Anomaly detection algorithms using active learning

Sofia Vergara (230259, sofia.vergarapuccini@tu-dortmund.de)

Abstract

Active learning for anomaly detection improves the accuracy of a model by providing iteratively a budget of data points to a human analyst for obtaining their true label. Usually anomaly detection models are based on profiling normal instances for determining what an anomaly is. An alternative to this, consists on isolating anomaly instances given a set of attributes. On this paper the BAL algorithm [4] using isolation Trees and the WisCon algorithm [1] for contextual anomaly detection are discussed. Furthermore, the research is extended to streaming data since a change in the distribution of the data might impose a challenge to the learning algorithm. In active learning, informative examples have to be provided to the analyst so these are useful to the algorithm to learn. This paper covers two query techniques which take into account how diverse are the anomalies and how relevant for the model are the subspaces where these are found.

1 Introduction

Anomaly detection consists on unveiling data points that present an abnormal behaviour compared to regular instances (nominals). Such data points are rare among the complete set and in unsupervised environments are hard to classify, since no hard boundary exists that is able to identify them. Anomaly detection has different fields of application ranging from fraud detection to the discovery of a new star in astronomy. Thus, it is of importance that the model is very precise with the correct classification of true anomalies (true positives). Considering active learning within the anomaly detection framework can enhance the performance of the detection algorithm. Active learning consists on repeatedly querying an oracle, e.g human analyst for labelling a budget of instances. The true labels are then provided to the algorithm so this can correctly learn the decision boundary. This approach is of great relevance when there is abundant unlabelled data and the process of labelling is expensive [8]. Some of the approaches considering active learning are combined with ensemble models given that these produce more robust results than a single detector.

On this paper, isolation Forest trees (IFOR) [7] using different weighting schemes are presented for anomaly detection. Furthermore, it is assume that the analyst is in the capacity to label a low number of instances compared to the size of the data set. Therefore, informative examples must be provided. Some of the strategies for providing such examples consist on taking the instances with the highest anomaly scores or choosing samples close to the uncertainty region. However, new strategies have been developed considering the overlapping attributes of the instances and the relevance of contextual features. These strategies are discussed on this paper. Anomaly detection using active learning, can be as well extended to streaming data. One of the challenges of streaming information is that incoming data might have a change in its distribution, affecting the performance of the model to unseen data. Hence, the distribution of the data and weighting of new instances have to be taken into account on the learning algorithms. One such model in this paper is the streaming active learning algorithm (SAL), in which the KL-Divergence is used to replace old data by new one when there is great divergence in the distribution of the set. This paper is organized as follows: Section 2 presents anomaly detection algorithms using active learning for static and streaming data proposed in [4]. Section 3 describes some limitations of the algorithms presented on section 2, and how other techniques overcome such limitations. Furthermore, section 4 shows some query strategies for obtaining informative instances and finally section 5 concludes this paper.

2 Active learning for anomaly detection

Active learning consists on the interaction between an analyst and a learning algorithm, in which the feedback of the analyst on true labels of the data is provided to the algorithm in order to adjust its parameters and improve the prediction accuracy.

In Active Anomaly detection, few but informative samples of the data (nominal or anomaly) are labeled by the user and with these, the algorithm solves an optimization problem in order to obtain new hyperparameters for the base detector(s), refining the performance of the algorithm. The goal in active anomaly detection is to maximize the amount of true positives (real anomalies) presented to the user from a limited amount of instances available. Unlike in classification problems, in anomaly detection there is no hard decision boundary separating anomalies from nominal data making the performance of a single detector highly susceptible to the data, type of anomalies, problem application, etc. Therefore, ensemble methods, i.e combining predictions from multiple detectors, achieve more reliable results and have fewer false positives. The next subsection discusses a framework for anomaly detection using active learning and isolation forest trees to target these instances.

2.1 Tree based anomaly detection ensembles algorithms

[4] proposes an active learning framework (Figure 1) consisting on multiple iterations and in each of these, the following steps are performed:

- Select one or more data points from the input data set based on a query strategy using the results of the initialized model, e.g highest scored instances, uncertainty sampling, among others.
- Provide the chosen data points to the analyst with a set of explanations or interpretative rules to carry on the labelling process.
- Based on the feedback, update the weights of the scoring functions so that the algorithm is consistent with the analyst's feedback.

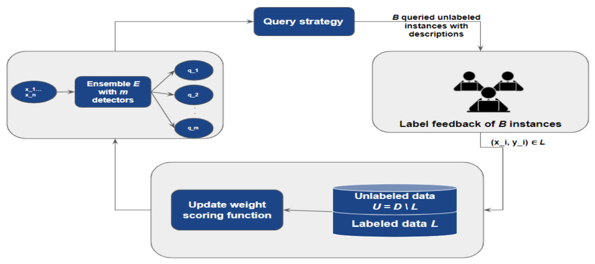

Query strategy
B queried unlabeled instances with descriptions
x_1... x_n
Ensemble E with m detectors
q_1
q_2
q_m
Label feedback of B instances
(x_i, y_i) ∈ L
Update weight scoring function
Unlabeled data U = D \ L
Labeled data L

Figure 1. Framework for Anomaly Detection with Active Learning

Let ε be the ensemble composed by m detectors, which assign scores $\mathbf{z}$, $\mathbf{z} = \{z_1, ..., z_m\}$, to each instance $\mathbf{x}_i$, $\mathbf{x}_i \in R^d$, and it has associated a hidden label $y_i = \{-1, +1\}$, where +1 corresponds to an anomaly. The scores $\mathbf{z}$ are assumed to be normalized, ranging from [-1,1] or [0,1] where higher values represent a higher degree of anomaly. For a tree ensemble the scoring function is a linear combination of the scores given by $S(\mathbf{x}) = \mathbf{w} * \mathbf{z}$, where $\mathbf{w}$ are the weights assigned to each detector, $\mathbf{w} \in R^m$, and represent the parameters Θ to tune using the analyst's feedback. The initialization values of these weights are important for improving the label-efficiency of active learning. Usually the detectors get the same weights as a starting point. The ideal set of weights, allows the scoring function to push anomalies in a extreme positive region of the scoring space, while nominal instances have lower scores. However, this ideal set is not often reached in practice, i.e the obtained parameters and ideal weights are misaligned, which contribute to the occurrence of false positives. Hence, providing the user good examples to label helps the algorithm on learning the optimal weights efficiently.

By designing a hyperplane that passes through the uncertainty region and choosing data points located in this, the model shapes the decision boundary reliably, even if the selected instance is a nominal (false positive).

One of the tree based ensembles used for anomaly detection, is Isolation forest (IFOR) [7]. This method consists on isolating anomalies instead of profiling how a normal instance should be. Since in a data set anomalies are few and some of their attributes are very different from nominal instances, these can be isolated by means of a binary tree structure, in which its partitions are generated by selecting randomly an attribute of the feature space and a random split point between the minimum and maximum of the selected attribute. This process is done recursively until the instance reaches a leaf node.

The number of partitions required to isolate a point is equivalent to the path length l from the root node to a leaf node. Since anomalies have distinguishable attributes from nominal values, these are susceptible to being isolated faster, i.e at earlier partitioning, which produces shorter path lengths from the root to the leaf (Figure 2). It is worth noting, that isolation forests are only used in ensembles, so the average path length of an anomaly is shorter. On the other hand, nominal values belong to denser regions, so more partitions are needed to reach a leaf node, reason why their path length is deeper.

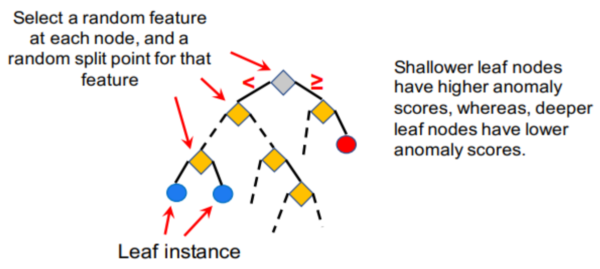

Select a random feature at each node, and a random split point for that feature
<
≥
Shallower leaf nodes have higher anomaly scores, whereas, deeper leaf nodes have lower anomaly scores.
Leaf instance

Figure 2. Isolation forest methodology [7]

After training a model with T trees, the leaf nodes of each tree represent the m ensembles. The path length l, from the root to the leaf, multiplied by -1 represents the score $\mathbf{score}_i = -l$ of each instance. If the instance does not belong to this partition (leaf), its score is equal to 0. Moreover, instances which are isolated faster have a less negative scores due to its shorter path length (anomalies) compared to nominal points. Afterwards, the scores are normalized, since these might vary in scale for each detector. Given that a data point belongs to only one leaf node per tree, and the model is trained with T trees, each instance belongs to a few leaf nodes (equal to T) so the score vector is sparse.

Among some of the advantages of IFOR, it is found that since this method relies on partitioning recursively the feature space, this does not utilize distance or density measures which require major computational costs as the dimensionality increases. Furthermore, IFOR has linear time complexity and a low memory requirement, given the sparse scoring vectors and it has the capacity to scale up to handle large and high dimensional data.

Other tree based ensemble models are HST and RSF [5]. These two techniques have a fixed depth, so the anomaly scores are computed using sample counts and densities at the leaf nodes.

2.2 Obtaining Compact subspaces

For subsection 2.2, 2.3 and 2.4 the general terms explaining each part of the algorithm have the following notation:

- ε is the ensemble composed by m detectors
- $Z = \{z_1, ..., z_p\}$ represent the instances to be described, e.g true anomalies discovered by the analyst and $|Z| = p$
- $S = s_1 \cup ... \cup s_p$ are the most relevant subspaces containing z_i and $|S| = k$
- $\mathbf{v}$ are the volumes of the subspaces in S, $\mathbf{v} \in R^k$
- $\mathbf{x}$ is a binary vector where 1 indicates that the subspace s_j is included in the covering set, $\mathbf{x} \in \{0, 1\}^k$, and 0 otherwise
- $\mathbf{u}_j$ is a vector for each instance z_i containing 1 if the instance belongs to subspace s_j and 0 otherwise, $\mathbf{u}_j \in \{0, 1\}^k$
- $U = [\mathbf{u}_1^T, ..., \mathbf{u}_p^T]^T$ is the matrix composed by the $\mathbf{u}_j$ vectors
- B is the budget of instances the analyst can label
- $\mathbf{w}$ are the weights of the m ensembles
- $\mathbf{H}$ is the score matrix of unlabeled instances obtained from the ensemble
- $\mathbf{H}_+$ is the score matrix for instances with a label +1 (received from analyst), i.e anomaly
- $\mathbf{H}_+$ is the score matrix for instances with a label -1, i.e nominal
- τ is the hyperparameter indicating the expected fraction of instances that are anomalous

It is of interest to provide the analyst descriptions of the instances to label because these are useful for understanding the predictions made by the model in a concise way. To achieve this, it is necessary to select a subset of the most relevant and less voluminous, i.e compact, subspaces which contain all instances to be described. This problem is treated as part of the set cover problem [6], since it is desired to get the smallest collection of subspaces including all data points to describe.

A compact set of subspaces containing all the candidate instances is obtained as:

$$S^* = \operatorname{argmin}(\mathbf{x} * \mathbf{v}), \qquad (1)$$

s.t $\mathbf{U} * \mathbf{x} \geq \mathbf{1}$, where $\mathbf{1}$ is a column vector of p 1's.

By obtaining a compact set of subspaces it is easier to quickly query on the next iterations different classes of anomalies belonging to different subspaces. Moreover, with this subset it is possible to get additional information such as the amount of instances per relevant subspace helping to prioritize the work of the analyst.

2.3 Obtaining precise subspaces

Although the previous approach provides the minimum amount of subspaces containing all instances to be described. It does not consider the precision of the subspace, i.e if many nominals are included in this, nor how easy the predicate rule defining the subspace is. An example of a predicate rule is the following: If credit score = 'Low' or (employed = False and savings < 100), then approve loan = False. In this rule, three features (credit-score, employed, savings) are taken into account for defining a description. As the length of the predicate rule increases, the definition of the subspace becomes harder to understand. To solve this, the Equation (1) is modified as:

$$S^* = \operatorname{argmin}(\mathbf{x} * (\mathbf{v} \circ (\mathbf{1}_k + \eta) + h)), \qquad (2)$$

s.t $\mathbf{U} * \mathbf{x} \geq \mathbf{1}_p$, where $\mathbf{1}_k$ is a column vector of k 1's, $\eta = \{\eta_1, ..., \eta_k\}$ are the amount of nominal data points present in the subspaces in S and $h = \{h_1, ..., h_k\}$ are the rules' complexities of the subspaces in S given by $h_j = 2^{\text{rule\_length}(s_j)-1}$, the rule_length is defined by the number of feature-range predicates required to define a description. This equation penalizes how imprecise and complex the subspace s_j is.

The procedure for finding the optimal set of subspaces starts by taking all candidate subspaces (leaf nodes) obtained by the ensemble and the set of instances to describe Z, which contain true positives and in addtion, unlabeled instances sampled as nominals. Using the feedback of the analyst, it is possible to identify the most relevant candidate subspaces for the selection process. The algorithm starts by calculating the volume, amount of nominals η and complexity of the subspaces . Afterwards, Equation 2 is applied, and the subset of subspaces S* is obtained. Furthermore, the precision of the subspaces is computed based on η, and only those having a precision greater than a threshold t are retained, i.e subspaces having only few false positives. This process uses stepwise selection and filtering for obtaining the optimal subset of subspaces providing simple descriptions for the analyst.

2.4 Update of ensemble weights

[4] proposed the model Batch Active learning (BAL) using Isolation Forest and active learning for finding anomalies. A key part of the model is the parameter tuning process. BAL updates the weights of the scoring function aiming to keep the hyperplane in the region of uncertainty through the budget B. This algorithm depends only on the hyperparameter τ, being this the fraction of instances that are anomalous. Hence, if the anomalies are rare, τ should be set to a small value.

The algorithm iterates through the instances of the budget B and calculates the anomaly scores as $\mathbf{a} = \mathbf{H} * \mathbf{w}$. In each iteration t, the highest scored instance is given to the analyst to label and if the label assigned is +1, the score of the data point is added to the matrix $\mathbf{H}_+$, otherwise to $\mathbf{H}_-$. Furthermore, the score of the instance is removed from the matrix $\mathbf{H}$ so that is not taken into account in the next iteration. By this point, the weights are recalculated by minimizing a loss function composed by the hinge loss, in which if the anomaly scores of the true positives in $\mathbf{H}_+$ computed with the current set of weights $\mathbf{w}^{(t)}$, are lower than in the previous iteration, the model is penalized, encouraging that the scores in $\mathbf{H}_+$ are higher for the current iteration than for the previous one and for $\mathbf{H}_-$ lower. The loss function as well takes into account the influence of the prior weights, and as more instances are labeled, the importance of the priors reduces. This optimization problem is solved using gradient descent and the weights of the base detectors take as initialization values $\mathbf{w}_{unif} = [\frac{1}{\sqrt{m}}, ..., \frac{1}{\sqrt{m}}]^T$. The algorithm finishes when the number of iterations is the same as the number of instances in B, and this returns the normalized vector of final weights and matrices of scores $\mathbf{H}$, $\mathbf{H}_+$, $\mathbf{H}_-$.

2.5 Ensemble algorithms for anomaly detection in streaming data

Tree ensembles can as well be extended to streaming data that comes in windows of size K and might be unlimited. The streaming active learning algorithm (SAL) handles the phenomenon of concept drift [9], which is characterized by the change of distribution of the data during the course of time, often producing that a model is built on obsolete data, forcing it to have a regular update to be consistent with the new behaviour. SAL starts by training all members of the ensemble with the first window. If the model is an IFOR and a new window arrives, only a subset of the trees of the current window is replaced with the information of the new window if there is a large change in the distribution of the data. This is evaluated using the KL divergence [2], which is an asymmetric measure on how one probability distribution P is different from a second one Q; in information theory it can as well be seen as the amount of information lost when Q is used to approximate to P. Thus, if there is a significant number of trees that are divergent in the current model, then all of the corresponding nodes and learned weights of these are discarded. On the other hand, the m' leaf nodes and weights of the new trees are added to the current model. The weights of the new trees are initialized in $v = \frac{1}{\sqrt{m'}}$.

The updated model is employed to determine which unlabeled instances to keep. This one retains only the most anomalous instances among the ones in memory and the new window, discarding the rest. Afterwards the weights are tuned with the analyst feedback using a process similar to BAL. The algorithm continues until no more windows are obtained or the budget B is exhausted. In general it is not easy to know how large is the change in distribution of the data from one window to another, therefore it is reasonable to change 20% of the older ensemble members for new ones.

3 Related Work

3.1 Ensemble algorithms for contextual anomaly detection

The BAL algorithm uses the full dimensional space to detect anomalies by means of tree ensembles. However, real world systems often produce anomalies that are catalogued as such depending on the situation. For example in cold regions, high energy consumption is normal during winter but the same behaviour might be abnormal in summer, i.e in this situation the environmental factor contextualizes what an anomaly is. Hence, considering only a global perspective can be misleading given that it can produce or hide abnormal instances. To leverage this, a technique named "Wisdom of contexts" (WisCon) [1] uses contextual attributes e.g "ambient temperature" for defining contexts and behavioural attributes e.g "energy consumption" to determine whether an instance deviates significantly from the other data points or not. A context C is defined as a subset of attributes from a set D, and the remaining represent the behavioural features $B = D \setminus C$. In a large data set, a context can be decided in many ways, therefore one can take initially all possible combinations of contextual attributes from the feature set.

The framework of WisCon is similar in structure to the one presented by [4]. However, WisCon calculates on the third step the importance of the contexts and prune those which are not relevant based on the analyst's feedback. Afterwards it combines all anomaly scores from the individual base detectors with their weights to generate the final anomaly scores. A brief description of the steps is as follows:

- Create multiple contexts from the feature space and generate a base detector for each of these, producing a set of anomaly scores.
- Provide a small budget of anomalies to the analyst, to catalogue them as anomalies or nominal values. Different query strategies can be employed at this point.
- Estimate the importance of each context based on the analyst feedback using a strategy similar to AdaBoost. Prune uninformative contexts and calculate a weighted aggregation of the remaining.

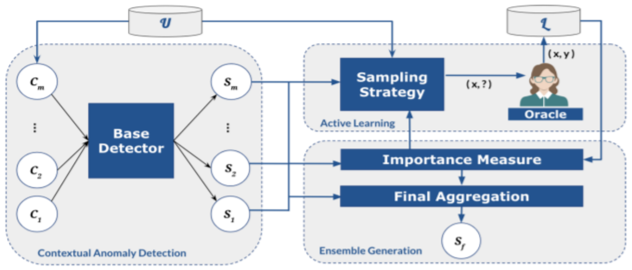


Figure 3. WisCon Framework

Let $U = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_j, ..., \mathbf{x}_n\}$ be an unlabeled dataset with features $F = \{f_1, f_2, ..., f_d\}$ and a set of contexts $\hat{C} = \{C_1, C_2, ..., C_m\}$,

$C \in F$ and $B = F \setminus C$ the behavioural features. The algorithm starts by estimating clusters (reference groups) $R(x_j, C_i)$ for each instance x_j w.r.t each C_i. This can be done by using any clustering algorithm suitable for the data set. Afterwards, using isolation forest trees the deviation of the instances x_j to their cluster is computed. This is achieved by creating M separate isolation trees for each cluster in C_i and using the respective behavioural attributes B_i the abnormal instances are isolated, i.e isolate those instances whose behaviour differ by far from their cluster group. The result from the iForest of all the clusters in the context C_i is the set of anomaly scores S_i, where $s_{i,j} \in S_i$ is the anomaly score of x_j in the context C_i and $|S_i| = n$. However, since the base detectors are applied on different contexts, the resulting scores vary in range and scale. Therefore, these are converted into probabilities.

On the second step of the framework, a query strategy is performed using the resulting scores and provide the most informative instances to the analyst for labelling. It is assumed that the analyst is able to categorize b instances and after each iteration, the importance scores of the contexts in $\hat{C}$ are updated in the model taking into account the feedback received. The importance of a context C_i quantifies how good this one is for discovering anomalies. This step aims to maximize the expected information gain of x_j, which is achieved by the type of query strategy performed. On this step the labeled instances belong to the set L, where $L = \{(x_1, y_1), ..., (x_t, y_t)\}$ at iteration t and $y_j \in \{0, 1\}$ is the label received.

On the third step, once an instance has been labeled by the analyst, the importance or weights of the contexts have to be updated. This task is carried out using the AdaBoost weighting scheme, which estimates the classification error of the base detectors to determine their usefulness.

Firstly the detection error is calculated by converting the score of the jth instance in context C_i into predictions, i.e transform the probability scores $S_i = \{s_{i,1}, s_{i,2}, ..., s_{i,n}\}$ into a hard label $P_i = \{p_{i,1}, ..., p_{i,n}\}$ where $p_{i,j} \in \{0, 1\}$ and 1 means that the ensemble classifies an instance as an anomaly. The hard label $p_{i,j}$ is calculated using a threshold between anomalies and nominals, e.g 0.9 keeping 0.1 as the false alarm rate. Afterwards, with the labeled data set L, the detection error ϵ for the context C_i at iteration t is:

$$\epsilon_{i,t} = \frac{\sum_{j=1}^{t} \theta_j l_{i,j}}{\sum_{j=1}^{t} \theta_j}, \qquad (3),$$

where θ_j is the weight for each point in the labeled set, $x_j \in L$, and $l_{i,j}$ is 0 if $p_{i,j} = y_j$, i.e if the prediction is the same as the true label and 1 otherwise. The weights θ_j depend on the query strategy used, if no differences between the instances are assumed then $\theta_j = \frac{1}{t}$ for each $x_j \in L$. With the detection error, it is possible to calculate the importance of the context as:

$$I_i = \frac{1}{2} ln(\frac{1 - \epsilon_{i,t}}{\epsilon_{i,t}}), \qquad (4)$$

Negative results of I_i indicate that the context performs worse than a random detector, with a detection error greater than 0.5. The calculation of the weights θ_j, detection error $\epsilon_{i,t}$ and importance I_i are recalculated every time a new instance is labeled by the analyst, added to the set L and removed from the unlabeled set U.

The last part of the algorithm consists on pruning the irrelevant contexts and updating the final anomaly scores. Pruning consists in removing the contexts that have a negative importance $I_i < 0$.

With the remaining p contexts and respective scores S_p, the final anomaly scores for the instances are calculated as:

$$s_j = \frac{\sum_{i=1}^{p} I_i \times s_{i,j}}{\sum_{i=1}^{p} I_i}, \qquad (5)$$

where $p = |S_p|$, and $s_{i,j} \in S_i$ is the anomaly score of x_j in context C_i. This result is a weighted sum of the importance and scores of the contexts, which is employed in the next iterations for finding the most informative instances.

3.2 Concept drift adaption using up to date and delayed feedback

Regarding the SAL algorithm for streaming data, the selection of instances given to the analyst is based on providing the most anomalous instances within the new and already stored data points. This process does not guarantee that the chosen samples correspond to up to date observations consistent with the new distribution of the data or if on the other hand, the queried instances have the highest anomaly scores but belong to past windows. Thus, although the most anomalous instances might be selected, these might not be good representatives of the current reality, e.g changes in behaviour of a fraudulent customer.

[3] proposes a fraud detection model for streaming data in which, delayed and up to date feedback are combined in order to provide accurate alerts to the analysts. In this model as well, the goal is to maximize the number of true positives, since otherwise the analysts might decide to ignore the incoming fraud alerts thrown by the model. The idea of this model consists on building a classifier, e.g Random forest, only with the instances of the new window t based on the latest binary classifier $K_{t-1} : R^n \longrightarrow \{+, -\}$ and the z, $z > 0$ transactions with highest probability of being an anomaly are given to the analyst to label, so $F_t = \{(\mathbf{x}, y), \mathbf{x} \in z\}$. t is assumed by the authors to be a window consisting of 1 day. Delayed feedback from instances older than γ days is also received at the time point t, i.e $D_{t-\gamma} = (\mathbf{x}, y), \mathbf{x} \in T_{t-\gamma}$, where $T_{t-\gamma}$ are the previous γ days. From this approach it can be noticed that the distribution of $D_{t-\gamma}$ is skewed towards the nominal instances, since this is the predominant class in a data set, while F_t is ideally skewed towards anomalies, since this is the class aimed to be presented to the analyst, but it can as well contain false positives. An ensemble of classifiers $\varepsilon_t^D = \{M_1, ..., M_\alpha\}$ is then used for training $D_{t-\gamma}$,

where each individual classifier M_i is trained on $D_{t-\gamma-i}$, $i = 1, ..., \alpha$, obtaining $|M| = \alpha$ classifiers. The anomaly score of this ensemble is obtained by averaging the probability scores of the individual classifiers. Finally the classifier for F_t and the ensemble ε_t^D are used to obtain the final probability scores of all instances as:

$$P_{A_t^\varepsilon} = \frac{P_{F_t}(+|\mathbf{x}) + P_{\varepsilon_t^D}(+|\mathbf{x})}{2}, \qquad (6)$$

where A_t^ε represent the set of alerts and P_{F_t} and $P_{\varepsilon_t^D}$ are the scoring probabilities being the posterior probability of obtaining an anomaly given x for subset F_t and $D_{t-\gamma-i}$ respectively. This calculation gives a higher weight to the classifiers using F_t, i.e to the most recent instances. Both classifiers are built separately and not with the complete set, because the classifier trained on F_t learns how to label transactions that are most likely to be anomalous but might not be precise with the majority of nominal instances.

4 Query strategies

4.1 Select-Diverse

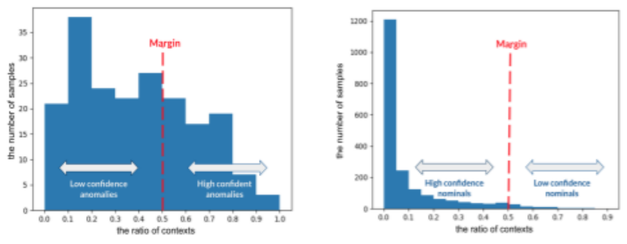
In section 2.1 it is showed that one of the techniques employed for the selection of instances for the analyst to label is taking the highest ranked data points. However, this strategy might lack diversity in the type of anomalies presented, since most of them might come from the same subspace (leaf node). Hence, the authors of BAL [4] propose a strategy to avoid this, called "Select-Diverse", which aims for looking at instances coming from subspaces having minimum overlap using Equation 1 of subsection 2.2. The process starts by getting the top ranked instances Z from the ensemble and their respective compact subspaces S^* (Equation 1). Having an empty vector Q, it takes the highest ranked instance of Z, adds it to Q and removes it from Z. On the next iteration, the algorithm takes again the highest ranked instance of Z having minimum overlapping with the instances in Q, and subsequently adds it to Q, removing it from Z. This process is repeated until $|Q|$ is equal to the amount of instances b given for labelling to the analyst. The resulting subset contains only instances being the most anomalous and having minimum overlap.

Select-Diverse however, does not guarantee that the most informative instances are used in the algorithm to learn correctly the decision boundary, since it takes into account the anomaly score but not how relevant is the subspace where these are found for discovering new anomalies.

4.2 Low Confidence Anomaly sampling

The authors of WisCon [1] propose a different strategy called "Low confidence anomaly sampling" (LCA) that aims to select instances that maximize the information gain on the relevance of the contexts. This follows the same approach as Query-by-Committee, in which the samples that generate the maximum disagreement among the committee models, e.g base detectors, are given to the analyst. LCA assumes that there are multiple contexts unveiling anomalies, but these are rare among all possible contexts of the feature set.

Figure 4.a and 4.b show the distributions of the true anomalies and true nominal instances. On Figure 4.a it is observed that many of the true anomalies are only scoring as anomalies in less than 20% of the contexts (left side of the distribution), so these are catalogued as low confidence anomalies because only a minority of contexts were able to unveil them. Thus, querying instances from a context belonging to this group of detectors, is most likely a good context and should have a high importance score. At the same time, Figure 4.b shows that on the left side are located the high confidence nominals, hence querying instances on the left side of the distribution would lead to query normal instances rather than anomalies since these are usually rare in data sets.



(a) Anomalous samples (b) Normal samples

Figure 4. True anomalies and normal instances

To select low confidence anomalies, LCA selects data points around the margin (e.g anomalies recognized in 50% of the contexts) using the importance scores of the contexts. The margin rate of an instance x_j, quantifies the closeness of x_j to the margin of the distribution as follows:

$$\text{margin}(x_j) = 100 \times (1 - |\frac{2\sum_{i=1}^{m} I_i \times p_{i,j}}{\sum_{i=1}^{m} I_i} - 1|), \qquad (7)$$

where I_i is the importance of the context C_i and $p_{i,j} \in \{0, 1\}$ is the hard label assigned to x_j using $s_{i,j}$ (subsection 3.1) , such that $\text{margin}(x_j) \in [0, 100]$. The low confidence anomaly sampling measure is defined as:

$$\text{Q}_{LCA} = \text{argmax} \frac{\exp(\lambda \times \text{margin}(x))}{u_x}, \qquad (8)$$

where λ is the bias factor controlling how influenced the sampling is towards margin rates, e.g $\lambda = 0$ is equal to random sampling. A reasonable bias factor is between 0.96 and 0.98. Moreover, u_x is an independently and uniformly distributed random variable drawn for each x. Q_{LCA} gives the instances with higher margin rates, a higher probability of being selected.

After the analyst labels a data point, the importance of the contexts and the margin rates are updated in order to increase the margin rates of low confidence anomalies. To avoid selecting normal samples, different weights θ_j are given to the instances after receiving their true label from the analyst. Given a sample $(x_j, y_j) \in L$ the weight θ_j of x_j is equal to $\text{margin}(x_j)$ if $y_j = 1$, i.e if it is a true anomaly, otherwise the weight is 0 (when it is a nominal). With this approach, the impact of normal data points is eliminated from the importance scores of the contexts. Hence, less informative instances have less impact on I_i even if they are sampled.

5 Conclusions

This report studied how ensemble methods and active learning are used in anomaly detection. Ensemble methods provide more reliable results compared to a single detector, since these are less susceptible to imbalanced data. On this paper it was introduced the Isolation Forest for anomaly detection. This ensemble method diverges from other models, since it does not search for profiling normal instances but isolating instances with extreme values of an attribute by repeteadly partitioning the feature space and choosing random split points of the attribute until all instances reach a leaf node. Furthermore, active learning is a strategy that consists on providing a small set of instances from a large data set to an oracle, i.e analyst, so that this can judge if the data point is an anomaly or a normal instance and subsequently forward the feedback to the model. This process is useful when there is abundant data and labelling all of the instances is too expensive. Good examples of instances need to be provided to the analyst, so the model can learn efficiently the decision boundary for recognizing true anomalies. For this reason, the query strategy chosen has a high relevance on this process. On this paper two strategies were presented, Select-Diverse consists on giving the analyst the most anomalous instances with the least possible subspace overlap, so that different types of anomalies are showed. The second one, Low Confidence Anomalies (LCA) chooses those anomalies which are identified only in a few possible scenarios, i.e contexts, with the idea to explore the capacity of each context to provide true anomalies.

On this report, two algorithms using isolation forest and active learning are studied; BAL (Batch Active learning) consists on running multiple trees and taking the leaf nodes as the base detectors for the model. The instances get a score by each detector, calculated as the length path from the root to the leaf and these are multiplied by the weight, i.e relevance, of each detector to get the total anomaly score. The weights of the leaves are computed by minimizing the amount of errors made by the model when assigning low scores to true anomalies and high scores to true normal points. WisCon on the other hand, considers that anomalies might be determined as such depending on the context they are being analysed, suggesting that having only a global perspective might hide true anomalies. Hence, this method clusters instances with respect to their context and calculates the isolation forest based on each cluster and their behavioural attributes. The importance, i.e weights of the contexts are calculated using a strategy similar to AdaBoost, taking into account the missclassified instances based on the analyst's feedback and the type of instance predicted (true anomaly or nominal). Only the contexts with a few amount of false positives are retained. We consider that WisCon, is a model with a broader application field since in real life problems, usually considering the full space of attributes to evaluate the data does not provide as much information, as just focusing on a few. A similar problem occurs with full space and subspace clustering, in which taking into account all attributes as equally important leads to deteriorating the accuracy of the clusters, since small sized clusters might be overlooked. In WisCon, as the instances are first grouped taking into account their contextual attributes, e.g environmental factors, and then evaluated based on their behaviour in that context, it is easier to find instances with abnormal patterns. However, determining what a context is can be a difficult task, since only considering random combinations of attributes or using PCA might not be a sensible approach.

Anomaly detection using active learning can be extended to streaming data. One challenge faced when building models for streaming data is that the distribution of the data might change during the course of time. Hence, the learning algorithm can become obsolete and fail to recognize new anomalies. For this reason, every time a new window of information arrives, the algorithm SAL (Streaming Active Learning) evaluates if there is a change in the distribution of the data, and replaces the obsolete information with the one that has changed. Afterwards, it uses the same approach as BAL to calculate the scores of the anomalies and update the weights of the detectors. Another approach, consists on implementing a Random Forest on the new data, using the latest classifier, and combining its predictions with a model built on the previous n days, to provide a set alerts with anomalies that should be checked in detail. The methods presented on this paper are an example of different strategies for implementing active learning in anomaly detection for static and streaming data. Further studies can evaluate a feasible approach for determining how a context can be defined. In terms of the definition of compact subspaces, it can be studied if it is a reasonable approach to penalize a subspace based on the amount of nominal values in this.

References

[1] E. Calikus, S. Nowaczyk, M. Bouguelia, and O Dikmen. 2021. Wisdom of the Contexts: Active Ensemble Learning for Contextual Anomaly Detection. (2021). https://doi.org/workingpaper

[2] Antonio Clim, Răzvan Daniel Zota, and Grigore TinicĂ. 2018. The Kullback-Leibler Divergence Used in Machine Learning Algorithms for Health Care Applications and Hypertension Prediction: A Literature Review. *Procedia Computer Science* 141 (2018), 448–453. https://doi.

org/10.1016/j.procs.2018.10.144

[3] A. Dal Pozzolo, G. Boracchi, O. Caelen, C. Alippi, and G. Bntempi. 2015. Credit Card Fraud Detection and Concept-Drift Adaptation with Delayed Supervised Information. (2015), 1–8. https://doi.org/10.1109/IJCNN.2015.7280527

[4] S. Das, M. Islam, N. Kannapan, and J. Doppa. 2019. Active Anomaly detection via Ensembles: Insights, Algorithms and Interpretability. *School of EECS, Washington State Univeristy* (2019). https://doi.org/1901.08930v1

[5] Hemant Ishwaran, Udaya B. Kogalur, Eugene H. Blackstone, and Michael S. Lauer. 2008. Random survival forests. *The Annals of Applied Statistics* 2, 3 (2008), 841 – 860. https://doi.org/10.1214/08-AOAS169

[6] Bernhard Korte and Jens Vygen. 2012. *Combinatorial Optimization: Theory and Algorithms* (5th ed.). Springer Publishing Company, Incorporated.

[7] F. Liu, K. Ting, and Z. Zhou. 2008. Isolation forest. *Eighth IEEE International Conference on Data Mining* (2008), 413–422. https://doi.org/10.1109/ICDM.2008.17

[8] Burr Settles. 2009. Active Learning Literature Survey. (2009).

[9] Alexey Tsymbal. 2004. The Problem of Concept Drift: Definitions and Related Work. (05 2004).