Loss Functions for Clustering in Multi-instance Learning

Marek Dědič[1,2], Tomáš Pevný[3], Lukáš Bajer[2], Martin Holeňa[4]

[1] Faculty of Nuclear Sciences and Physical Engineering, Czech Technical University in Prague, Trojanova 13, Prague, Czech Republic
[2] Cisco Systems, Inc., Karlovo náměstí 10, Prague, Czech Republic
[3] Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo náměstí 13, Prague, Czech Republic
[4] Institute of Computer Science, Czech Academy of Sciences, Pod vodárenskou věží 2, Prague, Czech Republic

Abstract: Multi-instance learning belongs to one of recently fast developing areas of machine learning. It is a supervised learning method and this paper reports research into its unsupervised counterpart, multi-instance clustering. Whereas traditional clustering clusters points, multi-instance clustering clusters bags, i.e. multisets of points or of other kinds of objects. The paper focuses on the problem of loss functions for clustering. Three sophisticated loss functions used for clustering of points, contrastive predictive coding, triplet loss and magnet loss, are elaborated for multi-instance clustering. Finally, they are compared on 18 benchmark datasets, as well as on a real-world dataset.

1 Introduction

Multi-instance learning (MIL) belongs to recently fast developing areas of machine learning. Though it is a supervised learning method, this paper addresses its application to unsupervised learning – clustering. Whereas traditional clustering is one of points, multi-instance clustering clusters multisets of points, also known as bags. Such a grouping of the points into bags is considered a property of the problem at hand and therefore sourced from the input data.

While there have been some previous attempts at multi-instance clustering, they use pairwise relations between all points in all bags, quickly becoming unwieldy and computationally infeasible. Our work uses multi-instance learning as a general toolkit for learning representations of bags and then clusters the bags using those representations. This builds on previous works by the authors and enables clustering of arbitrary data-structures by expressing them as hierarchies of multi-instance problems and then using the internal structure to better solve the problems at hand.

Multi-instance clustering is evaluated in Section 4 in the application domain of computer and network security. While this is only one of many possible applications of MIL due to its general expressive power for structured data, it is the domain of choice for the authors. Here, MIL is used to represent user activity as a bag of network connections for each user in a fixed time window. For clustering, this enables detection of compromised users based on their complex behaviours. Clustering opens a new window of opportunity here by e.g. grouping servers with similar behaviour together, allowing a human analyst to boost their work significantly by deciding about whole clusters of servers instead of each one individually.

A key prerequisite for the application of multi-instance learning to clustering is that loss functions for clustering, originally proposed for points, are adapted for bags. In this paper, such an adaptation is outlined for three sophisticated loss functions: contrastive predictive coding, triplet loss and magnet loss.

Of the three proposed methods, triplet loss and magnet loss utilize some labels as part of the training process and so are not truly unsupervised. As such, the authors expect them to outperform the method based on contrastive predictive coding, which on the other hand is the most promising and innovative approach from the theoretical point of view.

In the next section, basic properties of multi-instance learning and clustering are briefly introduced. The adaptation of the three considered loss functions is explained in Section 3. A comprehensive comparison of them is then presented in Section 4.

2 Multi-instance Learning and Clustering

Multi-instance learning (MIL) was first described by [7]. In its original form, it was developed and used for **supervised learning**. Some prior art also exists for **unsupervised learning**, such as [5, 31], however, it uses pairwise instance distances as a basis for clustering, which doesn't properly utilize the inherent structure of the data and quickly becomes computationally infeasible.

The MIL paradigm is a type of representation learning on data which has some internal structure. Therefore, it views a sample as a **bag** (i.e. a multiset) of an arbitrary number of objects. The basic elements of MIL are samples from a space $\mathcal{X}$ and their corresponding labels from a space $\mathcal{Y}$ (a space of classes). Compared to usual supervised learning, MIL replaces individual instances with bags of instances from the space $\mathcal{X}$ such that every instance in $\mathcal{X}$ belongs to at least one bag from the bag-space $\mathcal{B}$.

[7] provides an example of a multi-instance problem where each bag represents a key chain with some keys (instances). To solve the problem of finding which key opens a particular lock, a "proxy" MIL problem is presented – determining which key chain opens that particular lock. This line of thinking leads to the pivotal definition of the label of a bag as being positive iff the bag contains at least one

positive instance. In later works such as [23], this interpretation of MIL is abandoned in favor of a more general one. An instance is no longer viewed as having a meaning in and of itself, but only in the context of its bag. The notion of an instance-level label is dropped because in this interpretation, the bag is the atomic unit of interest. To this end, the embedded-space paradigm, described in Section 2.2, is used.

2.1 A probabilistic formulation of multi-instance learning

A probabilistic way of describing multi-instance learning was first introduced in [23] and builds on the previous work [19].

Let for the space $\mathcal{X}$ exist a measurable space $(\mathcal{X}, \mathcal{A})$, where $\mathcal{A}$ is a σ-algebra on $\mathcal{X}$. Let $\mathcal{P}^{\mathcal{X}}$ denote the set of all probability measures on $(\mathcal{X}, \mathcal{A})$. A bag B is viewed as a random sample with replacement of a random variable governed by a particular probability distribution $p_B \in \mathcal{P}^{\mathcal{X}}$, that is

$$B = \{x_i | x_i \sim p_B, i \in \{1, \dots, m\}\} \text{ where } m \in \mathbb{N}. \quad (1)$$

2.2 Embedded-space paradigm for solving multi-instance problems

While it is possible to use several approaches to solving multi-instance problems, in this work, the embedded-space paradigm was used. For an overview of the other paradigms, see [6].

In the embedded space paradigm, labels are only defined on the level of bags. In order for these bag labels to be learned, an embedding function of the form $\phi : \mathcal{B} \to \bar{\mathcal{X}}$ must be defined, where $\bar{\mathcal{X}}$ is a latent space, which may or may not be identical to $\mathcal{X}$. Using this function, each bag can be represented by an object $\phi(B) \in \bar{\mathcal{X}}$, which makes it possible to use any off-the-shelf supervised learning algorithm acting on $\bar{\mathcal{X}}$. Among the simplest embedding functions are, e.g. element-wise minimum, maximum, and mean. A more complicated embedding function may for example apply a neural network to each instance of the bag and subsequently pool the instances using one of the aforementioned functions.

Specifically to the experiments presented in Section 4, [22] present an approach to learning to classify HTTP traffic by utilizing sets of URLs. Neural networks are used to transform both instance-level and bag-level representations. The model is as follows: A neural network $f_I : \mathcal{X}_1 \to \mathcal{X}_2$ is used to transform instances, followed by an aggregation function

$$g : \mathcal{B}_{\mathcal{X}_2} \to \bar{\mathcal{X}}_1.$$

Finally, a second deep neural network $f_B : \bar{\mathcal{X}}_1 \to \bar{\mathcal{X}}_2$ is used to transform the representation of each bag. Combining these functions gives the embedding function

$$\phi(B) = f_B\left(g\left(\{f_I(\boldsymbol{x}) | \boldsymbol{x} \in B\}\right)\right).$$

The functions f_I and f_B are realized by a deep neural network using ReLU as its activation function. The aggregation g is realized as an element-wise mean or maximum of all the vectors.

The structure of the data is highly exploited using the embedded-space paradigm. The approach uses multiple layers of MIL nested in each other – the instances of a bag do not necessarily need to be feature vectors, but can be bag in themselves. The HTTP traffic of a particular client is therefore represented as a bag of all second-level domains the client has exchanged datagrams with. Each second-level domain is then represented as a bag of individual URLs which the client has connected to. Individual URLs are then split into 3 parts, domain, path and query, and each part is represented as a bag of tokens, which can then be broken down even further. In the end, the model consists of 5 nested MIL problems.

A benefit of MIL is also its easy use for explainability and interpretability. The authors of [22] present a way to extract indicators of compromise and explain the decision using the learned MIL model.

2.3 Clustering

Clustering is a prime example of a problem typically associated with unsupervised learning. The problem at hand, however, is not one of clustering ordinary number vectors in a linear space. Instead, a clustering of objects represented by bags is explored, as that is the problem solved for datasets introduced later in Section 4. While [28] present a clustering of bags using a modified Hausdorff distance on bags and [17] present a clustering using maximum mean discrepancy, a different approach to clustering of bags is used in this work. Among the main reasons for this choice is the prohibitively high computational complexity of the previously mentioned approaches on large datasets and the possibility of utilizing the representations previously introduced in [22].

An approach based on the embedded-space paradigm for MIL was chosen. In order to utilize the structure of the data, a MIL model is used to represent each bag in the latent space $\bar{\mathcal{X}}$. This presents the issue of how to train the embedding function ϕ, because its learning in standard MIL is supervised.

The embedding function ϕ can be actually written as $\phi = \phi(B, \boldsymbol{\theta})$ where $B \in \mathcal{B}$ and $\boldsymbol{\theta}$ are the parameters of the embedding, typically learned during the training phase. In the context of clustering, these parameters still need to be learned over some kind of training. This itself presents another challenge though – off-the-shelf algorithms typically work in some constant Hilbert space, whereas the latent space needs to change over the learning period. As is the case for MIL itself, end-to-end learning is used. A cluster-loss function $L_C : \mathcal{B} \to \mathbb{R}$ is chosen and used to express the actual loss for the embedding model ϕ and its parameters $\boldsymbol{\theta}$ as

$$L(\phi, \boldsymbol{\theta}) = L_C\left(\{\phi(B, \boldsymbol{\theta}) | B \in \mathcal{B}\}\right). \quad (2)$$

If the cluster-loss L_C is chosen correctly, minimizing L over the learning period will yield a latent space $\bar{\mathcal{X}}$ in which the bags are already naturally clustered according to the design of the cluster-loss function. Applying any off-the-shelf clustering algorithm on $\bar{\mathcal{X}}$ will then give good results. How to choose the cluster-loss function L_C is the focus of Section 3.

2.4 Clustering evaluation metrics

In our research, several clustering evaluation metrics have been used. The primary metrics are based on the Silhouette coefficient, the secondary ones on the kNN algorithm. The metrics are somewhat different to the ones traditionally used in evaluating clustering methods as all the datasets used in this work are originally classification datasets and therefore have classes available. Each class can then be viewed as a cluster target and the learned clustering evaluated against these targets.

The first two metrics measure the **homogeneity** of clusters (that is the property of instances of one cluster being close to one another) and their **separation** (that is the property of instances of different clusters being far apart), see [9]. To measure the non-homogeneity of a cluster, the average distance between items in a cluster is measured and averaged over all clusters, giving the following metric for items x_j in clusters C_i:

$$\text{nonhomo}\left(C_1, \dots, C_n\right) = \frac{1}{n} \sum_{i=1}^{n} \sum_{x_j \in C_i} \left\| x_j - \mu\left(C_i\right) \right\|,$$

where

$$\mu\left(C_i\right) = \frac{1}{\left|C_i\right|} \sum_{x_j \in C_i} x_j,$$

with $|C_i|$ being the number of instances in the bag C_i.

To measure the separation of clusters, the average distance between cluster centres was taken:

$$\text{sep}\left(C_1, \dots, C_n\right) = \frac{2}{n\left(n-1\right)} \sum_{\substack{i,j \in \hat{n} \\ i<j}} \left\| \mu\left(C_i\right) - \mu\left(C_j\right) \right\|.$$

The final primary metric is the Silhouette coefficient:

$$\text{silhouette}\left(C_1, \dots, C_n\right) = \frac{\text{sep}\left(C_1, \dots, C_n\right)}{\text{nonhomo}\left(C_1, \dots, C_n\right)}. \quad (3)$$

For the secondary metrics, the kNN algorithm in the representation space is used and its accuracy measured. While the primary metric measures the quality of the clustering in general, this metric focuses on the utility of the learned representation to a specific and useful task - classification. The training data of the kNN classifier is chosen randomly and is in this work referred to as **seed data** to avoid confusion with the data used to train the embedding. The value of $k = 3$ was chosen by experimental evaluation. This gives an insight into the robustness of the clustering, as high-quality embeddings would need only a small amount of seed data points to reach relatively high accuracy. For this measure, the kNN algorithm was run thrice and the results averaged to compensate for high dependence on the particular seed points chosen.

3 Investigated Loss Functions

In Section 2.4, a method for learning a representation ϕ was described. In order to learn the parameters of the representation, a clustering-loss function is required in the form

$$L_C : \mathcal{P}^M\left(\bar{\mathcal{X}}\right) \rightarrow \mathbb{R}.$$

In this section, three ways of constructing such a clustering-loss function are explored. First, an unsupervised method constructing a clustering loss-function is presented, followed by two supervised methods.

3.1 Contrastive Predictive Coding

Contrastive predictive coding (**CPC**) is a technique first introduced by [21]. The method builds the model on the ideas from predictive coding [8]. CPC represents time series by modelling future data from the past. To this end, the model learns high-level representations of the data and discards noise. The further in the future the model predicts, the less shared information is available and thus the global structure needs to be inferred better.

The core concept taken from CPC is a loss function called InfoNCE in the prior art. Given a time series of values x_t, an autoregressive model produces a context c_t from all x_i up to the point t. For a training set $\mathbb{X}$, predicted future value x_{t+k}, the current context c_t, and a model f_k, the InfoNCE loss is defined as

$$-\mathbb{E}_{\mathbb{X}}\left[\log f_k\left(x_{t+k}, c_t\right) - \log \sum_{x_j \in \mathbb{X}} f_k\left(x_j, c_t\right)\right].$$

Optimizing this value will result in f_k modelling the density ratio

$$f_k\left(x_{t+k}, c_t\right) \propto \frac{p\left(x_{t+k} | c_t\right)}{p\left(x_{t+k}\right)}$$

and therefore in maximizing the mutual information between x_{t+k} and c_t.

The actual application to MIL is based on the following idea. If a bag is split into two parts, it is reasonable to expect that the representations of these two parts would be close to one another. On the other hand, if a random bag were to be drawn from the data (as a *simple random sample with replacement*), it is reasonable to expect it to be relatively far from any actual bag present in the data. These assumptions can be proven correct by using the probabilistic formalism for MIL (see Subsection 2.1). Given that each bag B_k is viewed as a set of realizations of a probability distribution $P_k \in \mathcal{P}^{\mathcal{X}}$, it follows that the two parts of the bag, $B_k^{(1)}$ and $B_k^{(2)}$, are sets of realizations from the same distribution P_k and therefore should be statistically indistinguishable. On the other hand, a randomly sampled bag B'_j does not share the same probability distribution. Using these assumptions, the following clustering loss is constructed:

$$L_{\text{CPC}} = \log \left\| \phi\left(B_k^{(1)}\right) - \phi\left(B_k^{(2)}\right) \right\|^2 - \log \sum_{j=1}^{K} \left\| \phi\left(B_k^{(1)}\right) - \phi\left(B'_j\right) \right\|^2 .$$

where the value $K \in \mathbb{N}$ is a hyper-parameter of this method. The first term of the loss function depicts the notion that the representations of the two parts of the bag should be close to one another and corresponds to the first term of InfoNCE, which maximizes the prediction of a matching future sample from the current context. The second term depicts the notion that a random bag should be far from all the bags[1] and corresponds to the second term of InfoNCE, which minimizes the prediction of a random sample from the current context. Choosing to only use the first part of the bag in the second term has no effect as the two parts are chosen randomly.

This method can be further modified to make it less computationally complex. In order not to have to draw a lot of random bags, it can be reasonably expected that, on average, the representations of two parts of two mismatched bags $B_{k_1}^{(1)}$ and $B_{k_2}^{(2)}$ should be far apart. A matrix $\mathbf{D}$ is constructed as

$$\mathbf{D}_{ij} = \left\| \phi\left(B_i^{(1)}\right) - \phi\left(B_j^{(2)}\right) \right\|^2 . \tag{4}$$

The distances of the corresponding halves are found on the diagonal of $\mathbf{D}$, whereas the distances of mismatched halves are in the rest of the matrix. Under this assumption the final loss for the CPC method is

$$L_{\text{CPC}} = \frac{1}{n} \sum_{i=1}^{n} \left(\log\left(\mathbf{D}_{ii}\right) - \log \sum_{\substack{j=1 \\ j \neq i}}^{n} \mathbf{D}_{ij} \right) , \tag{5}$$

where n is the number of bags.

Out of the three methods presented in this section, the approach based on contrastive predictive coding seems the most promising, as it is the only unsupervised method and could therefore be used in a broader class of applications. This, however, also has its drawbacks – unsupervised methods are generally harder to learn correctly. For that reason, the two supervised methods were selected as a safer option.

3.2 Triplet Loss

The performance of the k-nearest neighbour algorithm depends heavily on the distance metric used. Typically, the Euclidean distance is used. However, ideally, the metric should adapt to the problem at hand. [29] present a way to learn a Mahalanobis distance for kNN such that it has the *homogeneity* and *separation* properties of clustering. This distance, the description of which follows, has been utilized in this work as the basis for the desired loss function L_C.

Let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ be a training dataset with $\boldsymbol{x}_i \in \mathbb{R}^d$ and y_i discrete class labels. In the original work, the goal is to learn a linear transformation

$$\boldsymbol{L} : \mathbb{R}^d \to \mathbb{R}^d.$$

This linear transformation is then used to compute squared distances as

$$\mathcal{D}\left(\boldsymbol{x}_i, \boldsymbol{x}_j\right) = \left\| \boldsymbol{L}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \right\|^2 .$$

A helper matrix $\mathbf{y}$ is used such that

$$\mathbf{y}_{ij} = \begin{cases} 0 & \text{for} \quad y_i \neq y_j \\ 1 & \text{for} \quad y_i = y_j \end{cases} .$$

In addition to this, for each input $\boldsymbol{x}_i$, k target neighbours are defined that are supposed to be close to $\boldsymbol{x}_i$. Euclidean distance may be used to find the target neighbours. A matrix η is used to indicate target neighbours where $\eta_{ij} = 1$ if $\boldsymbol{x}_j$ is a target neighbour of $\boldsymbol{x}_i$ and 0 else.

The cost function features two competing terms. The first term depicts the notion that an input should be close to its target neighbours. The second term depicts the notion that inputs of a different class should be far from one another. The loss is of the form

$$\sum_{i,j=1}^{n} \eta_{ij} \left\| \boldsymbol{L}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \right\|^2 + c \sum_{i,j,l=1}^{n} \eta_{ij} \left(1 - \mathbf{y}_{il}\right) \max\left(0, 1 + \left\| \boldsymbol{L}\left(\boldsymbol{x}_i - \boldsymbol{x}_j\right) \right\|^2 - \left\| \boldsymbol{L}\left(\boldsymbol{x}_i - \boldsymbol{x}_l\right) \right\|^2 \right) ,$$

where $c > 0$ is a hyper-parameter of this method.

To adapt the original method, the definitions need to be modified to leverage the bagging of the data. The matrix $\mathbf{y}$ is defined in the same way, only with labels on the level of bags. The value $\eta_{ij} = 1$ iff the bag B_j is a target neighbour for the bag B_i. Then, using the matrix $\mathbf{D}$ defined in equation 4 directly instead of the linear transformation, the loss function can be expressed as

$$L_{\text{triplet}} = \sum_{ij} \eta_{ij} D_{ij} + \\ + c \sum_{ijl} \eta_{ij} \left(1 - y_{il}\right) \max\left(0, 1 + D_{ij} - D_{il}\right) , \tag{6}$$

where $c > 0$ is again a hyper-parameter of the method. Although [29] suggest finding η as the k-nearest neighbours of a data point, the loss has been simplified by setting $\eta_{ij} = \mathbf{y}_{ij}$ for $i, j = 1, \dots, n$ in this work, making all the neighbours of a cluster its target neighbours.

[1] On the off-chance a random bag would be close to $B_k^{(1)}$, this would be outweighed by the other terms.

3.3 Magnet Loss

Magnet loss is an enhancement of the previously described triplet loss, first introduced by [26]. Whereas the triplet loss function essentially goes over all triplets of a data point, its target neighbour and a point from a different class and optimizes their distances, magnet loss maintains an explicit model of the distributions of different classes in the representation space. To capture the distributions, the model uses simple clustering models for each class. Differently to triplet loss, for which the target neighbourhood is set *a priori* and is based on similarity in the original input space, magnet loss uses similarity in the space of representations and updates it periodically. Magnet loss also pursues local separation instead of global – representations of different classes should be separated, but can be interleaved.

Let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{n}$ be a training dataset where $\boldsymbol{x}_i \in \mathbb{R}^d$ and y_i are one of C discrete class labels. A general parameterized map $\boldsymbol{f}(\cdot;\Theta)$ embeds the inputs into a representation space:

$$\boldsymbol{r}_i = \boldsymbol{f}(\boldsymbol{x}_i;\Theta).$$

For each class c, K cluster assignments are obtained with the K-means++ algorithm [18, 2] where $K \in \mathbb{N}$ is a hyper-parameter of the method:

$$\mathcal{I}_1^c, \dots, \mathcal{I}_K^c = \operatorname*{arg\,min}_{I_1^c, \dots, I_K^c} \sum_{k=1}^{K} \sum_{\boldsymbol{r} \in I_k^c} \left\| \boldsymbol{r} - \frac{1}{|I_k^c|} \sum_{\boldsymbol{s} \in I_k^c} \boldsymbol{s} \right\|^2 .$$

The centre of each cluster is denoted $\boldsymbol{\mu}_k^c$:

$$\boldsymbol{\mu}_k^c = \frac{1}{|\mathcal{I}_k^c|} \sum_{\boldsymbol{r} \in \mathcal{I}_k^c} \boldsymbol{r}.$$

For each input, the centre of the cluster in which its representation falls is denoted as

$$\boldsymbol{\mu}_i = \operatorname*{arg\,min}_{\boldsymbol{\mu}_k^c} \|\boldsymbol{r}_i - \boldsymbol{\mu}_k^c\|$$

and the squared distance of its representation from the centre of the cluster is denoted as

$$N_i = \|\boldsymbol{r}_i - \boldsymbol{\mu}_i\|^2 .$$

The variance of the distance of the representations from their respective centres can then be calculated as

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^{n} N_i .$$

For r_i, the dissimilarity to all the inputs of different classes is calculated as

$$M_i = \sum_{c \neq y_i} \sum_{k=1}^{K} \exp\left(-\frac{1}{2\sigma^2} \|\boldsymbol{r}_i - \boldsymbol{\mu}_k^c\|^2 \right).$$

The magnet loss is then defined as

$$L_{\text{magnet}} = \frac{1}{n} \sum_{i=1}^{n} \max\left(0, 1 + \log \frac{\exp\left(-\frac{1}{2\sigma^2} N_i - \alpha\right)}{M_i} \right), \tag{7}$$

where $\alpha \in \mathbb{R}$ is a hyper-parameter of the method. Since the method standardizes both of the terms of the innermost fraction by σ^2, α is the desired cluster separation gap expressed in the units of the variance.

The magnet loss is taken almost verbatim from the original article. The only change needed is a different way to obtain the representation space by taking

$$\boldsymbol{r}_i = \phi(B_i).$$

In this usage, y_i is the class label assigned to the bag B_i (making magnet loss a supervised method) and n is the number of bags in the dataset.

Due to the choice of the hyper-parameter K being non-obvious for most problems, alternative clustering methods that would not need hyper-parameter tuning were explored. One such method, **self-tuning spectral clustering** [30] was implemented as a replacement for the K-means algorithm used in the original method. Section 4.3 presents a comparison of the performance of these two methods.

Since magnet loss is an enhancement of triplet loss, it is reasonable to assume it would perform the same or better than triplet loss. One drawback of magnet loss is the higher number of hyper-parameters which need to be tuned in order for the method to work correctly. This problem is partially solved by the introduction of self-tuning spectral clustering [30], however, that is outside the scope of this paper.

4 Experimental Comparison

The three loss functions presented in Section 3 were experimentally evaluated on 18 publicly available datasets and on a corporate dataset from the domain of computer security. The 2 metrics presented in Section 2.4 were used, i.e. the Silhouette coefficient and the accuracy of a kNN classifier built on the embedding. Both of these metrics were tracked over the learning period to also evaluate how the models are learning in addition to their final performance.

4.1 Datasets

The evaluation was done on a set of 18 publicly available datasets, listed in Table 1. All the datasets as used were also made public[2].

The models were also evaluated on a proprietary dataset provided by Cisco Cognitive Intelligence, consisting of records of network connections from clients (e.g. user computers or mobile devices) to some on-line services.

[2] https://github.com/marekdedic/MilDatasets.jl

Source	Datasets
[4]	BrownCreeper, WinterWren
[1]	Elephant, Fox, Tiger
[7]	Musk1, Musk2
[27]	Mutagenesis1, Mutagenesis2
[33]	Newsgroups1, Newsgroups2, Newsgroups3
[24, 25]	Protein
[15]	UCSBBreastCancer
[32]	Web1, Web2, Web3, Web4

Table 1: The 18 publicly available datasets used.

The dataset represents HTTP traffic of more than 100 companies. Two datasets were collected, each spanning 1 day of traffic. The training data was traffic from 2019-11-18, while the data used for testing was collected the following day, 2019-11-19. For each connection, a proprietary classification system based on [14] provided labels, classifying the connections either as legitimate or malicious (connected to malware activity). The data was sampled to include 90 % of negative bags and 10 % of positive bags. For each connection, 20 connection features were used, as well as a MIL model of the server URL, which is visualized in Figure 1.

4.2 Experimental Design

The models for evaluation were implemented in the Julia programming language [3] using the Flux.jl framework for machine learning [13] and the Mill.jl framework for multi-instance learning[3].

The particular architecture of the models was chosen based upon previous experience with using neural networks in multi-instance setting [22]. Several architectures were tested and the best one selected for the experiments. The embedding ϕ was realised by a MIL neural network consisting of 2 per-instance layers of 30 neurons, followed by aggregation formed by concatenating element-wise mean and element-wise maximum of all instances in a bag, followed by 2 per-bag layers of 30 neurons. All the neurons used the ReLU activation function [12]. Layer weights were initialized using Glorot initialization [11], bias vectors were initialized to zeros. ADAM [16] was used as the optimization method.

For each of the datasets, 80 % of bags were randomly chosen as the training data, with the rest being testing data. The models were trained using 100 mini-batches of size of 50.

In order to provide some baseline against which the models could be compared (as there is no prior art for this problem), two other models were introduced. A non-machine-learning model was introduced as a baseline, which all models should surpass. This model implements the embedding ϕ as an element-wise mean of all instances of a bag. This is a natural embedding of a bag as its expected value. This model is referred to as the **mean model** in the rest of this work. A classification model has been introduced as a target to beat. This model was realised by a MIL neural network identical to the previously described one, with a final layer of 2 output neurons with an identity activation function added. The accuracy of the model has been evaluated by selecting the optimal threshold on its output. This model mirrors the model used in the proposed methods, but replaces the clustering with simple classification. This model is referred to as the **classification model** in the rest of this work.

Some of the three proposed clustering-losses have some hyper-parameters which need to be tuned. A range of values was tried for each hyper-parameter in order to select the best configuration for each on each of the datasets. For L_{triplet}, the values $c \in \{0.01, 0.1, 1, 10, 100\}$ have been tested. For L_{magnet}, the values $K \in \{2, 3, 8, 16\}$, $\alpha \in \{0, 0.1, 0.5\}$ and the cluster index update frequency in $\{5, 10, 25, 70\}$ have been tested.

4.3 Comparison of Results

Table 2 lists the accuracy of a kNN classifier build on the embedding for all of the methods and all of the datasets. Figure 2 shows the value of the Silhouette coefficient and the accuracy on 3 selected datasets. As can be seen, CPC is the worst performing of the methods overall, and, more significantly, shows no clear improvement over the learning periods for both the Silhouette coefficient and the kNN classifier accuracy. Between triplet loss and magnet loss, magnet loss is somewhat better in the classification accuracy, whereas triplet loss is somewhat better in the Silhouette coefficient, indicating better separation of individual clusters. Of note is also the differing behaviour for various datasets, for example the clear dominance of triplet loss w.r.t. the Silhouette coefficient on the Web3 dataset. This shows that none of the methods is a clear winner in all scenarios and the best one needs to be carefully selected.

4.4 Statistical significance of the results

The null hypothesis that all three methods are the same as measured by the accuracy can be rejected at a 5 % level of significance by the Friedman two-way analysis of variance by ranks [10]. Triplet and magnet losses are better than CPC at a 5 % level of significance by the post-hoc Nemenyi pairwise test [20]. However, the null hypothesis that magnet loss and triplet loss are the same cannot be rejected at the 5 % significance level by this test. This further supports the differing results between these two methods as mentioned in the previous subsection.

4.5 HTTP dataset

Table 3 compares the three methods on the proprietary dataset from Cisco Cognitive Intelligence. Here, magnet

[3] https://github.com/pevnak/Mill.jl

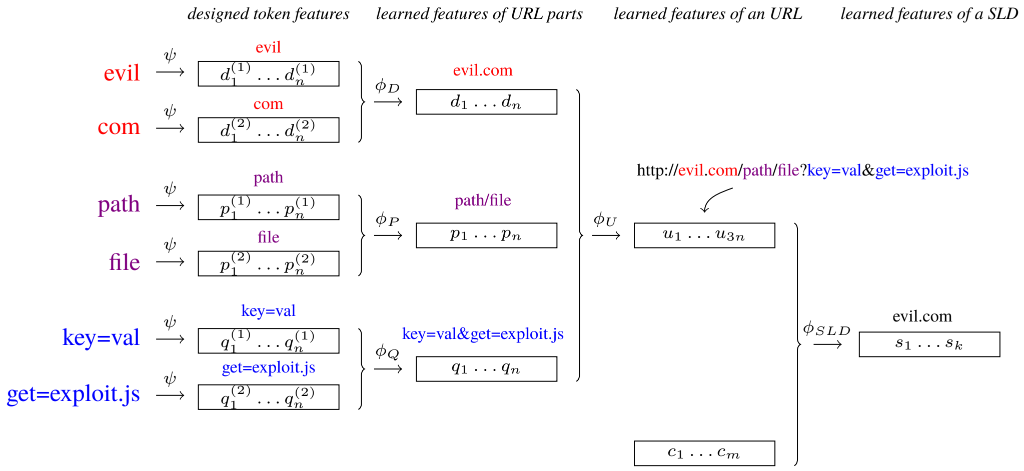

designed token features
learned features of URL parts
learned features of an URL
learned features of a SLD
evil
com
path
file
key=val
get=exploit.js
ψ
$d_1^{(1)} \dots d_n^{(1)}$
$d_1^{(2)} \dots d_n^{(2)}$
$p_1^{(1)} \dots p_n^{(1)}$
$p_1^{(2)} \dots p_n^{(2)}$
$q_1^{(1)} \dots q_n^{(1)}$
$q_1^{(2)} \dots q_n^{(2)}$
ϕ_D
ϕ_P
ϕ_Q
evil.com
$d_1 \dots d_n$
path/file
$p_1 \dots p_n$
key=val&get=exploit.js
$q_1 \dots q_n$
ϕ_U
http://evil.com/path/file?key=val&get=exploit.js
$u_1 \dots u_{3n}$
$c_1 \dots c_m$
ϕ_{SLD}
evil.com
$s_1 \dots s_k$

Figure 1: Hierarchical model of a URL. The vector $c_1, \dots, c_m$ represents the connection features.

Dataset	CPC	Triplet loss	Magnet loss
BrownCreeper	**0.900**	0.882	**0.900**
Elephant	0.575	**0.900**	0.825
Fox	0.400	0.675	**0.725**
Musk1	0.667	**0.889**	**0.889**
Musk2	0.750	**0.950**	**0.950**
Mutagenesis1	0.658	0.816	**0.912**
Mutagenesis2	0.708	**1.000**	**1.000**
Newsgroups1	0.533	0.950	**1.000**
Newsgroups2	0.600	0.900	**0.950**
Newsgroups3	0.683	0.700	**0.850**
Protein	0.820	**0.923**	0.897
Tiger	0.642	**0.825**	**0.825**
UCSBBreastCancer	0.333	0.667	**1.000**
Web1	0.667	0.733	**0.800**
Web2	0.689	**0.800**	0.733
Web3	0.733	0.800	**1.000**
Web4	0.622	0.800	**0.930**
WinterWren	0.936	**0.955**	0.936

Table 2: The accuracy of the final models for the publicly available datasets.

Variant	CPC	Triplet loss	Magnet loss
2 classes	0.920	0.910	**0.930**
20 classes	0.893	0.868	**0.904**

Table 3: The accuracy of the final models for the corporate datasets.

loss is the best of the three methods, however, only by a small margin over the other 2 methods. Since the datasets consisted of 90 % of negative samples however, none of the results is particularly remarkable and some further tuning would be needed if any of the methods were to be used in real environment.

5 Conclusion

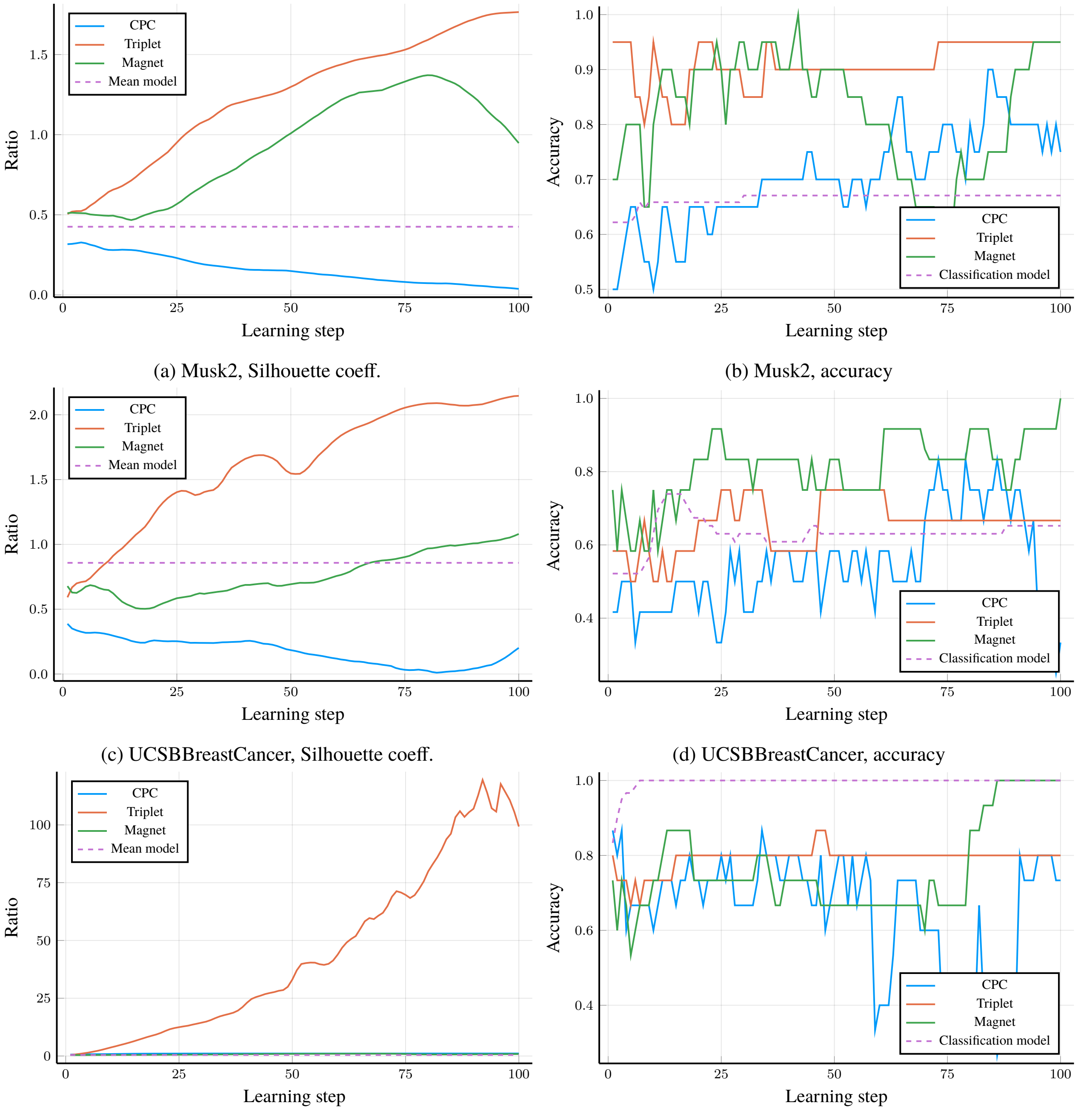
In our work, the underexplored research area of multi-instance clustering was investigated. Three methods for clustering of bags were introduced, of which one is unsupervised (CPC) and two are supervised (Triplet loss and Magnet loss). For each of the methods, the prior art it builds on was presented, along with its modification for the purposes of multi-instance clustering. All three methods were theoretically and experimentally evaluated and compared. The experiments were conducted first on publicly available datasets in a reproducible fashion. Following that, a corporate dataset of network security data was used as it is the intended application domain for this work.

Comparing the methods on the publicly available datasets shows the method based on contrastive predictive coding to perform the worst, with the other having no statistically significant difference between them. Similar results were also obtained on the corporate dataset of HTTP traffic, albeit none of the results were as good as anticipated. The method based on contrastive predictive coding performed poorly on both the publicly available datasets and the corporate one, however, the comparison might not be fair as the CPC method is unsupervised, whereas the other two can utilize labels on the training data, giving them a strong advantage.

The initial expectation of CPC being outperformed proved to be true. Even as such, the result is interesting and has a value of its own in providing a baseline and a comparison for these and future methods.

Acknowledgement

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 18-21409S and partially supported by the GAČR grant 18-18080S.



(a) Musk2, Silhouette coeff.

(b) Musk2, accuracy

(c) UCSBBreastCancer, Silhouette coeff.

(d) UCSBBreastCancer, accuracy

(e) Web3, Silhouette coeff.

(f) Web3, accuracy

Figure 2: Accuracy and the Silhouette coefficient on three selected datasets.

References

[1] Stuart Andrews, Ioannis Tsochantaridis, and Thomas Hofmann. “Support Vector Machines for Multiple-Instance Learning”. In: *Advances in Neural Information Processing Systems* 15 (Jan. 2002), pp. 561–568.

[2] David Arthur and Sergei Vassilvitskii. *k-means++: The Advantages of Careful Seeding*. Technical Report 2006-13. Stanford InfoLab, June 2006, pp. 1027–1035. URL: http://ilpubs.stanford.edu:8090/778/.

[3] J. Bezanson et al. “Julia: A Fresh Approach to Numerical Computing”. In: *SIAM Review* 59.1 (Jan. 2017), pp. 65–98. ISSN: 0036-1445. DOI: 10.1137/141000671. URL: https://epubs.siam.org/doi/10.1137/141000671 (visited on 09/01/2018).

[4] Forrest Briggs, Xiaoli Fern, and Raviv Raich. “Rank-loss support instance machines for MIML instance annotation”. In: *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Aug. 2012. DOI: 10.1145/2339530.2339616.

[5] Ying Chen and Ou Wu. “Contextual Hausdorff dissimilarity for multi-instance clustering”. In: *Fuzzy Systems and Knowledge Discovery (FSKD)*. Sichuan, China: IEEE Computer Society Press, May 2012, pp. 870–873. ISBN: 978-1-4673-0024-7. DOI: 10.1109/FSKD.2012.6233889. URL: https://ieeexplore.ieee.org/abstract/document/6233889/ (visited on 05/14/2018).

[6] Marek Dědič. “Optimalization of distances for multi-instance clustering”. MA thesis. Prague: Czech Technical University in Prague, Jan. 2020.

[7] Thomas G. Dietterich, Richard H. Lathrop, and Tomás Lozano-Pérez. “Solving the multiple instance problem with axis-parallel rectangles”. In: *Artificial Intelligence* 89.1 (Jan. 1997), pp. 31–71. ISSN: 0004-3702. DOI: 10.1016/S0004-3702(96)00034-3. (Visited on 05/31/2017).

[8] P. Elias. “Predictive coding–I”. In: *IRE Transactions on Information Theory* 1.1 (Mar. 1955), pp. 16–24. ISSN: 2168-2712. DOI: 10.1109/TIT.1955.1055126.

[9] Brian S. Everitt, Sabine Landau, and Morven Leese. *Cluster Analysis*. 4th ed. Taylor & Francis, 2001. ISBN: 978-0-340-76119-9.

[10] Milton Friedman. “The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance”. In: *Journal of the American Statistical Association* 32.200 (Dec. 1937). Publisher: Taylor & Francis, pp. 675–701. ISSN: 0162-1459. DOI: 10.1080/01621459.1937.10503522. URL: https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522.

[11] Xavier Glorot and Yoshua Bengio. “Understanding the difficulty of training deep feedforward neural networks”. In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. Mar. 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (visited on 09/01/2018).

[12] Richard H. R. Hahnloser et al. “Digital selection and analogue amplification coexist in a cortex-inspired silicon circuit”. In: *Nature* 405.6789 (June 2000), pp. 947–951. ISSN: 1476-4687. DOI: 10.1038/35016072. URL: https://www.nature.com/articles/35016072 (visited on 08/27/2018).

[13] Mike Innes. “Flux: Elegant machine learning with Julia”. In: *Journal of Open Source Software* (2018). DOI: 10.21105/joss.00602.

[14] Ján Jusko. “Graph-based Detection of Malicious Network Communities”. PhD thesis. Prague: Czech Technical University in Prague, 2017. URL: https://dspace.cvut.cz/handle/10467/73702 (visited on 04/02/2019).

[15] Melih Kandemir, Chong Zhang, and Fred A. Hamprecht. “Empowering Multiple Instance Histopathology Cancer Diagnosis by Cell Graphs”. In: *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*. Ed. by Polina Golland et al. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2014, pp. 228–235. ISBN: 978-3-319-10470-6. DOI: 10.1007/978-3-319-10470-6_29.

[16] Diederik P. Kingma and Jimmy Ba. “Adam: A Method for Stochastic Optimization”. In: *arXiv:1412.6980 [cs]* (Jan. 2017). arXiv: 1412.6980. URL: http://arxiv.org/abs/1412.6980 (visited on 06/25/2020).

[17] J. Kohout and T. Pevný. “Network Traffic Fingerprinting Based on Approximated Kernel Two-Sample Test”. In: *IEEE Transactions on Information Forensics and Security* 13.3 (Mar. 2018), pp. 788–801. ISSN: 1556-6013. DOI: 10.1109/TIFS.2017.2768018.

[18] J. MacQueen. “Some methods for classification and analysis of multivariate observations”. In: *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*. Vol. 1. Berkeley, California: University of California Press, 1967, pp. 281–297.

[19] Krikamol Muandet et al. “Learning from Distributions via Support Measure Machines”. In: *Advances in neural information processing systems*. 2012, pp. 10–18. URL: http://papers.nips.cc/paper/4825-learning-from-distributions-

via-support-measure-machines (visited on 06/29/2017).

[20] PB Nemenyi. “Distribution-free multiple comparisons (doctoral dissertation, princeton university, 1963)”. In: *Dissertation Abstracts International* 25.2 (1963), p. 1233.

[21] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. “Representation Learning with Contrastive Predictive Coding”. In: *arXiv:1807.03748 [cs, stat]* (Jan. 2019). arXiv: 1807.03748. URL: http://arxiv.org/abs/1807.03748 (visited on 12/29/2019).

[22] Tomas Pevny and Marek Dedic. “Nested Multiple Instance Learning in Modelling of HTTP network traffic”. In: *arXiv:2002.04059 [cs]* (Feb. 2020). arXiv: 2002.04059. URL: http://arxiv.org/abs/2002.04059 (visited on 06/25/2020).

[23] Tomáš Pevný and Petr Somol. “Using Neural Network Formalism to Solve Multiple-Instance Problems”. In: *Advances in Neural Networks - ISNN 2017*. Springer, Cham, June 2017, pp. 135–142. ISBN: 978-3-319-59072-1. DOI: 10.1007/978-3-319-59072-1_17. URL: https://link.springer.com/chapter/10.1007/978-3-319-59072-1_17 (visited on 05/23/2018).

[24] Soumya Ray and Mark Craven. “Learning Statistical Models for Annotating Proteins with Function Information using Biomedical Text”. In: *BMC Bioinformatics* 6.1 (May 2005), S18. ISSN: 1471-2105. DOI: 10.1186/1471-2105-6-S1-S18. URL: https://doi.org/10.1186/1471-2105-6-S1-S18 (visited on 01/01/2020).

[25] Soumya Ray and Mark Craven. “Supervised versus multiple instance learning: An empirical comparison”. In: *Proceedings of the 22nd International Conference on Machine Learning*. Jan. 2005, pp. 697–704. DOI: 10.1145/1102351.1102439.

[26] Oren Rippel et al. “Metric Learning with Adaptive Density Discrimination”. In: *arXiv:1511.05939 [cs, stat]* (Mar. 2016). arXiv: 1511.05939. URL: http://arxiv.org/abs/1511.05939 (visited on 06/05/2020).

[27] A. Srinivasan, Stephen Muggleton, and Robert King. “Comparing the use of background knowledge by inductive logic programming systems”. In: *Proceedings of the 5th International Workshop on Inductive Logic Programming*. 1995, pp. 199–230.

[28] Jun Wang and Jean-Daniel Zucker. “Solving Multiple-Instance Problem: A Lazy Learning Approach”. In: *Proceedings of the Seventeenth International Conference on Machine Learning*. Ed. by Pat Langley. Stanford University, Stanford, CA, USA: Morgan Kaufmann, 2000, pp. 1119–1125. URL: http://cogprints.org/2124/ (visited on 07/01/2017).

[29] Kilian Q Weinberger, John Blitzer, and Lawrence K. Saul. “Distance Metric Learning for Large Margin Nearest Neighbor Classification”. In: *Advances in Neural Information Processing Systems 18*. Ed. by Y. Weiss, B. Schölkopf, and J. C. Platt. MIT Press, 2006, pp. 1473–1480. URL: http://papers.nips.cc/paper/2795-distance-metric-learning-for-large-margin-nearest-neighbor-classification.pdf.

[30] Lihi Zelnik-manor and Pietro Perona. “Self-Tuning Spectral Clustering”. In: *Advances in Neural Information Processing Systems 17*. Ed. by L. K. Saul, Y. Weiss, and L. Bottou. MIT Press, 2005, pp. 1601–1608. URL: http://papers.nips.cc/paper/2619-self-tuning-spectral-clustering.pdf (visited on 01/06/2020).

[31] Min-Ling Zhang and Zhi-Hua Zhou. “Multi-instance clustering with applications to multi-instance prediction”. en. In: *Applied Intelligence* 31.1 (Aug. 2009), pp. 47–68. ISSN: 1573-7497. DOI: 10.1007/s10489-007-0111-x. URL: https://doi.org/10.1007/s10489-007-0111-x (visited on 07/30/2020).

[32] Zhi-Hua Zhou, Kai Jiang, and Ming Li. “Multi-Instance Learning Based Web Mining”. In: *Applied Intelligence* 22.2 (Mar. 2005), pp. 135–147. ISSN: 1573-7497. DOI: 10.1007/s10489-005-5602-z. URL: https://doi.org/10.1007/s10489-005-5602-z (visited on 01/01/2020).

[33] Zhi-Hua Zhou, Yu-Yin Sun, and Yu-Feng Li. “Multi-Instance Learning by Treating Instances As Non-I.I.D. Samples”. In: *Proceedings of the 26th International Conference On Machine Learning, ICML 2009* (July 2008). DOI: 10.1145/1553374.1553534.