Robust Representation for Domain Adaptation in Network Security

Karel Bartos[1,2] and Michal Sofka[2]

[1] Faculty of Electrical Engineering, Czech Technical University in Prague, Prague, Czech Republic
[2] Cisco Systems, Karlovo namesti 10, 12000 Prague, Czech Republic
kbartos@cisco.com, msofka@cisco.com

Abstract. The goal of domain adaptation is to solve the problem of different joint distribution of observation and labels in the training and testing data sets. This problem happens in many practical situations such as when a malware detector is trained from labeled datasets at certain time point but later evolves to evade detection. We solve the problem by introducing a new representation which ensures that a conditional distribution of the observation given labels is the same. The representation is computed for bags of samples (network traffic logs) and is designed to be invariant under shifting and scaling of the feature values extracted from the logs and under permutation and size changes of the bags. The invariance of the representation is achieved by relying on a self-similarity matrix computed for each bag. In our experiments, we will show that the representation is effective for training detector of malicious traffic in large corporate networks. Compared to the case without domain adaptation, the recall of the detector improves from 0.81 to 0.88 and precision from 0.998 to 0.999.

1 Introduction

In supervised learning, the domain adaptation solves the problem when a joint distribution of the labels and observations differs for training (source) and testing (target) data. This can happen as a result of target evolving after the initial classifier was trained. For example, in network security, the classifier is trained from network traffic samples of malware communication which can change as a result of evolving malware. Under the assumption that the source and target distribution do not change arbitrarily, the goal of the domain adaptation is to leverage the knowledge in the source domain and transfer it to the target domain. In this work, we focus on the case where the conditional distribution of the observation given labels is different, also called a conditional shift.

The knowledge transfer can be achieved by adapting the detector using importance weighting such that training instances from the source distribution match the target distribution [16]. Another approach is to transform the training instances to the domain of the testing data or to create a new data representation with the same joint distribution of observations and labels [1]. The challenging

part is to design a meaningful transformation that transfers the knowledge from the source domain and improves the robustness of the classifier on the target domain.

In this paper, we present a new invariant representation of network traffic data suitable for domain adaptation under conditional shift. The representation is computed for bags of samples, each of which consists of features computed from network traffic logs. A bag is constructed for every user and all network communication with each domain. The representation is designed to be invariant under shifting and scaling of the feature values and under permutation and size changes of the bags. This is achieved by constructing an invariant self similarity matrix for each bag. Pairwise relevance measure is trained to reliably assign previously-unseen bags to existing categories or to create a new category.

The proposed similarity measure and the new invariant representation is applied to detect malicious HTTP traffic in network security. We will show that the classifier trained on malware communication samples from one category can successfully detect new samples from a different category. This way, the knowledge of the malware behavior is correctly transferred to the new domain which improves the classifier. Compared to the case without adaptation with 0.81 recall and 0.998 precision, the new approach has recall 0.88 and precision 0.999.

2 Problem Statement

The paper deals with the problem of supervised classification of bags of samples into categories with a lack of labeled data. The labels for positive and negative samples are often very expensive to obtain. Moreover, sample distribution typically evolves in time, so the probability distribution of training data differs from the probability distribution of test data. In contrast to the case when enough samples is available in each category and their distributions are stationary, the knowledge needs to be transferred in time within categories but also across categories using labeled samples. In the following, the problem is described in more detail.

Each sample is represented as an n-dimensional vector $\mathbf{x} \in \mathbb{R}^n$. Samples are grouped into bags, where i-th bag is a set of m_i samples $X_i = \{\mathbf{x}_1, \ldots, \mathbf{x}_{m_i}\} \in \mathcal{X}$. A single category y_i can be assigned to each bag from the set of categories $\mathcal{Y} = \{y_1, \ldots, y_N\}$. Note that not all categories are included in the training set. The probability distribution on training (labeled) and test bags for category y_j will be denoted as $P^L(X|y_j)$ and $P^T(X|y_j)$, respectively. Moreover, the probability distribution of training data differs from the probability distribution of testing data, a problem dealt with in the domain adaptation [2] (also called a conditional shift [18]):

$$P^L(X|y_j) \neq P^T(X|y_j), \;\; \forall y_j \in \mathcal{Y}. \tag{1}$$

The purpose of the domain adaptation is to acquire knowledge from the training (source) domain and apply it to the testing (target) domain. The relation between $P^L(X|y_i)$ and $P^T(X|y_i)$ is not arbitrary, otherwise it would not be possible to transfer any knowledge. Therefore there is a transformation τ,

which transforms the feature values of the bags onto a representation, in which $P^L(\tau(X)|y_i) = P^T(\tau(X)|y_i)$. We assume that $\tau(X)$ is any representation that is invariant against shift, scale, permutation, and size changes of the bag. The goal is to find this representation, allowing to classify individual bags X_i into categories $\mathcal{Y} = \{y_1, \ldots, y_N\}$ under the conditional shift.

A number of other methods for transfer learning have been proposed, including kernel mean matching [10], kernel learning approaches [8], maximum mean discrepancy [11], or boosting [7]. These methods try to solve a general data transfer with relaxed conditions on the similarity of the distributions during the transfer. The downside of these methods is the necessity to specify the target loss function and the availability of large amount of labeled data.

Our solution to the conditional shift problem is to transform the features to a new representation. The advantage of this approach is that it is independent of the classification loss function and similarity between the probability distributions does not need to be given. The method achieves the knowledge transfer by changing the original feature values. The feature values are transformed into a new representation that is invariant against shift, scale, permutation, and size changes of the bags (number of samples within each bag). Once the data are transformed according to the proposed representation, the new feature values do not follow the original distribution and therefore they are not influenced by the shift.

To compensate for the lack of labeled data, a simple online linear transformation is applied. The transformation learns a set of weights on the new features to match the training and test distributions of the bags from the same category. At the same time, the weights are optimized to separate bags belonging to different categories. This way, bags belonging to the same category are assigned the same label during classification.

3 Invariant Representation of Bags

In this Section, an invariant representation of bags is proposed to overcome the problem of domain shift introduced in Section 2. The new representation is calculated with a transformation τ that consists of three steps to ensure that the new representation will be independent on the mean, and invariant against scaling, shifting, permutation and size of the bags. In the following, the individual steps are discussed in more detail.

3.1 Shift Invariance with Self-similarity Matrix

As stated in Section 2, the probability distribution of bags from the training set and the testing set can be different. Therefore, in the first step, the representation of bags is transformed to be invariant against this shift. The traditional representation of i-th bag X_i that consists of a set of m samples $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ is

typically in a form of a matrix:

$$X_i = \begin{pmatrix} \mathbf{x}_1 \\ \vdots \\ \mathbf{x}_m \end{pmatrix} = \begin{pmatrix} x_1^1 & x_1^2 & \dots & x_1^n \\ & & \vdots & \\ x_m^1 & x_m^2 & \vdots & x_m^n \end{pmatrix},$$

where x_l^k denotes k-th feature value of l-th sample from bag X_i. This form of representation of samples and bags is widely used, as it is straightforward to compute. It is a reasonable choice in many applications with negligible difference in probability distributions. However, when the difference becomes more serious, the traditional representation often leads to unsatisfactory results. Therefore, the following transformation is proposed to overcome the difference typically caused by the dynamics of the domain, making the solution for the classification problem more effective. As a first step, the representation is transformed to be invariant against shift of the feature values.

Shift invariance guaranties that even if some original feature values of all samples in a bag are increased/decreased by a given amount, the values in the new representation remain unchanged.

Let us define a *translation invariant distance function*, which is a distance function $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$ such that:

$$d(x_1, x_2) = d(x_1 + a, x_2 + a). \tag{2}$$

Let x_p^k, x_q^k be k-th feature values of p-th and q-th sample from bag X_i, respectively. It is possible to express the relation between the values as follows:

$$x_p^k = x_q^k - s_{pq}^k, \tag{3}$$

where s_{pq}^k is the difference between values x_p^k, x_q^k. Then it holds for each translation invariant distance function $d : \mathbb{R} \times \mathbb{R} \to \mathbb{R}$:

$$d(x_p^k, x_q^k) = d(x_p^k, x_p^k + s_{pq}^k) = d(0, s_{pq}^k) = s_{pq}^k.$$

Therefore, the new feature value $d(x_p^k, x_q^k)$ expresses the distance between the two values of k-th feature regardless of their absolute values. This value is more robust, however it could be less informative, as the information about the absolute values was removed. To compensate for the possible loss of information, the bags are represented with a matrix of these distances $d(x_p^k, x_q^k)$, which is called a self-similarity matrix S^k. Self-similarity matrix is a symmetric positive semidefinite matrix, where rows and columns represent individual samples and (i, j)-th element corresponds to the distance between i-th and j-th sample. Self-similarity matrix has been already used thanks to its properties in several applications (e.g. in object recognition [12] or music recording [14]). However, only a single self-similarity matrix for each bag has been used in these approaches.

This paper proposes to compute a set of similarity matrices, one for every feature. More specifically, a per-feature self-similarity set of matrices $\mathbf{S}_i = \{S_i^1, S_i^2, \ldots, S_i^n\}$ is computed for i-th bag X_i, where

$$S_i^k = \begin{pmatrix} s_{11}^k & s_{12}^k & \ldots & s_{1m}^k \\ s_{21}^k & s_{22}^k & \ldots & s_{2m}^k \\ & & \vdots & \\ s_{m1}^k & s_{m2}^k & \ldots & s_{mm}^k \end{pmatrix} = \begin{pmatrix} d(x_1^k, x_1^k) & d(x_1^k, x_2^k) & \ldots & d(x_1^k, x_m^k) \\ d(x_2^k, x_1^k) & d(x_2^k, x_2^k) & \ldots & d(x_2^k, x_m^k) \\ & & \vdots & \\ d(x_m^k, x_1^k) & d(x_m^k, x_2^k) & \ldots & d(x_m^k, x_m^k) \end{pmatrix}, \quad (4)$$

and $s_{pq}^k = d(x_p^k, x_q^k)$ is a distance between feature values x_p^k and x_q^k of k-th feature. This means that the bag X_i with m samples and n features will be represented with n self-similarity matrices of size $m \times m$.

3.2 Scale Invariance with Local Feature Normalization

As explained in the previous section, self-similarity matrix S_i of the bag X_i captures mutual distances among the samples included in X_i. Therefore, the matrix describes inner temporal dynamics of bags [12], [13]. In other words, it describes how the bag is evolving in time. In case of a bag, where all samples are the same, the matrix S_i will be composed of zeros. On the other hand, in case of a bag with many different samples, the self-similarity matrix will be composed of a wide range of values.

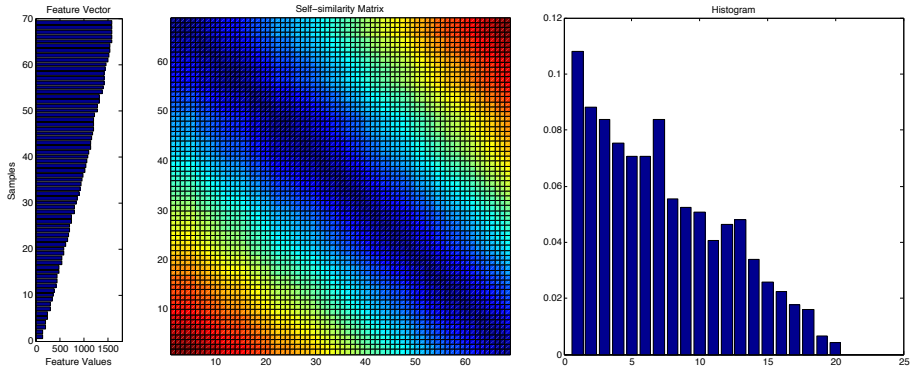
The next step is to transform the matrix S_i^k to be invariant against scaling. **Scale invariance** guarantees that even if some original feature values of all samples in a bag are multiplied by a common factor, the values in the new representation remain unchanged. To guarantee the scale invariance, the matrix S_i^k needs to be locally normalized onto the interval $[0, 1]$ as follows:

$$\tilde{S}_i^k = \begin{pmatrix} \tilde{s}_{11}^k & \tilde{s}_{12}^k & \ldots & \tilde{s}_{1m}^k \\ \tilde{s}_{21}^k & \tilde{s}_{22}^k & \ldots & \tilde{s}_{2m}^k \\ & & \vdots & \\ \tilde{s}_{m1}^k & \tilde{s}_{m2}^k & \ldots & \tilde{s}_{mm}^k \end{pmatrix}, \qquad \tilde{s}_{pq}^k = \frac{s_{pq}^k - \min_{i,j}(s_{ij}^k)}{\max_{i,j}(s_{ij}^k) - \min_{i,j}(s_{ij}^k)}. \quad (5)$$

Note that the maximum and minimum value is computed only from the samples within the bag, therefore the normalization is referred to as local. After the local scaling, the matrices $\tilde{\mathbf{S}}_i = \{\tilde{S}_i^1, \tilde{S}_i^2, \ldots, \tilde{S}_i^n\}$ are invariant against shifting and scaling, focusing purely on the dynamics among the samples (matrix of differences) and not on the absolute values of the differences. An example of an input feature vector and the corresponding locally-normalized self-similarity matrix is illustrated in Figure 1 (a) and Figure 1 (b).

3.3 Permutation and Size Invariance with Histograms

Representing bags with locally-scaled self-similarity matrices $\tilde{\mathbf{S}}$ achieves the scale and shift invariance. However, as there are no restrictions on the size of the



(a) (b) (c)

Fig. 1: Graphical illustration of the individual steps that are needed to transform the bag (set of samples) into the proposed invariant representation. First, the bag is represented with a standard feature vector (a). Then the locally normalized self-similarity matrix (b) is computed for each feature. Finally, values from the matrix will create a new histogram (c), which is invariant on the number or the ordering of the samples within the bag.

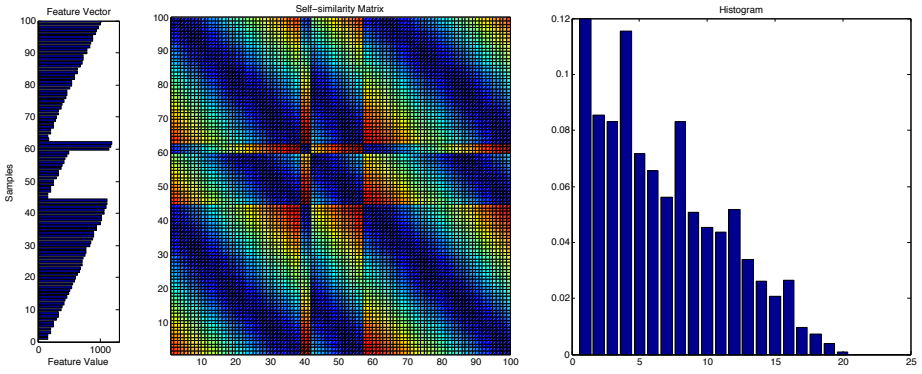
bags (i.e. how many samples are included in a bag), the corresponding self-similarity matrices can be of various sizes. The various sizes of the matrices make their comparison difficult. This is solved by introducing **size invariance** which ensures that the representation does not depend on the size of the bags. Moreover, in highly dynamic environments, the samples may occur in a variable ordering. Since the sample order does not matter for the representation of the bags, the robustness to reordering of rows and columns is guaranteed by the **permutation invariance**.

The final step of the proposed transformation is the transition from the scaled self-similarity matrices $\tilde{\mathbf{S}}_i = \{\tilde{S}_i^1, \tilde{S}_i^2, \ldots, \tilde{S}_i^n\}$ into histograms. Every matrix $\tilde{S}_i^k$ is transformed into a single histogram $\mathbf{h}_i^k$ with a predefined number of bins. Each bin of a histogram $\mathbf{h}_i^k$ represents one feature value in the proposed new representation.

Overall, i-th bag is represented as a vector $\mathbf{h}_i$ of size $n \times l$ as follows:

$$\mathbf{h}_i = (\mathbf{h}_i^1, \mathbf{h}_i^2, \ldots, \mathbf{h}_i^n), \tag{6}$$

where n is the number of features (and histograms) and l is the number of bins. The whole transformation is depicted in Figure 1. Figure 2 illustrates the invariant properties of the representation. Even though the bags from Figure 1 (a) and Figure 2 (a) have different number of samples, ordering, and range of the original feature values, the output histograms are similar.



(a) (b) (c)

Fig. 2: Graphical illustration showing the invariant properties of the proposed representation for one feature. Even tough the bag (and thus the input feature vector (a)) has more samples than the bag from Figure 1, the histogram (c) computed from the self-similarity of the samples (b) is similar.

4 Online Similarity Learning

Representing input bags, as proposed in Section 3, ensures invariance against the conditional shift described in Section 2. Therefore, the transformed feature values can be used for learning a classifier that classifies the bags into categories.

As mentioned in Section 2, some categories may be missing in the training set. The classification method should be able to identify them in the test data and separate them from the rest of the categories. Several existing approaches have been proposed to address the classification problem with missing labels in the training set, e.g. zero-shot learning with semantic output codes [15] or through cross-modal transfer [17]. However, these approaches are typically designed for many labeled samples. When the number of labeled samples is limited, a similarity-based approach [5] can be used.

Similarity-based classifiers estimate the category label from a pairwise similarity of a testing bag and a set of labeled training bags. The comparison between two bags is performed by computing a similarity of feature vectors $\mathbf{h}_i$ and $\mathbf{h}_j$ using a similarity matrix $\mathbf{W}$. The similarity matrix is trained by using a pairwise relevance measure $r : \mathbb{R}^{n \cdot l} \times \mathbb{R}^{n \cdot l} \rightarrow \mathbb{R}$, designed to evaluate how relevant the two feature vectors are. Note that n and l denotes the number of features and the number of bins, respectively (both are defined in Section 3). The benefit of this approach lies in the fact that the algorithm requires only a limited number of labeled samples. The samples are labeled in a way to expresses relation, whether one pair of feature vectors is more relevant than the other. The relevance measure should satisfy the following conditions:

1. Let $\mathbf{h}_i, \mathbf{h}_j$ be two feature vectors from category y_m and $\mathbf{h}_k$ be from a different category y_n (or is unlabeled). Then $r(\mathbf{h}_i, \mathbf{h}_j) > r(\mathbf{h}_i, \mathbf{h}_k)$.

2. Let $\mathbf{h}_i, \mathbf{h}_j$ be two feature vectors from category y_m and $\mathbf{h}_k, \mathbf{h}_l$ be from different categories y_{n1} and y_{n2} (or are not labeled). Then $r(\mathbf{h}_i, \mathbf{h}_j) > r(\mathbf{h}_k, \mathbf{h}_l)$.

The first condition defines the basic requirement to consider two bags from the same category more relevant than two bags from different categories. The second condition ensures that two bags from the same category are more relevant to each other than two unlabeled bags. The training is done by using the passive-aggressive algorithm [6] OASIS [4] originally designed for recognizing similar images. The algorithm iteratively adjusts the weights of the similarity matrix to best fit the previous as well as the new training samples (see Algorithm 1). In [4] it has been shown that the algorithm converges fast with relatively small number of training pairs.

The algorithm finds a bilinear form $\mathbf{W}$ for which:

$$\mathbf{h}_i \, \mathbf{W} \, \mathbf{h}_j > \mathbf{h}_i \, \mathbf{W} \, \mathbf{h}_k + 1,$$

where $\mathbf{h}_i, \mathbf{h}_j$, and $\mathbf{h}_k$ are three feature vectors from the first condition mentioned earlier in this Section. In case of a hinge loss function defined as:

$$l_{\mathbf{W}}(\mathbf{h}_i \, \mathbf{h}_j, \mathbf{h}_k) = \max\{0, 1 - \mathbf{h}_i \, \mathbf{W} \, \mathbf{h}_j + \mathbf{h}_i \, \mathbf{W} \, \mathbf{h}_k\},$$

the goal is to minimize a global loss $L_{\mathbf{W}}$ over all possible triples:

$$L_{\mathbf{W}} = \sum_{i,j,k} l_{\mathbf{W}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k).$$

To minimize the global loss $L_{\mathbf{W}}$, a passive-aggressive algorithm is applied to optimize $\mathbf{W}$ over all feature vectors. The algorithm starts with the initial similarity matrix $\mathbf{W} = \mathbf{I}$ (identity matrix). In this case, the similarity is a simple dot product of the two feature vectors $\mathbf{h}_i^T \mathbf{I} \mathbf{h}_j = \mathbf{h}_i^T \cdot \mathbf{h}_j$. The algorithm then iterates over the training samples to adjust the similarity matrix $\mathbf{W}$ to satisfy the conditions (1) and (2) defined above. In each step, the algorithm randomly selects a pair of feature vectors from the same category and one feature vector from a different category (or an unlabeled bag). The purpose of each iteration is to optimize a trade-off between $\mathbf{W}$ computed so far and the current loss $l_{\mathbf{W}}$. More specifically, the algorithm solves the following convex problem with soft margin:

$$\begin{aligned} \mathbf{W}^i &= \arg\min_{\mathbf{W}} \frac{1}{2} \|\mathbf{W} - \mathbf{W}^{i-1}\|_{Fro}^2 + C\xi \qquad (7) \\ \text{s.t.} \quad & l_{\mathbf{W}}(\mathbf{h}_i, \mathbf{h}_j, \mathbf{h}_k) \leq \xi \quad \text{and} \quad \xi \geq 0, \end{aligned}$$

where $\|.\|_{Fro}$ is the Frobenius norm and the parameter C controls the trade-off. The solution of the optimization problem [4] from Equation 7 is described in Algorithm 1. The training ends after a predefined number of iterations or when the similarity between the training pairs is below a given threshold.

In the testing phase, the similarity is used to create clusters of similar feature vectors, where all vectors from one cluster belong to the same category. As the last stage of the training procedure, the algorithm computes centroids $\mathbf{c}_i$ of the

Algorithm 1 Training similarity matrix

function TRAINSIMILARITYMATRIX
  $\mathbf{W}^0 = \mathbf{I}$
  repeat
    sample three feature vectors:
      $F = \mathbf{h}_i, F_+ = \mathbf{h}_j, F_- = \mathbf{h}_k$
      such that $r(F, F_+) > r(F, F_-)$
    $\mathbf{V}^i = [F^{(1)}(F_+ - F_-), \ldots, F^{(N)}(F_+ - F_-)]^T$,
      where $F^{(i)}$ denotes i-th component of F
    $l_{W^{i-1}}(F, F_+, F_-)$
      $= \max\{0, 1 - FW^{i-1}F_+ + FW^{i-1}F_-\}$
    $\tau_i = \min\{C, \frac{l_{W^{i-1}}(F,F_+,F_-)}{\|\mathbf{V}^i\|^2}\}$,
      where C is aggressiveness parameter
    $\mathbf{W}^i = \mathbf{W}^{i-1} + \tau_i \mathbf{V}^i$
  until (stopping criterion)
  return W
end function

clusters C_i and threshold t. The threshold t is computed as an average similarity of a centroid with the rest of the vectors within a cluster. This is calculated across all clusters as follows:

$$t = \frac{\sum_{i,j} \mathbf{c}_i^T \, \mathbf{W} \, \mathbf{h}_j^{(i)}}{\text{number of all feature vectors } \mathbf{h}_j^{(i)}}, \tag{8}$$

where $\mathbf{h}_j^{(i)}$ denotes that j-th feature vector from i-th cluster. In case of a vector not similar to any of the existing centroids (the similarity is below the threshold t), this vector will create a new centroid and thus a new category.

5 Application in Network Security

We applied the combination of the proposed representation with the similarity learning to classify unseen malware bags in network security domain. The next section provides specification of the datasets, followed by the results from the experimental evaluation.

5.1 Specification of the Datasets

This section provides detailed description of the datasets and features used in the experimental evaluation. The datasets are divided into two disjoint parts: training, and testing. Both datasets were obtained from 1 month of real network traffic of 80 international companies (more than 500,000 users) in form of proxy logs. These logs contain HTTP/HTTPS flows, where one flow represents one communication between a user and a server. More specifically, one flow is a

Fig. 3: URL decomposition into seven parts.

Features	Features applied on all URL parts + referer
duration	length
HTTP status	digit ratio
is URL encrypted	lower case ratio
is protocol HTTPS	upper case ratio
number of bytes up	vowel changes ratio
number of bytes down	has repetition of '&' and '='
is URL in ASCII	starts with number
client port number	number of non-base64 characters
server port number	has a special character
user agent length	max length of consonant stream
MIME-Type length	max length of vowel stream
number of '/' in path	max length of lower case stream
number of '/' in query	max length of upper case stream
number of '/' in referer	max length of digit stream
is second-level domain rawIP	ratio of a character with max occurrence

Table 1: List of features extracted from proxy logs. Features from the right column are applied on all URL parts.

group of packets with the same source and destination IP address, source and destination port, and protocol. As flows from the proxy logs are bidirectional, both directions of a communication are included in each flow.

A flow consists of the following fields: user name, source IP address, destination IP address, source port, destination port, protocol, number of bytes transferred from client to server and from server to client, flow duration, timestamp, user agent, URL, referer, MIME-Type, and HTTP status. The most informative field is URL, which can be decomposed further into 7 parts as illustrated in Figure 3. We extracted 317 features from the flow fields (see the list in Table 1). Features from the right column are applied on all URL parts, including the URL itself and the referer.

Flows are grouped into bags, where each bag contains flows with the same user (or source IP) and the same second-level domain. Thus, each bag represents communication of a user with a particular domain. The size of a bag is at least 5 flows to be able to compute a representative histogram from feature values. As the datasets were originally unlabeled, we used available blacklists and other malware feeds from Collective Intelligence Framework (CIF) [9] to add positive labels to the training dataset. All bags with domains marked as malicious by CIF (or by other external tools) were labeled as positive.

Malware Category	Samples	
	Flows	Bags
C&C malware	30,105	532
DGA malware	3,772	105
DGA exfiltration	1,233	70
Click fraud	9,434	304
Trojans	1,230	12
Background	867,438	15,000

Table 2: Number of flows and bags of malware categories and background traffic.

Representation	Training Data					Test Data				
	TP	FP	TN	precision	recall	TP	FP	TN	precision	recall
baseline	304	0	6976	1.0	1.0	584	13	7987	0.998	0.81
self-similarity	304	0	6976	1.0	1.0	**633**	**6**	**7994**	**0.999**	**0.88**

Table 3: Summary of the SVM results from the baseline and the proposed representation. Both classifiers have the same results on the training set, however SVM classifier where the bags were represented with the proposed self-similarity approach achieved better performance on the test data.

There are 5 malware categories: malware with command & control channels (marked as C&C), malware with domain generation algorithm (marked as DGA), DGA exfiltration, click fraud, and trojans. The summary of malicious categories is shown in Table 2. The rest of the background traffic is considered as legitimate.

5.2 Experimental Evaluation

This section shows the benefits of the proposed representation for a two-class and a multi-class classification problem in network security. The feature vectors described in Section 5.1 correspond to input feature vectors $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$ defined in Section 2. These vectors were transformed to the proposed histogram representation $\{\mathbf{h}^1, \ldots, \mathbf{h}^n\}$, as described in Section 3. Each histogram $\mathbf{h}^i$ had 32 bins ($l = 32$). The proposed approach was compared with a baseline representation, where each bag is represented as a joint histogram of the input feature values $\{\mathbf{x}_1, \ldots, \mathbf{x}_m\}$. This means that one histogram was computed from values of every feature and bag, and the histograms were then concatenated to one final feature vector for each bag. Note that the baseline representation differs from the proposed representation in the fact that the baseline does not compute histograms from self-similarity matrices, but directly from the input feature values. Comparing these two approaches will show the importance of the self-similarity matrix, when dealing with domain adaptation problems.

First, a two-class SVM classifier was evaluated on both representations. To demonstrate the conditional shift of positive bags, only click fraud bags were used

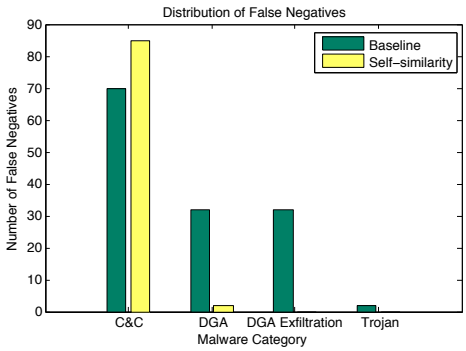
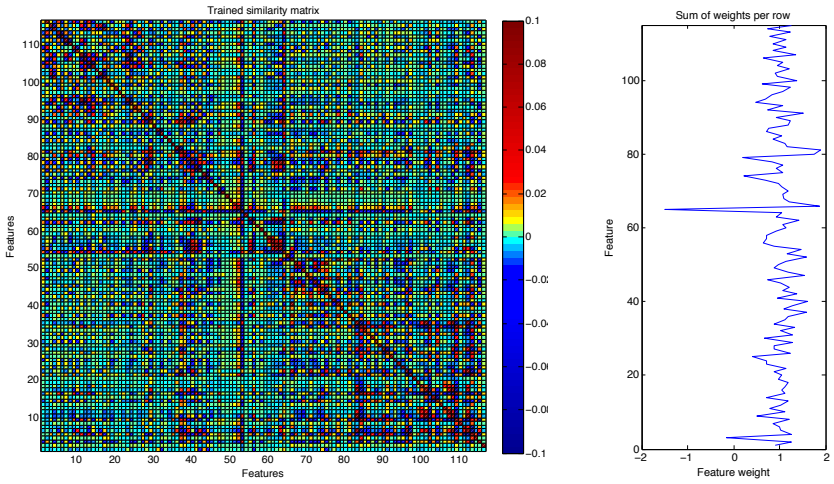


Fig. 4: Analysis of false negatives for both approaches. Thanks to the proposed self-similarity representation, SVM classifier was able to correctly classify all DGA exfiltration, trojan, and most of DGA malware bags, with a slight increase of false negatives for C&C.



(a) (b)

Fig. 5: Graphical illustration of a similarity submatrix W trained according to Algorithm 1 (a) and the corresponding sum of weights for each row (b). The matrix can also serve for feature selection, as some features have a negligible weight.

in the training set as positive bags. A total of 6976 negative bags were included in the training set. The SVM classifier was evaluated on bags from C&C and DGA malware, DGA exfiltration, trojans, and 8000 negative background bags. The results are shown in Table 3. Both classifiers have the same results on the training set, however the SVM classifier using the data represented with the proposed self-similarity approach achieved better performance on the test data.

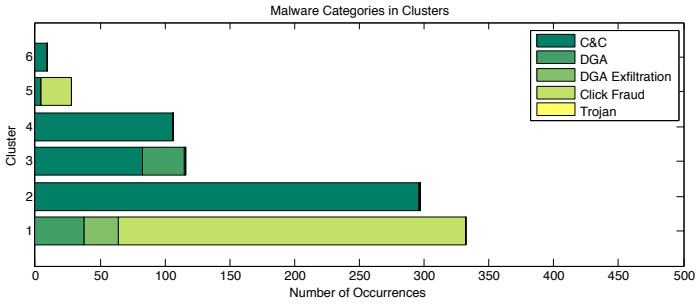


Fig. 6: Distribution of malware categories in clusters with more than 5 bags. Input bags are represented with the baseline approach. C&C bags are scatted across more clusters, and trojan malware bags were not clustered at all.

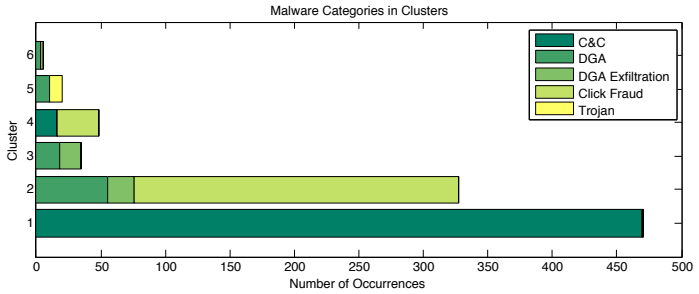


Fig. 7: Distribution of malware categories in clusters with more than 5 bags. Input bags are represented with the proposed approach. Most C&C bags were placed into a single cluster. Trojan bags were successfully found in cluster 5.

More detailed analysis of false negatives for both approaches in provided in Figure 4. Thanks to the proposed self-similarity representation, the SVM classifier was able to correctly classify all DGA exfiltration, trojan, and most of DGA malware bags. There is only a slight increase in the number of false negatives for C&C. Overall, the proposed self-similarity representation shows better robustness than the baseline approach.

Next, the performance on a multi-class problem with missing labels is evaluated with the similarity learning algorithm described in Section 4. Two malware categories were included in the training set (click fraud and C&C) together with 5000 negative bags. Similarity matrix W, trained according to the Algorithm 1, is depicted in Figure 5.

In the next experiment, similarity matrix W was used to create an adjacency matrix of all bags in the test set, where i, j-th component of this matrix is computed as $\mathbf{h}_i^T \mathbf{W} \mathbf{h}_j$. This means that i, j-th component expresses the distance between i-th and j-th bag in a metric space defined by the learned similarity matrix W. Modularity clustering [3] was used to cluster the bags according to the adjacency matrix. The distribution of categories in malicious clusters with

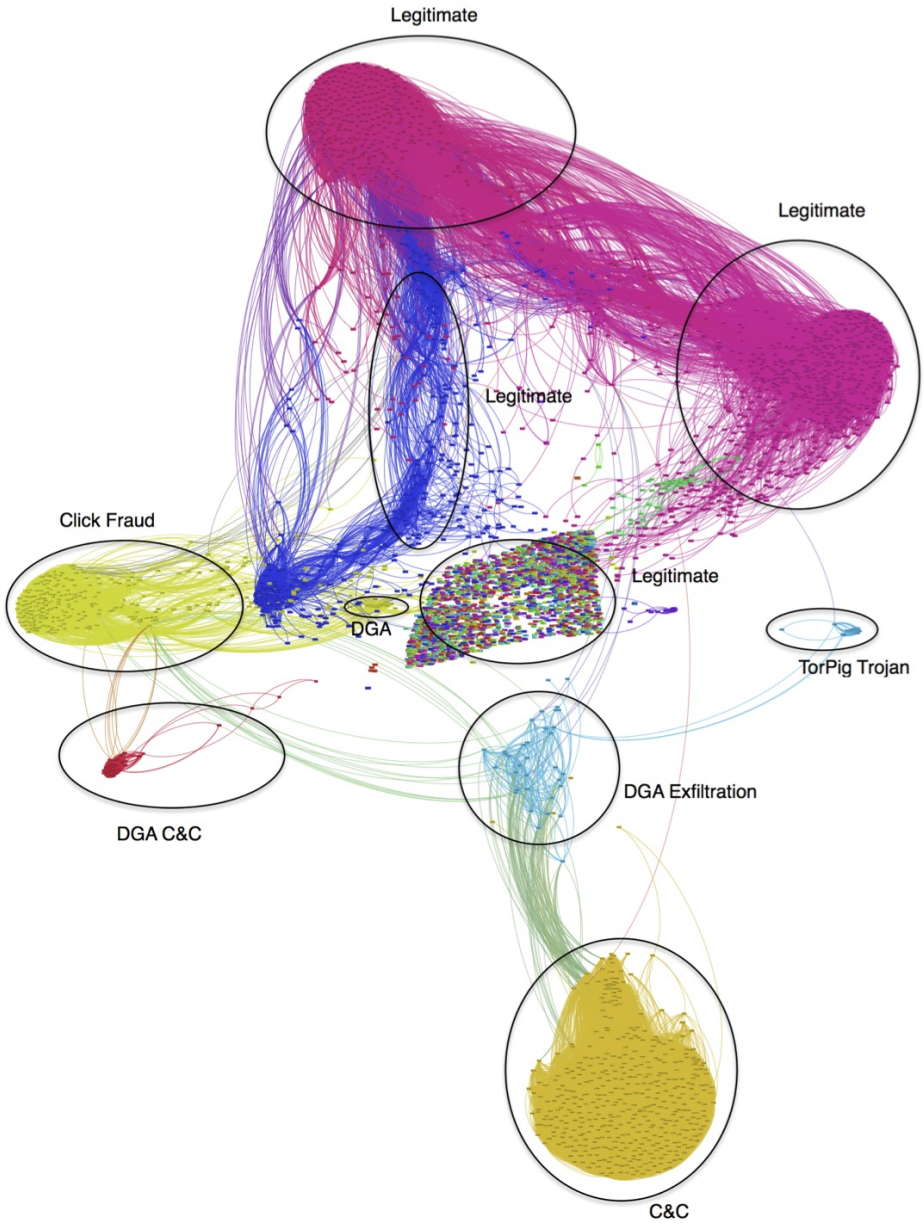


Fig. 8: Graphical illustration of the clustering results, where the input bags were represented with the proposed representation. Legitimate bags are concentrated in three large clusters on the top and in a group of non-clustered bags located in the center. Malicious bags were clustered into six clusters.

more than 5 bags is depicted in Figure 6 (for the baseline representation) and in Figure 7 (for the proposed representation). In contrast to the baseline results, most C&C bags are concentrated in a single cluster. Moreover, trojan bags were successfully found (in cluster 5) as opposed to the baseline. The overall clustering results are illustrated in Figure 8. The legitimate bags are concentrated in three large clusters on the top and in a group of non-clustered bags located in the center, while the malicious bags were clustered to six clusters.

6 Conclusion

This paper proposed a robust representation of bags of samples suitable for the domain adaptation problems with conditional shift. Under conditional shift, the probability distributions of the observations given labels is different in the training (source) and testing (target) data which complicates standard supervised learning algorithms. The new representation is designed to be invariant under common changes between the source and target data, namely shifting and scaling of the feature values and permutation and size changes of the bags. This is achieved by computing a self-similarity measure of the bags using sample features. The representation is used in online similarity learning which results in a robust algorithm for multi-class classification with missing labels.

The proposed representation was evaluated and compared with the baseline representation without adaptation in two network security use cases. First, in a binary classification of malicious network traffic, the new invariant representation improved the recall of an SVM classifier from 0.81 to 0.88 and the precision from 0.998 to 0.999. Second, in a modularity clustering of network traffic, the proposed approach correctly grouped malware according to their categories and even identified a new category, previously unseen in the training data. These results demonstrate the invariant properties of the representation which make it useful in network security.

There are several remaining challenges in the domain adaptation for network security. With constantly evolving malware, conditional shift might still occur even when the new malware families are represented as outlined in this paper. There are other types of malware, some of which have not been identified or fully understood, that have different behavioral patterns making it impossible to transfer knowledge from the source to the target domain. Some of these challenges might be solved by introducing nonlinearity to the malware similarity measure. As in the presented online similarity learning, the measure could use the known samples to learn the differences between malicious and legitimate traffic. This is the direction of our future research.

References

1. S. Ben-David, J. Blitzer, K. Crammer, F. Pereira, et al. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19:137, 2007.

2. J. Blitzer, R. McDonald, and F. Pereira. Domain adaptation with structural correspondence learning. In *Proceedings of the 2006 conference on empirical methods in natural language processing*, pages 120–128. Association for Computational Linguistics, 2006.
3. U. Brandes, D. Delling, M. Gaertler, R. Gorke, M. Hoefer, Z. Nikoloski, and D. Wagner. On modularity clustering. *Knowledge and Data Engineering, IEEE Transactions on*, 20(2):172–188, 2008.
4. G. Chechik, V. Sharma, U. Shalit, and S. Bengio. Large scale online learning of image similarity through ranking. *The Journal of Machine Learning Research*, 11:1109–1135, Mar. 2010.
5. Y. Chen, E. K. Garcia, M. R. Gupta, A. Rahimi, and L. Cazzanti. Similarity-based classification: Concepts and algorithms. *The Journal of Machine Learning Research*, 10:747–776, 2009.
6. K. Crammer, O. Dekel, J. Keshet, S. Shalev-Shwartz, and Y. Singer. Online passive-aggressive algorithms. *The Journal of Machine Learning Research*, 7:551–585, Dec. 2006.
7. W. Dai, Q. Yang, G.-R. Xue, and Y. Yu. Boosting for transfer learning. In *Proceedings of the 24th international conference on Machine learning*, pages 193–200. ACM, 2007.
8. L. Duan, I. W. Tsang, and D. Xu. Domain transfer multiple kernel learning. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(3), 2012.
9. G. Farnham and K. Leune. Tools and standards for cyber threat intelligence projects. Technical report, SANS Institute InfoSec Reading Room, 10 2013.
10. A. Gretton, A. Smola, J. Huang, M. Schmittfull, K. Borgwardt, and B. Schölkopf. Covariate shift by kernel mean matching. *Dataset shift in machine learning*, 2009.
11. A. Iyer, S. Nath, and S. Sarawagi. Maximum mean discrepancy for class ratio estimation: Convergence bounds and kernel selection. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 530–538, 2014.
12. I. N. Junejo, E. Dexter, I. Laptev, and P. Perez. View-independent action recognition from temporal self-similarities. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 33(1):172–185, 2011.
13. M. Krner and J. Denzler. Temporal self-similarity for appearance-based action recognition in multi-view setups. In R. Wilson, E. Hancock, A. Bors, and W. Smith, editors, *Computer Analysis of Images and Patterns*, volume 8047 of *Lecture Notes in Computer Science*, pages 163–171. Springer Berlin Heidelberg, 2013.
14. M. Müller and M. Clausen. Transposition-invariant self-similarity matrices. In *In Proceedings of the 8th International Conference on Music Information Retrieval (ISMIR)*, pages 47–50, 2007.
15. M. Palatucci, D. Pomerleau, G. E. Hinton, and T. M. Mitchell. Zero-shot learning with semantic output codes. In *Advances in neural information processing systems (NIPS)*, pages 1410–1418, 2009.
16. H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2):227–244, 2000.
17. R. Socher, M. Ganjoo, C. D. Manning, and A. Ng. Zero-shot learning through cross-modal transfer. In *Advances in Neural Information Processing Systems*, pages 935–943, 2013.
18. K. Zhang, B. Schölkopf, K. Muandet, and Z. Wang. Domain adaptation under target and conditional shift. In S. Dasgupta and D. Mcallester, editors, *Proceedings of the 30th International Conference on Machine Learning (ICML-13)*, volume 28, pages 819–827. JMLR Workshop and Conference Proceedings, 2013.