Online semi-supervised learning for network intrusion applications: A meta-learning approach

Herna Viktor
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
hviktor@uottawa.ca

Yasaman Shahrasbi
School of Electrical Engineering and Computer Science
University of Ottawa
Ottawa, Canada
yasaman.shahrasbi@uottawa.ca

***Abstract*—As the number of data instances being generated continuously increases over time, and obtaining their labels is a time-consuming task, one of the main goals in Semi-supervised learning tasks is to find ways to efficiently find labels of the unlabeled instances. In this work, a meta-learning approach is introduced and compared against several self-training methods. We find that this approach cannot efficiently find labels of the unlabeled data within the scenario we defined and experimented with.**

***Index Terms*—Semi-supervised Learning, Online Learning, Machine Learning**

I. Introduction

Data streams are sequences of data instances arriving one at a time. As opposed to the traditional machine learning setting, in the streaming model, all the training data is not available at the time of model training. Algorithms employed for data streams should be able to process each data instance in a short amount of time without storing them in the memory for long. Moreover, these algorithms should be able to provide answers, either for classification or regression problems, at any time when a query is made. [**1**]

Data streams flow plentifully and at high speeds. Such faced-paced incoming data streams, as well as the nature of some problems, make it impractical to obtain labels right after observing the data instances. Semi-supervised learning (SSL) methods are approaches that try to employ unlabelled instances as well as labeled instances to build models that are able to predict the labels of new, unseen data. Also, semi-supervised learning approaches are used in problems where it is assumed that the number of unlabelled instances is far more than the labeled ones.

Essentially, the goal of semi-supervised learning approaches is to make the most use of the unlabelled data to build classifiers that perform better than the classifiers built only on the labeled data. Self-training approaches are a branch of SSL methods. In such methods, in each iteration of the algorithm, one or more base models are trained on the labeled set of data. Then the labels of unlabelled instances are predicted using the trained models, and the instances with the highest confidence are added to the labeled set. This continues until all the instances in the data set are labeled. In this work, KNN is used as the base learner of the self-training method. In a parallel manner, several self-training approaches are run. In each of which, a KNN model with a different parameter (indicating the number of neighbors) is used. Furthermore, to assess the performance (accuracy) of each KNN model as a self-training model, a different Hoeffding tree model is employed. The main objective is to see what parameter as the number of nearest neighbors has the highest performance in assigning labels to unlabelled instances within a cybersecurity context using the NSL-KDD dataset. Lastly, after finding the performance of each KNN model using the corresponding Hoeffding tree model, two questions arise

– Is there a relation between the performance of the Hoeffding tree models and the number-of-neighbors parameter of the KNN models?
– Does only one KNN model perform the best on the whole dataset, or is it possible that different KNN models would work better on different subsets of the data set?

To answer these two questions, the performance of Hoeffding tree models is compared for each window of data instance. More details regarding these questions are provided in the Framework and Experimental Evaluation Sections.

II. Background and related work

A. *Hoeffding Tree Algorithm*

The Hoeffding tree algorithm is a very fast decision tree used for streaming data. It has been proved that the Hoeffding tree algorithm built through time in a streaming setting converges to a decision tree algorithm that is built by the whole dataset. Hoeffding tree algorithm is based on Hoeffding's bound; in essence, in each iteration, for splitting the data, the split gain, i.e., the impurity of the split, is calculated, and if the difference between the best attribute and the second-best attribute is greater than the Hoeffding bound, the tree will be split on the best attribute. [**2**]

B. *k-nearest Neighbours Algorithm*

K-nearest neighbor (kNN) is one of the simplest classification methods in which no information regarding the distribution of the instances is available. In kNN, after training the model with the training instances, given a new instance and the number of neighbors (k) that it should calculate the distance of the new instance to, kNN will output the majority vote of the labels of the k nearest neighbors as the label of the newly seen instance.

C. Meta-learning

According to the no free lunch theorem [**3**], there is no single machine learning algorithm that outperforms other algorithms for all the existing problems. One of the automated algorithm selection methods is meta-learning-based algorithms. [**4**] Meta-learning refers to the learning process of learning, where several machine learning methods are employed on different sets of problems through which the meta-learning gains experience and would be able to recommend suitable algorithms for new tasks in the future. [**4**, **5**]

D. Semi-supervised Learning

In supervised learning tasks, the input is a set of data points (training data) along with their labels, real values for regression, and categorical for classification tasks. In these problems, the goal is to infer a function (a learner) from the training data so that given a new instance, the classifier/regressor is able to find the label. On the other hand, in unsupervised learning tasks, a set of data points is given as the training data; however, the labels are not known. In such problems, the goal is to find patterns and model the underlying structure of data. Further, given unknown instances, the model is able to find its labels. Moreover, there is a third group of tasks where a small number of labeled instances and a large number of unlabeled instances are available. Semi-supervised learning is a type of machine learning approach that targets this group of tasks. In such problems, acquiring the labels of unlabeled instances is expensive or hard; therefore, the main idea in semi-supervised learning approaches is to make use of the unlabeled data along with the labeled data, if there are enough unlabeled instances available and semi-supervised learning assumptions are satisfied. The goal of using unlabeled data along with labeled data in semi-supervised learning methods is to build a learner that outperforms a learner built only with labeled instances. The SSL assumptions are smoothness, low density, and manifold. In the smoothness assumption, the idea is that if two data instances are close enough in the input space, their class labels are close, too. Transitive relation also exists for the unlabeled data, meaning that if instance a is labeled and close to b and instance c is close to b and not a, then b and c also have the same label as instance a. According to the low-density assumption, the decision boundary lies in a region with the fewest number of data instances. In the third assumption, the manifold assumption, it is assumed that the data instances are composed of several distributions each of them lying on a lower-dimensional manifold. Also, the points on the same manifold have the same label. In SSL, the unlabeled data instances are useful when they contain information more than ones already existed in the labeled data. Furthermore, the SSL methods should be able to bring out this extra information; however, there are no strategies to find which SSL algorithm fits which problem except for reasoning. For example, as graph-based SSL models are based on local similarities, they would be good options for problems in which local similarities can be logically defined. Also, SSL models that are variants of supervised learning models are based on the same assumptions, so if supervised learning models work well for a problem, the SSL variant would also perform well in the same problem. For example, SSL and supervised support vector machines rely on the low-density assumption. It is worth noting that using unlabeled data in SSL does not always certainly outperform the learner trained on only the labeled data; the SSL methods are only another set of techniques that are used in hope of coming up with learners that improves the performance of previously used models. Semi-supervised learning methods differ based on how they use unlabelled data, the SSL assumptions, and their relationships to supervised learning methods. There are two main groups of SSL methods: Inductive, in which the goal is to find a model, and transductive, where no learners are built, and these methods are able to predict the labels only based on the training data it has seen during training time. Wrapper methods are among inductive methods. In wrapper methods, one or more supervised learning algorithms are used to train a learner and use the learner(s) to give labels to the unlabeled data. Depending on the number of classifiers used in the wrapper methods, there are three groups of approaches: self-training, co-training, and pseudo-labeled boosing methods. In the self-training approach, one supervised learning model is used as the base learner, where first the learner is trained on the labeled data, then the learner is used to predict the labels of the unlabelled data. The most confident pseudo-labeled instances are added to the labeled instances, and this process continues until all the unlabelled instances are labeled. [**2**]

E. Related Works

The authors in [**8**] have introduced a self-training windowing ensemble model that using the output weight of the ensemble, predicts the labels for the unlabeled instances. The drawback of this approach is that it can reinforce errors.

In [**9**], a 1NN semi-supervised learning approach is provided that in order to assign a label to an unlabeled instance, in each iteration, a 1NN is trained on the currently labeled instances, and if the distance between the first nearest neighbor to the current unlabeled instance is less than a specific threshold epsilon, the same label as the nearest neighbor, will be assigned to the unlabeled instance. This approach is dependent on the value of parameter epsilon, where the label differs using different thresholds and might lead to wrong results.

The authors in [**10**] have presented an approach that combines a self-training semi-supervised approach and an active learning method to predict the labels for the unlabeled instances. In even iterations, a classifier calculates the prediction probabilities of the unlabeled data, and the most confident ones will be added to the set of labeled instances. On the other hand, in the odd iterations, the entropy of the unlabeled instances is calculated. Then they are sorted from the most confusing to the least confusing, and using a human oracle, the labels for the top instances are obtained. The problem with this method is that in both approaches, SSL and AL, the number of "top" instances is not defined, and using different numbers affects the result. Also, this method is highly reliable on the AL part, using the AL approach for 50% of the unlabeled instances; However, the human oracle is not always available, and it is not easy to get the real labels.

F. Cybersecurity

Intrusion Detection: Intrusion is any unauthorized activity from unknown hackers from outside of a system or insiders who try to steal important and valuable data from computer systems. As the internet is becoming an inseparable part of people's lives, the attackers seek new ways to exploit the internet users' sensitive information and put the confidentiality, integrity, and availability of data and networks at risk [**6**]. An intrusion detection system (IDS) is software that scans the network to detect intrusions [**7**]. The two major intrusion detection systems that have been proposed are anomaly-based intrusion detection systems (AIDS) and signature-based intrusion detection systems (SIDS).

Signature-based IDS (SIDS): In Signature-based IDS, once a threat is revealed, it is added to the database of the threats. Incoming data are compared to the detected threats' database to detect whether it is malicious or not. SIDS can achieve high performance with almost no false positives; the reason is that the predictions are based on known malicious data. However, the drawback of this method is the weakness in detecting zero-day vulnerabilities. From the SIDS point of view, network intrusion detection problem is formulated as follows: considering the stream of the data flowing in, we need to assign each data packet to one of the two classes representing normal traffic or intrusions.

Anomaly-based IDS (AIDS): As another method of intrusion detection, Anomaly-based IDS makes use of machine learning or statistical-based methods to detect malicious data. As a result, it becomes more helpful to detect unseen data like zero-day attacks as opposed to the SIDS methods [**6**]; however, we may have more false-positive predictions compared to SIDS.

III. Framework

The goal is to compare several KNN algorithms as self-training algorithms and a meta-learning algorithm as a labeling scheme in semi-supervised learning problems. The strategy is first to observe instances from a stream, window by window, with pre-defined window size. In a semi-supervised scenario like this work, each window might contain several labeled and unlabeled instances. The second step is to separate labeled and unlabeled instances and keep the labeled instances, their labels, and unlabeled instances in three distinct lists. In the next step, which is the core part of this work, several k-nearest neighbor algorithms as self-training algorithms are used first to be trained on the labeled instances and then assign labels to unlabeled instances by predicting their labels. In the next step, after assigning labels to the unlabeled instances using KNN algorithms, several Hoeffding tree algorithms are used to calculate the performance (accuracy) of each KNN algorithm. Lastly, the meta-learning framework works as follows: for each incoming window, comparing the accuracy performances of Hoeffding trees show which KNN model is the most promising one; for instance, there might be a case in which 1nn works better in labeling the unlabeled data in the *i*th window, but 3nn work better for the *(i+k)*th window. [1]

Algorithm 1 Main Algorithm

```
1:  stream: A stream that includes both labeled and unlabeled instances
2:  y: Real labels of instances
3:  n: set of numbers used as nearest neighbors
4:  assigned_labels: A dictionary to keep the labels that are assigned to the unlabeled data instances
5:  for The stream of data is coming do
6:      for $i < window\_size$ do
7:          add an element from the stream to the window
8:      end for
9:      separate window into labeled_instances_X, labeled_instances_y, and unlabeled_instances lists
10:     for each nn in range(0, n, 2) do
11:         train knn on pairs of (labeled_instances_X, labeled_instances_y)
12:         assigned_labels ← predict the labels of unlabeled_instances list using the knn model.
13:     end for
14:     for each nn in range(0, n, 2) do
15:         for instance in labeled_instances do
16:             prediction.append(Ht_nn.predict(instance))
17:             Ht_nn.partial_fit(instance, y ∈ labeled_instances_y)
18:         end for
19:         for instance in unlabeled_instances do
20:             prediction.append(Ht_[nn].predict(instance))
21:             Ht_nn.partial_fit(instance, y ∈ assigned_labels
22:         end for
23:         Calculate accuracy(prediction, true_labels)
24:     end for
25: end for
```

[1]https://github.com/Yacmean/meta-learning.git

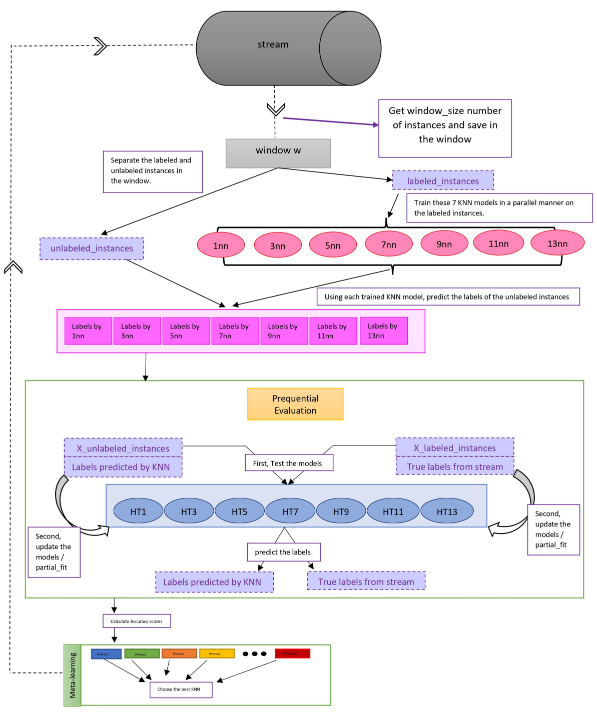

Fig. 1. Framework of the proposed method

IV. Experimental Evaluation

All the experiments are written in Python, and Scikit-learn and Scikit-multiflow are employed to write this code. Default parameters were used for the Hoeffding tree and the KNN algorithms; also, as the number of neighbors parameter, 1, 3, 5, 7, 9, 11, and 13 are chosen. Furthermore, the window size is 300, and the first 20% of the labeled instances from the stream are used to pre-train the Hoeffding tree algorithm. Overall there are 10000 data instances in this stream. Also, in this experiment, different levels of unlabeled ratio are compared: 0%, 10%, 25%, 50%, 75%, and 90%, and different numbers of neighbors.

For the dataset, KDDTrain+.TXT and KDDTest+.TXT files from the Canadian Institute for Cybersecurity website are used. First, these two datasets are merged and shuffled. This dataset has 43 features, where the last one corresponds to the intensity level of the traffic, so this feature is removed as this project is intended to tackle a binary classification scenario. Moreover, to handle the three categorical features -protocol_type, service, and flag-, label encoder is fit on the whole dataset to convert them to numerical data. Next, the min-max scaler is fit on the dataset to normalize the data.

In table I, the comparison between the mean accuracy score of the seven different Hoeffding tree models for different levels of unlabeled ratios along 300 windows are shown. According to table I, this labeling strategy works the best for 10% of unlabeled data and gets worse as the ratio of unlabeled data increases.

Moreover, figures Fig. 2 through 6 show the accuracy comparison of each Hoeffding tree when using different KNN models for labeling the unlabeled instances. Also, according to figure. 3, all the KNN models almost work the same, except for the 13NN, where there is a sudden decrease in the performance of the Hoeffding tree model around the 10th window forward. Figure Fig. 7 compares all the different levels of unlabeled ratio as well as the supervised learning method. To compare the meta-learning approach, each plot shows accuracy scores when using the best performing KNN per each window. As it is shown, comparing to the supervised learning baseline, the method proposed in this work works the best when there is only 10% of unlabeled data. For other levels of unlabeled ratio, the results looks somewhat the same, and this approach is not much promising.

HT	0	10	25	50	75	90
1nn	88.68	84.50	77.18	58.63	55.70	53.43
3nn	88.68	84.48	77.65	58.77	55.93	52.79
5nn	88.68	84.43	77.09	58.59	53.51	52.85
7nn	88.68	84.41	77.95	58.68	53.08	53.13
9nn	88.68	84.27	77.92	58.64	53.41	52.82
11nn	88.68	84.49	77.61	58.67	53.23	52.87
13nn	88.68	84.59	77.88	54.94	53.05	52.82

TABLE I
Performance Accuracy of Hoeffding tree models when each KNN model is used for labeling the unlabeled instances.

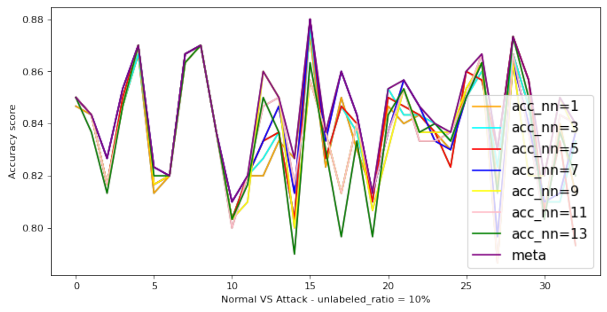

Fig. 2. Comparing the accuracy performance of Hoeffding tree for each KNN model along all the windows, when having 10% unlabeled data.

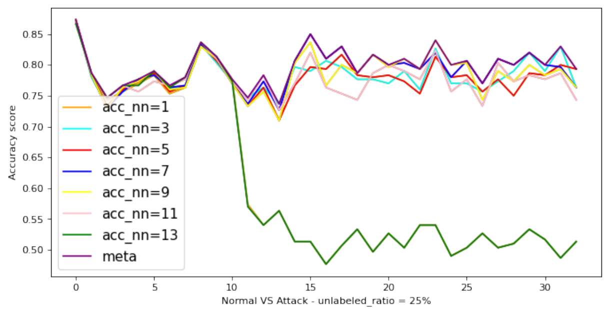

Fig. 3. Comparing the accuracy performance of Hoeffding tree for each KNN model along all the windows, when having 25% unlabeled data.

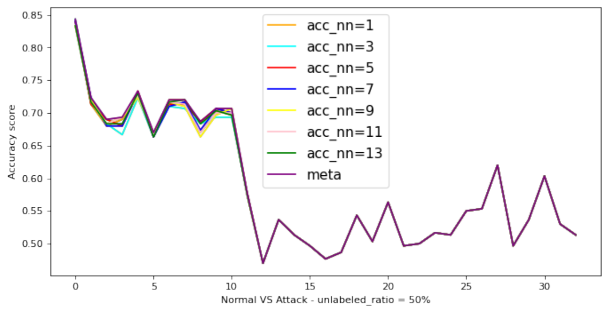

Fig. 4. Comparing the accuracy performance of Hoeffding tree for each KNN model along all the windows, when having 50% unlabeled data.

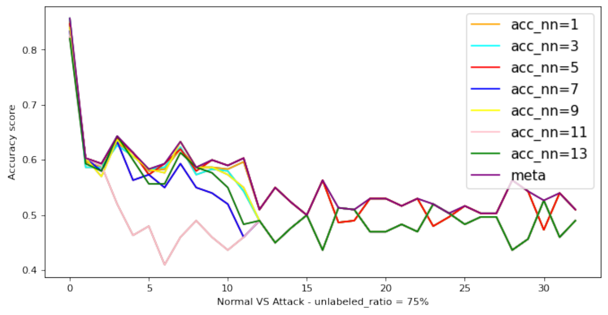

Fig. 5. Comparing the accuracy performance of Hoeffding tree for each KNN model along all the windows, when having 75% unlabeled data.

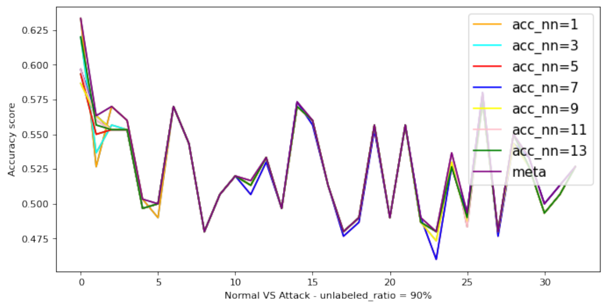

Fig. 6. Comparing the accuracy performance of Hoeffding tree for each KNN model along all the windows, when having 90% unlabeled data.

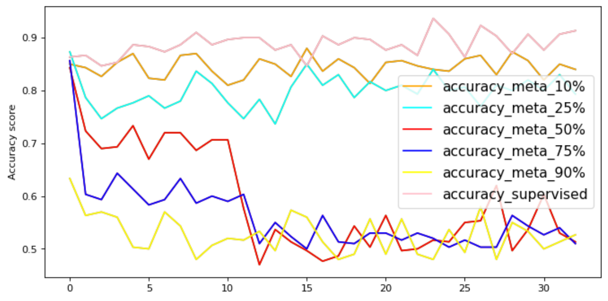

Fig. 7. Comparing the proposed method for different unlabeled ratios in one sight.

V. Conclusion

In this specific scenario, using this dataset, and parameters for the models, as shown in the Experimental Evaluation section, when the number of nearest neighbors of KNN model increases, the Hoeffding trees' performance decreases.

Also, when using the meta-learning approach, in which the best performing KNN is used in each window for labeling the unlabeled data, increasing the unlabeled ratio, decreases the performance of the model.

As future work, other semi-supervised algorithms such as co-training, boosting, or graph-based algorithms for handling the unlabeled data, and different datasets, imbalanced or ones with concept drift can be used.