{(-n)} + \mathbf{V}_{(ll)}^{(-n)} \Big)^{\top} \\
&\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^{\top} + \mathbf{U}^{(n)} (\mathbf{U}^{(n)})^{\top} \\
&\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(li)}^{(-n)} - \mathbf{V}_{(il)}^{(-n)} + \mathbf{V}_{(ll)}^{(-n)} \Big)^{\top} \Bigg) \mathbf{U}^{(n)} \Bigg]_{+}
\end{aligned} \tag{A.30}$$

$$\begin{aligned}
\frac{\partial L_3^{(F)}}{\partial \mathbf{U}^{(n)}} &= 2 \sum_{i=1, j \rightsquigarrow i, l}^{M} (1 - y_{il}) \\
&\Bigg[\Bigg(\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big)^{\top} \mathbf{T}^{(n)} + \\
&\mathbf{T}^{(n)} \Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(ji)}^{(-n)} - \mathbf{V}_{(ij)}^{(-n)} + \mathbf{V}_{(jj)}^{(-n)} \Big)^{\top} \Bigg) \mathbf{U}^{(n)} - \\
&\Bigg(\Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(li)}^{(-n)} - \mathbf{V}_{(il)}^{(-n)} + \mathbf{V}_{(ll)}^{(-n)} \Big)^{\top} \mathbf{T}^{(n)} + \\
&\mathbf{T}^{(n)} \Big(\mathbf{V}_{(ii)}^{(-n)} - \mathbf{V}_{(li)}^{(-n)} - \mathbf{V}_{(il)}^{(-n)} + \mathbf{V}_{(ll)}^{(-n)} \Big)^{\top} \Bigg) \mathbf{U}^{(n)} \Bigg]_{+}
\end{aligned} \tag{A.31}$$

The loss function for LMTD-C is defined as follows:

$$L^{(C)} = \mu L_1 + \beta L_2^{(C)} + \gamma L_3^{(C)}$$

The first term, L_1, is the same as in LMTD-F. The second term, $L_2^{(C)}$, stands for the total distances between the instance and its targets in the reduced space (over the core tensors).

$$L_2^{(C)} = \sum_{i=1, j \rightsquigarrow i}^{M} \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)} - \mathbf{G}_{(j)}^{(n)})^\top (\mathbf{G}_{(i)}^{(n)} - \mathbf{G}_{(j)}^{(n)})\Big) \tag{A.32}$$

Let's define $\mathbf{P}_{(ij)}^{(-n)} = \mathbf{X}_i^{(n)} \mathbf{R}^{(-n)} (\mathbf{R}^{(-n)})^\top (\mathbf{X}_j^{(n)})^\top$. Then the gradient of $L_2^{(C)}$ is calculated as below,

$$\begin{aligned}
\frac{\partial L_2^{(C)}}{\partial \mathbf{U}^{(n)}} &= \sum_{i=1, j \rightsquigarrow i}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Bigg(\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \\
&\quad \Big(\mathbf{X}_{(i)}^{(n)} \mathbf{R}^{(-n)} (\mathbf{R}^{(-n)})^\top (\mathbf{X}_{(i)}^{(n)})^\top - \mathbf{X}_{(j)}^{(n)} \mathbf{R}^{(-n)} (\mathbf{R}^{(-n)})^\top (\mathbf{X}_{(i)}^{(n)})^\top - \\
&\quad \mathbf{X}_{(i)}^{(n)} \mathbf{R}^{(-n)} (\mathbf{R}^{(-n)})^\top (\mathbf{X}_{(j)}^{(n)})^\top + \mathbf{X}_{(j)}^{(n)} \mathbf{R}^{(-n)} (\mathbf{R}^{(-n)})^\top (\mathbf{X}_{(j)}^{(n)})^\top \Big) \Bigg) \\
&= \sum_{i=1, j \rightsquigarrow i}^{M} \frac{\partial}{\partial \mathbf{U}^{(n)}} \mathrm{tr}\Big(\mathbf{U}^{(n)} (\mathbf{U}^{(n)})^\top \Big(\mathbf{P}_{ii}^{(-n)} - \mathbf{P}_{ji}^{(-n)} - \mathbf{P}_{ij}^{(-n)} + \mathbf{P}_{jj}^{(-n)} \Big) \Big) \\
&= 2 \sum_{i=1, j \rightsquigarrow i}^{M} \Big(\mathbf{P}_{ii}^{(-n)} - \mathbf{P}_{ji}^{(-n)} - \mathbf{P}_{ij}^{(-n)} + \mathbf{P}_{jj}^{(-n)} \Big) \mathbf{U}^{(n)}
\end{aligned} \tag{A.33}$$

The total distances to the impostors in the reduced space, $L_3^{(C)}$, is calculated as follows:

$$\begin{aligned} L_3^{(C)} &= \sum_{i=1,j\rightsquigarrow i,l}^{M} (1-y_{il})\Big[C + \\ &\quad \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)} - \mathbf{G}_{(j)}^{(n)})^\top (\mathbf{G}_{(i)}^{(n)} - \mathbf{G}_{(j)}^{(n)})\Big) - \\ &\quad \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)} - \mathbf{G}_{(l)}^{(n)})^\top (\mathbf{G}_{(i)}^{(n)} - \mathbf{G}_{(l)}^{(n)})\Big)\Big]_+ \\ &= \sum_{i=1,j\rightsquigarrow i}^{M} (1-y_{il})\Big[C + \\ &\quad \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)})^\top \mathbf{G}_{(i)}^{(n)}\Big) - \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)})^\top \mathbf{G}_{(j)}^{(n)}\Big) - \\ &\quad \mathrm{tr}\Big((\mathbf{G}_{(j)}^{(n)})^\top \mathbf{G}_{(i)}^{(n)}\Big) + \mathrm{tr}\Big((\mathbf{G}_{(j)}^{(n)})^\top \mathbf{G}_{(j)}^{(n)}\Big) - \\ &\quad \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)})^\top \mathbf{G}_{(i)}^{(n)}\Big) + \mathrm{tr}\Big((\mathbf{G}_{(i)}^{(n)})^\top \mathbf{G}_{(l)}^{(n)}\Big) + \\ &\quad \mathrm{tr}\Big((\mathbf{G}_{(l)}^{(n)})^\top \mathbf{G}_{(i)}^{(n)}\Big) - \mathrm{tr}\Big((\mathbf{G}_{(l)}^{(n)})^\top \mathbf{G}_{(l)}^{(n)}\Big)\Big]_+ \end{aligned} \tag{A.34}$$

Using the derivation of gradient of $L_2^{(C)}$, one can easily get:

$$\begin{aligned} \frac{\partial L_3^{(C)}}{\partial \mathbf{U}^{(n)}} &= 2 \sum_{i=1,j\rightsquigarrow i,l}^{M} (1-y_{il}) \\ &\quad \Bigg[\Big(\mathbf{P}_{ii}^{(-n)} - \mathbf{P}_{ji}^{(-n)} - \mathbf{P}_{ij}^{(-n)} + \mathbf{P}_{jj}^{(-n)}\Big) - \\ &\quad \Big(\mathbf{P}_{ii}^{(-n)} - \mathbf{P}_{li}^{(-n)} - \mathbf{P}_{il}^{(-n)} + \mathbf{P}_{ll}^{(-n)}\Big)\Bigg] \mathbf{U}^{(n)} \end{aligned} \tag{A.35}$$