Stochastic Domain Transfer Multiple-Kernel Boosting with Application to Anomaly Detection in Encrypted Network Traffic

Noah Reef
Undergraduate - Department of Computer Science
California State Polytechnic University, Pomona
Pomona, CA
nbreef@cpp.edu

Dr. Abdelfattah Amamra
Faculty - Department of Computer Science
California State Polytechnic University, Pomona
Pomona, CA
aamamra@cpp.edu

I. Introduction

In the field of *Cybersecurity* it is important to be able to detect and flag network activity as either containing malware or not containing malware, but this task becomes increasingly difficult as encryption technologies obfuscate the contents of such traffic. As of 2021, 46 percent of malware detected on the internet was concealed by TLS [5] and was projected to increase.

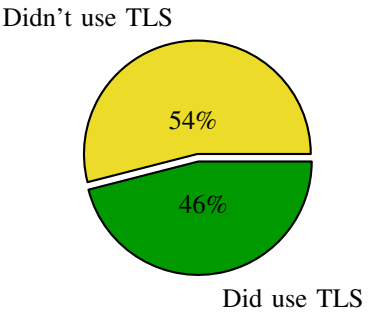

Fig. 1. Percentage of Malware That Used TLS in 2021[5]

Many proposed solutions to detecting malware in encrypted network traffic attempt to use machine learning classifiers, such as Random Forest, Naive-Bayes, and Neural Network models to predict the presence of malware in such traffic [7, 8, 9, 1, 11].

However, there still exists the issue that access to the contents of such encrypted data can be difficult due to privacy concerns which may lead to not having enough data to properly train a machine learning model. One solution to this problem is **Domain Adaptation** (DA), which allows us to utilize knowledge from *source(s)* domain with more data and create a accurate classifier for our target domain.

In the case of Encrypted Network Traffic we can utilize datasets of *unencrypted* data, which is more accessible, and the fact that both *encrypted* and *unencrypted* have similar meta-data features (i.e. Connection time, Communication ports, etc.), to train a classifier to detect malicious activity and utilize DA to detect malicious network activity in Encrypted Network Traffic. Since we receive data from multiple log files and datasets with different distributions the application of a **Multiple Kernel Classifier** will be our main focus.

In this paper we will:

- Introduce a novel framework of Stochastic Domain Transfer Multi-Kernel Learning, which introduces a stochastic boosting approach to Domain Transfer Multiple-Kernel Learning
- Perform experiments comparing the accuracy of Stochastic DTMKB against other conventional Multi-Kernel Transfer Learning Algorithms

II. Related Work

In [3] they introduce a Domain-Transfer Multiple Kernel Learning algorithm (DTMKL), that learns a classifier and optimal kernel matrix simultaneously. This idea is expanded upon in [12] and [2] which applies Adaboosting techniques to the DTMKL algorithm introduced in [3]. Unlike developing a regression classifier as in [12], we design our algorithm for a binary classification, that is whether malware is or is not present in current network traffic. e The application of Domain Adaptation Multi-Kernel Learning has been applied in many different fields such as EEG imaging[2], Email Spam Detection[3], and Electric Load Forecasting [12] but never in the field of Cybersecurity, especially in anomaly detection in Encrypted Network Traffic.

III. Stochastic Domain Transfer Multiple Kernel Learning

A. DTMKL

Let us denote the data set patterns from the target domain as $D^T = (x_i^T, y_i^T)|_{i=1}^T$, where y_i^T is the label of x_i^T, and $D^S = (x_i^S, y_i^S)|_{i=1}^S$. We denote the marginal distribution of D^T as $\mathcal{P}$ and of D^S as $\mathcal{Q}$. We form our training data from both domains as, $D = D^T \cup D^S$. In the DTMKL framework our objective is to learn the true classifier for the target domain, namely

$$f(x) = \sum_{i=1}^{n} \gamma_i k(x_i, x) + b$$

as well as the kernel function k simultaneously, where γ_i is the factor term of the kernel expansion for $f(x)$ and b is our bias term.

The learning framework of DTMKL is formulated as,

$$[k, f] =_{k,f} \Omega(^2_k(D^S, D^T)) + \theta R(k, f, D) \tag{1}$$

where $\Omega(.)$ is a monotonic increasing function, $\theta > 0$ is a trade-off parameter between the discrepancy of data distributions and structural risk, $R(k, f, D)$ is the Structural Risk, and $_k(D^S, D^T)$ is the *Maximum Mean Discrepancy* (MMD) which is defined as,

$$_k(D^S, D^T) = \left\| \frac{1}{N_S} \sum_{i=1}^{N_S} \phi(x_i^S) - \frac{1}{N_T} \sum_{i=1}^{N_T} \phi(x_i^T) \right\| \tag{2}$$

DTMKL works to minimize the distribution distance between our domains, D^T and D^S by using MMD. We first define $\Phi = [\phi(x_1^S), \ldots, \phi(x_{N_S}^S), \phi(x_1^T), \ldots, \phi(x_{N_T}^T)]$ as the kernel matrix and a column vector $\mathbf{z}$ with $N_S + N_T$ entries, with the first N_S entries as $1/N_S$ and the remaining entries as $-1/N_T$. We also simplify the argument in (2) as Φ_Z. Thus (2) is rewritten as,

$$^2_k(D^S, D^T) = \|\Phi_Z\|^2 = (\Phi^T \Phi \mathbf{Z}) = (\mathbf{KZ}) \tag{3}$$

where,

$$\mathbf{Z} = z^T z \tag{4}$$

$$\mathbf{K} = \Phi^T \Phi \tag{5}$$

Instead of trying to solve the non-parametric kernel matrix $\mathbf{K}$, we assume that the kernel can be represented as a linear combination of base kernels,

$$k = \sum_{i=1}^{I} d_i k_i,$$

where $d_i \geq 0$ and $\sum_{i=1}^{I} d_i = 1$. Thus we can define $\Omega((\mathbf{KZ}))$ as

$$\Omega((\mathbf{KZ})) = \frac{1}{2}((\mathbf{KZ}))^2 = \frac{1}{2}\left(\left(\sum_{i=1}^{I} d_i \mathbf{K}_i \mathbf{Z}\right)\right) = \frac{1}{2}\mathbf{d}^T \mathbf{w}\mathbf{w}^T \mathbf{d},$$

where,

$$\mathbf{w} = [w_1, \ldots, w_I]^T,$$
$$w_i = (\mathbf{K_i Z}), \text{ where } \quad \mathbf{K_i} = [k_i(x_a, x_b)]$$
$$\mathbf{d} = [d_i, \ldots, d_I]^T$$

and thus (1) can be rewritten and our optimization problem becomes,

$$\min_{d \in \mathcal{D}} \left(\frac{1}{2} \mathbf{d}^T \mathbf{w}\mathbf{w}^T \mathbf{d} + \theta J(\mathbf{d}) \right) \tag{6}$$

where $J(\mathbf{d}) = \min_f R(\mathbf{d}, f, \mathcal{D})$ and $\mathcal{D} = \{\mathbf{d} | \mathbf{d} \geq 0, \mathbf{d}^T 1_I = 1\}$ is the feasible set of $\mathbf{d}$ and f is the target function. Next, by letting our Structural Risk be *Hinge Loss* as in [4] [3] we get

$$\min_{\mathbf{d} \in \mathcal{D}} \min_f \frac{1}{2} d^T w w^T d + \theta \text{SVM}^{\text{primal}}(\mathbf{d}, f, D)$$

which we will rewrite as

$$\min_{\mathbf{d} \in \mathcal{D}} \min_{\mathbf{g_m}, b, \xi} \frac{1}{2} d^T w w^T d + \theta \underbrace{\left(\frac{1}{2} \sum_{m=1}^{M} \frac{\|\mathbf{g_m}\|^2}{w_m} + C \sum_{i=1}^{n} \xi_i \right)}_{\text{J(d)}} \tag{7}$$

$$\text{s.t} \quad y_i \left(\sum_{m=1}^{M} \mathbf{g_m^T} \phi_m(\mathbf{x_i}) + b \right) \geq 1 - \xi_i, \xi \geq 0 \tag{8}$$

Following from [3], the right-half of (7) is convex and we write its Lagrangian dual with dual coefficients α as,

$$J(\mathbf{d}) = \max_{\alpha \in \mathcal{A}} \mathbf{1}'_n \alpha - \frac{1}{2}(\alpha \circ y)' \left(\sum_{m=1}^{M} d_m K_m \right)(\alpha \circ y) \tag{9}$$

where we hold $\mathbf{d}$ to be constant and $\alpha \in \mathcal{A} = \{\alpha | \alpha^T \mathbf{y} = 0, \mathbf{0_n} \leq \alpha \leq C\mathbf{1}_n\}$. If we let α be constant we can update $\mathbf{d}$ by using the reduced gradient described in [10],

$$\nabla h = \mathbf{z}\mathbf{z}^T \mathcal{D} + C \nabla J \tag{10}$$

where ∇J is the reduced gradient of J in (6) and the Hessian Matrix is derived as,

$$\nabla^2 h = \mathbf{z}\mathbf{z}^T + C \nabla^2 J \tag{11}$$

We define $v = (\nabla^2 h)^{-1} \nabla h$ as the updating direction and our updating of our weights as,

$$d_{t+1} = d_t - \eta_t v_t \tag{12}$$

Thus by gradient descent, we can start by letting $\mathbf{d}$ be constant and solve for the optimal α (by standard SVM), then use the α found as a constant in updating $\mathbf{d}$ by equation (12).

Algorithm 1 DTMKL Algorithm

INITIALIZATION: d = $\frac{1}{M}$ 1_M

for t = 1, …, T_{max} **do**

Solve α in

$$\max_{\alpha \in \mathcal{A}} \mathbf{1}'_n \alpha - \frac{1}{2}(\alpha \circ y)' \left(\sum_{m=1}^{M} d_m K_m \right)(\alpha \circ y)$$

Update d using

$$d_{t+1} = d_t - \eta_t g_t \in \mathcal{D}$$

and get the following classifier,

$$f(\mathbf{x}) = \text{sign}\left(\sum_{i\neq 0} \alpha_i y_i \sum_{m=1}^{M} d_m k_m(\mathbf{x_i}, \mathbf{x}) + b\right)$$

B. Non - Stochastic DTMKB

We will now introduce the Boosting approach to DTMKL proposed in ([2],[4],[12]) we start similar to [3] with our goal of learning a classifier for our target domain as,

$$f(x_i) = \sum_{i=t}^{T} \alpha_t f_t(x_i) \tag{13}$$

with the problem of trying to learn the optimal f_t and α_t for our target classifier, which is addressed via a booting method, that is during each boosting trial, t, we sample a subset of data from D by distribution D_t and train a weak classifier using all the kernels in the set of base kernels K and compute its error,

$$\epsilon_t = \sum_{i=1}^{N} D_t(i)(h_t(x_i) \neq y_i) \tag{14}$$

and compute the weak classifiers weight by,

$$\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right) \tag{15}$$

Then after each boosting trial we update D_t by,

$$D_{t+1}(i) + \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & , h_t(x_i) = y_i \\ e^{\alpha_t} & , h_t(x_i) \neq y_i \end{cases} \tag{16}$$

Algorithm 2 Non-Stochastic DTMKB Algorithm

INPUT:
- Training data: $D = D^T \cup D^S$
- Initial Data Distribution $(D_1(i) = 1/N; i = 1, \dots N)$
- Set of base kernels: $\mathbf{K}, k_j \in \mathbf{K}, j = 1, \dots, J$
- Number of iterations T

for $t = 1, \dots, T$ **do**
- sample n samples using distribution D_t
- train weak classifier with set of base kernels $\mathbf{K}$
  - $h_t : \mathcal{X} \to \{-1, +1\}$
- compute the training error over D_t:
  - $\epsilon_t = \epsilon(h_t) = \sum_{i=1}^{N} D_t(i)(h_t(x_i) \neq y_i)$
- compute weight α_t:
  - $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
- update $D_{t+1}(i)$:
  - $D_{t+1}(i) + \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & , h_t(x_i) = y_i \\ e^{\alpha_t} & , h_t(x_i) \neq y_i \end{cases}$

End

OUTPUT: $f(x) = \text{sign}(\sum_{i=1}^{T} \alpha_t h_t(x))$

C. Stochastic DTMKB

This method works well for cases where we have a small number of base kernels, but can become computationally taxing when dealing with a larger set of base kernels. Thus we will apply the idea of *Stochastic Multi-kernel Boosting* proposed in [13], that is instead of training every kernel during each boosting trial we will instead train only a subset K_t of the base kernels by some distribution. We start by introducing a variable, S_t which represents the sampling probability for each base kernel to be chosen during a given boosting trial. We start with all base kernel probabilities at 1, that is

$$S_t(i) = 1; \quad i = 1, \dots, J$$

and update the probability of M base kernels chosen during a given boosting trial by,

$$S_{t+1}(m) = S_t(m)\beta^{\epsilon_t}; \quad m = 1, \dots M$$

where $\beta \in (0, 1)$ is a constant parameter representing the sampling decay and ϵ_t is the error of the t_{th} weak classifier over the entire dataset. Lastly we normalize the probabilities of all the base kernels by,

$$S_{t+1}(i) = S_{t+1}(i)/\max S_{t+1}; \quad i = 1, \dots, J$$

Algorithm 3 Stochastic DTMKB Algorithm

INPUT:
- Training data: $D = D^T \cup D^S$
- Initial Data Distribution $(D_1(i) = 1/N; i = 1, \dots N)$
- Set of base kernels: $\mathbf{K}, k_j \in \mathbf{K}, j = 1, \dots, J$
- Initial Kernel Distribution: $S_1(j) = 1; j = 1, \dots, J$
- Sampling Decay Rate: $0 < \beta < 1$
- Number of iterations T

for $t = 1, \dots, T$ **do**
- sample n samples using distribution D_t
- train weak classifier by DTMKL with subset of M base kernels
  - $k_1, \dots, k_M \in \mathbf{K_t} \subseteq \mathbf{K}$:
    - $h_t : \mathcal{X} \to \{-1, +1\}$
  - compute the training error over D_t:
    - $\epsilon_t = \epsilon(h_t) = \sum_{i=1}^{N} D_t(i)(h_t(x_i) \neq y_i)$
  - compute weight α_t:
    - $\alpha_t = \frac{1}{2}\ln\left(\frac{1-\epsilon_t}{\epsilon_t}\right)$
  - update $S_{t+1}(m) = S_t(m)\beta^{\epsilon_t}; m = 1, \dots, M$
  - normalize $S_{t+1}(i) = S_{t+1}(i)/max(S_{t+1})$
- update $D_{t+1}(i)$:
  - $D_{t+1}(i) + \frac{D_t(i)}{Z_t} \times \begin{cases} e^{-\alpha_t} & , h_t(x_i) = y_i \\ e^{\alpha_t} & , h_t(x_i) \neq y_i \end{cases}$

End

OUTPUT: $f(x) = \text{sign}(\sum_{i=1}^{T} \alpha_t h_t(x))$

IV. Experiment

A. Dataset

Here we use two datasets, Scenario A and Scenario B, from the ISCXTor2016 [6], one as our Source Domain and the other as our Target Domain. Scenario A has labels Tor/Non-Tor traffic. Scenario B uses labels of 7 different types of traffic: Browsing, Email, Chat, Audio-Streaming, Video-Streaming, FTP, VoIP, and P2P all encrypted by Tor. Both datasets use:

Source/Destination IP and Port, Protocol, Flow Duration. Flow of Bytes and Packets, Flow IAT (mean, std, max, min), Fwd IAT (mean, std, max, min), Bwd (mean, std, max, min), Active (mean, std, max, min), and Idle (mean, std, max, min) as features.

B. Experimental Setup

For our experiment we run Stochastic DTMKB on the ISCXTor2016 [6] Scenario A and Scenario B datasets mentioned above. Here we use Scenario B as our Source domain and Scenario A as our Target domain. Here we make the labels in Scenario A as the 1 for Tor and 0 for Non-Tor traffic. To keep our goal of binary classification, we make Audio in Scenario B be our 1 label and the rest 0. For our preliminary experiments we use 200 samples from the Scenario B and 20 samples from Scenario A for our training data, and use 150 samples from Scenario A as our testing data. For our Stochastic DTMKB algorithm we use the following base kernels:

- Linear

$$k(x, y) = x^T y$$

- Polynomial

$$k(x, y) = (\gamma x^T y)^d \text{ for } d = 1, \ldots, 4 \text{ and } \gamma = \frac{1}{\text{\# of features}}$$

- Sigmoid

$$k(x, y) = \tanh(\gamma x^T y) \text{ with } \gamma = 2^{-6}, 2^{-3}, 1, 2^3, 2^6$$

- Gaussian

$$k(x, y) = \exp\left(-\gamma \|x - y\|^2\right) \text{ with } \gamma = 2^{-6}, 2^{-3}, 1, 2^3, 2^6$$

for a total of 15 base kernels. Following [12] [13] we let $T = 100$, $\theta = 1$, and use a Sampling Ratio of 0.2. Additionally we let our sampling decay, $\beta = 2^{-5}$ and let $M = 3$ be the number of kernels sampled during each boosting trial. In the case of DTMKL for training our weak classifiers, we let $\epsilon = 10^{-2}$ by [4] and let the maximum number of iterations for convergence be 10. Code for the experiment is available at: shttps://github.com/Noah-B-Reef/Stochastic-Domain-Transfer-MKB

C. Results

We ran the Stochastic DTMKB algorithm 20 times on the same training and testing data mentioned above, we found that in most cases Stochastic DTMKB had an accuracy of 0.89 in 17 of the iterations with a much lower accuracy in 3 of the iterations. The results are shown below in Table 1.

TABLE I
AUDIO/NON-AUDIO TO TOR/NON-TOR RESULTS

T	Time	Accuracy
1	424.9183722	0.1933333333
2	397.4979217	0.8933333333
3	405.539094	0.14
4	401.8992786	0.8933333333
5	367.1108491	0.8933333333
6	386.1906197	0.8933333333
7	399.4398687	0.8933333333
8	387.6284196	0.8933333333
9	387.1271613	0.8933333333
10	381.1384251	0.8933333333
11	393.5002234	0.8933333333
12	436.6479099	0.46
13	409.8955326	0.8933333333
14	418.1498699	0.8933333333
15	428.5180151	0.8933333333
16	399.6815038	0.8933333333
17	402.6275311	0.8933333333
18	422.2576659	0.8933333333
19	396.3889489	0.8933333333
20	388.53458	0.8933333333

V. Conclusions and Future Work

Here we see that Stochastic DTMKB has promising results of a high accuracy in making predictions in the target domain when only trained on a labeled dataset with most samples from the source domain. For future experiments using the Stochastic DTMKB algorithm, it would be worth seeing how accuracy changes when using larger Training and Testing datasets, as well as varying parameters such as: number of boosting trials, sampling ratio, number of base kernels, and sampling decay. In our experiment we used classification of the type of network traffic as our source task and classification of encrypted or not encrypted network traffic as our target task, for future experiments using unencrypted network malware detection as our source domain and encrypted network malware detection as our target domain would be desired.

Acknowledgment

This material is based upon work supported by the National Science Foundation.

A. Disclaimer

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation

References

[1] Dmitri Bekerman et al. "Unknown malware detection using network traffic classification". In: *2015 IEEE Conference on Communications and Network Security (CNS)* (2015). DOI: 10.1109/cns.2015.7346821.

[2] Mengxi Dai et al. "Domain transfer multiple kernel boosting for classification of EEG motor imagery signals". In: *IEEE Access* 7 (2019), pp. 49951–49960. DOI: 10.1109/access.2019.2908851.

[3] Lixin Duan, I. W. Tsang, and Dong Xu. “Domain transfer multiple kernel learning”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.3 (2012), pp. 465–479. DOI: 10.1109/tpami.2011.114.

[4] Lixin Duan et al. “Domain transfer SVM for video concept detection”. In: *2009 IEEE Conference on Computer Vision and Pattern Recognition* (2009). DOI: 10.1109/cvpr.2009.5206747.

[5] Written by Sean Gallagher. *Nearly half of malware now use TLS to Conceal Communications*. Aug. 2021. URL: https://news.sophos.com/en-us/2021/04/21/nearly-half-of-malware-now-use-tls-to-conceal-communications/.

[6] Arash Habibi Lashkari et al. “Characterization of tor traffic using time based features”. In: *Proceedings of the 3rd International Conference on Information Systems Security and Privacy* (2017). DOI: 10.5220/0006105602530262.

[7] Jarilyn Hernandez Jimenez and Katerina Goseva-Popstojanova. “Malware detection using power consumption and network traffic data”. In: *2019 2nd International Conference on Data Intelligence and Security (ICDIS)* (2019). DOI: 10.1109/icdis.2019.00016.

[8] Michal Piskozub et al. “Malphase: Fine-grained malware detection using network flow data”. In: *Proceedings of the 2021 ACM Asia Conference on Computer and Communications Security* (2021). DOI: 10.1145/3433210.3453101.

[9] Paul Prasse et al. “Malware detection by analysing network traffic with Neural Networks”. In: *2017 IEEE Security and Privacy Workshops (SPW)* (2017). DOI: 10.1109/spw.2017.8.

[10] Alain Rakotomamonjy, Francis Bach, Stéphane Canu, et al. “SimpleMKL”. In: *Journal of Machine Learning Research* 9 (2008), pp. 2491–2521.

[11] Phani Vadrevu et al. “Measuring and detecting malware downloads in live network traffic”. In: *Lecture Notes in Computer Science* (2013), pp. 556–573. DOI: 10.1007/978-3-642-40203-6_31.

[12] Di Wu et al. “Multiple kernel learning-based transfer regression for electric load forecasting”. In: *IEEE Transactions on Smart Grid* 11.2 (2020), pp. 1183–1192. DOI: 10.1109/tsg.2019.2933413.

[13] Hao Xia and Steven C. Hoi. “MKBoost: A framework of multiple kernel boosting”. In: *Proceedings of the 2011 SIAM International Conference on Data Mining* (2011). DOI: 10.1137/1.9781611972818.18.