Rule Extraction in Unsupervised Outlier Detection for Algorithmic Transparency: Application to OneClass SVM

Alberto Barbado

Telefónica – LUCA Data Unit.

Ronda de la Comunicación, s/n, 28050,

Madrid, Spain

alberto.barbadogonzalez@telefonica.com

ABSTRACT

OneClass SVM is a popular method to perform unsupervised outlier detection on features that compose a dataset. However, this method is generally considered a *black box* due to the fact that it's difficult to justify, in an intuitive and simple way, why the decision frontier is classifying the data points in the different categories. This problem appears in supervised classification tasks and even more in unsupervised learning for anomaly detection, where the data points are labelled as outliers or not outliers. To be able to obtain those intuitive explanations from that algorithm this article proposes a method to infer rules that justify why a point is labelled as an outlier.

1. INTRODUCTION

Fair AI is an emerging discipline that is gaining more relevance both in academic research and in industry, as nowadays more companies are defining their policies and criterias for the usage of data and the development of support decision systems. This matter can be addressed, for instance, with the definition of what is known as *AI principles* that could serve as a guide for it, like it is been done in Telefónica [1].

These principles address several issues regarding those sustainable developments that could be condensed in the following categories.

One consists in detecting sensible data in the datasets used to train machine learning models (for example, the use of *gender* information to support the decision of giving a credit score) or detecting if there is a bias in the available data that could affect the conclusions of the system. Besides analysing directly the dataset, there are also methods to perform an evaluation on an already trained model to check if there is any bias on it's decisions. This is done using a group of metrics known as *fairness metrics*.

Another issue would be explaining how an algorithm reaches a conclusion in a way that is clear and intuitive for a human being. This is crucial not only to avoid the fear of considering AI as *black boxes* that could suddenly make harmful decisions in the future, but also to contribute to the democratization of this technology and

increase the trust that users give to these systems.

The first issue mentioned is heavily addressed right now from many researchers that are creating tools to audit both datasets and trained models to detect those risks mentioned and at the same time they provide solutions to mitigate those bias discovered [6], [7], [8].

Due to that, this work will be focused on the second area, dealing with algorithmic transparency.

2. OUTLIER DETECTION WITH UNSUPERVISED MODELS

2.1. INTRODUCTION

There are many machine learning models that could potentially be considered *black boxes* and because of that would need solutions to explain how they reached a conclusion, but this is even more necessary in the case of unsupervised learning algorithms because they tend to be an even *blacker* box than other methods due to the fact that the output of the system is not linked to an a priori knowledge like it does with supervised learning.

This is critical, for example, in the case of unsupervised anomaly detection where data points are classified as *regular* data or *outliers*. It of paramount importance to be able to justify why a data point is labeled as such, and that is why the research in applying algorithmic transparency for those algorithms is fundamental.

There are many algorithms to perform unsupervised outlier detection, being the most popular the *IsolationForest*, the *Local Outlier Factor (LOF)* and the *OneClass SVM*. OneClass SVM has huge advantages over the other ones, mainly the computational performance due to the fact that it creates a decision frontier using only the support vectors (like general supervised svm) and that the training of the model leads always to the same solution because the optimization problem is a convex one.

However, SVMs and OneClass SVMs by extension are the most difficult algorithms to explain due to the mathematically complex method that obtains the decision frontier, and in the scenario of unsupervised learning described, OneClass SVM is clearly the most opaque algorithm of the three.

2.2. ONECLASS SVM INTUITION

SVMs used in classification theoretically map the data points available in the data set to a higher dimensional space than the one determined by their features where the separation of the different classes could be linearly done, using to do it a hyperplane obtained from data points from all of the classes. These data points, known as *support vectors*, are the ones that are closer to each other and the only ones needed to determine the decision frontier. However, is not really necessary to map to a higher dimension due to the fact that the equation that appears in the optimization of the algorithm uses a dot product of those mapped points. Because of that, the only things that needs to be calculated is that dot product and that could be accomplished with the well-known *kernel trick*, so instead of calculating explicitly the mapping to a higher dimension the equation is solved using a kernel function.

In OneClassSVM [9] there are no labels, and because of that at the beginning all points could be considered belonging to a same class (hence the name). The decision frontier is computed trying to separate the region of the hyperspace that has a high number of data points close to each other from another region that has small density, considering those points as outliers. To do so the algorithm tries to define a decision frontier that maximizes the distance to the origin of the hyperspace and that at the same time separates from it the maximum number of

data points. This compromise between those factors lead to the optimization of the algorithm and allows to obtain the optimal decision frontier. Those data points that are separated are labeled as *normal data* (+1) and those who don't are labeled as *outliers* (-1).

The optimization problem is reflected in the following equations:

$$min_{w,\,\xi_i,\,\rho}\ \tfrac{1}{2}\,\|w\|^2\ +\ \tfrac{1}{\nu n}\sum_{i=1}^{n}\xi_i\ -\ \rho$$

subject to:

$$(w.\phi(x_i)) \geq \rho - \xi_i \ for\ i = 1, ..., n$$

$$\xi_i \geq 0 \ for\ i = 1, ...,\ n$$

In that equation, ν is an hyper-parameter know as *rejection rate.* It needs to be selected by the user and that sets an upper bound on the fraction of outliers that can be considered. It also defines a lower bound on the fraction of support vectors that can be considered.

Using Lagrange techniques the decision frontier obtained is the following one:

$$f(x)\ =\ sgn((w.\phi(x_i))\ -\ \rho)\ =>$$

$$f(x)\ =\ sgn(\sum_{i=1}^{n}\alpha_i K(x_i, x)\ -\ \rho)$$

Because of that, the hyper-parameters that would be defined in this method will be the ν rejection rate and the ones that appear as usual in the kernel function used.

3. RULE EXTRACTION IN ONECLASS SVM

Different articles already published [3] deal with the importance of algorithmic transparency applied to models such as SVMs. That article mentioned in fact covers different techniques to do so in SVMs trying to resolve the *black box* problem in SVMs for supervised classification tasks.

The way proposed to do so is by obtaining a set of rules that explain in a simple way what are the boundaries that contain the values of the different classes. Thanks to that anyone can understand what are the conditions that will identify a data point as belonging to one class or to another (in oneclass svm will be belonging to class +1, normal data, or class -1, outlier). The challenge consists in discovering a way to map the results from the algorithm to a particular set of rules.

There are two general ways to do it. One consists in inferring the rules directly from the decision frontier obtained using a decompositional rule extraction technique. The other technique, more simple, uses a method called pedagogical rule extraction technique that does not care about the decision frontier itself and considers the algorithm as a black box from where to extract those rules depending on which class it uses to classify different relevant data points used as an input.

The first method is clearly more transparent as it deals directly with the inner structure of the model, but on the other hand is traditionally more difficult to implement. On way to implement it is a technique known as SVM+ Prototypes [2] that consist in finding hypercubes using the centroids (or prototypes) of data points of each class and using as vertices the data points from that hyperspace area that are farther away from that centroid. It will then infer a rule from the values of the vertices of the hypercube that contain the limits of all the points inside it, creating one rule for each hypercube.

For example, a dataset that contains two numerical features X and Y will be defined in a 2 dimensional space and for that case the algorithm will create a square that contains the data points on each of the classes. This can be seen in the following example:

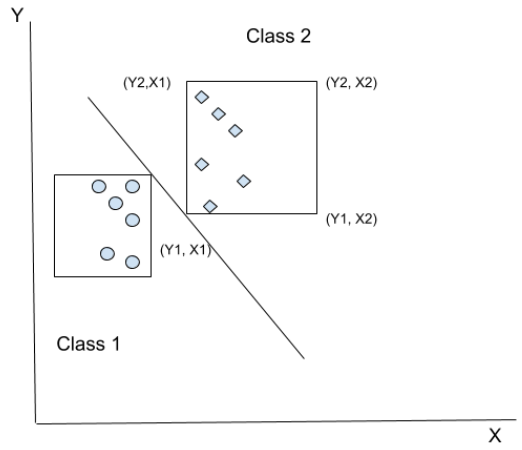

Fig 1. SVM with linear kernel classifying data points of two classes

In this example, the rule that would justify that a data point would belongs to class 2 will be the following one:

- Rule 1: CLASS 2... IF X >= X1 AND Y >= Y1 AND X <= X2 AND Y <= Y2

The downside of that method is that in supervised classification tasks, specially where the SVM uses soft-margin technique, the subareas of the hyperspace that contains the data points classified for each class could potentially contain points that are misclassified, so the SVM+ Prototypes propose to iteratively create different and smaller hypercubes on each of the subregions until the hypercubes created contain (mostly) only points belonging to the correct class, and that leads to the generation of a vast number of rules that make the output very difficult to understand, even more when the number of features that define the hyperspace are enormous.

However, in the scenario of OneClass SVM these difficulties don't appear. In this case there is only one hypercube due to the fact that is an unsupervised learning algorithm, which means there are no misclassified points, and because the system only needs to justify why a data point is an outlier it does not need to use more than one hypercube.

The created hypercube could be use to infer a rule to indicate why a datapoint is an anomaly (hypercube for the outliers) or when a data point is not an anomaly (hypercube of the data not labeled as an outlier). In general is better to use the second approximation due to the fact that the first one will fail to give the adequate limits when the kernel used is RBF because in this case the normal data would be surrounded by the decision frontier and the outliers will be outside of it.

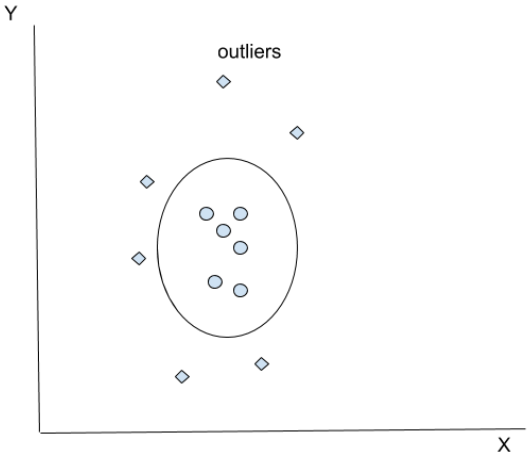

Fig 2. With a RBF Kernel the correct hypercube will be the one that encloses the points that are not outliers due to the fact that the OneClass SVM algorithm will try to enclose most of the points inside the decision frontier and leave the outliers outside.

It is true that for other kernels, such as the linear one, both approximations would be correct, but to define a general approximation, the library will indicate when a data point is not an outlier.

Thanks to all that the rule will always be inferred from the vertices of one hypercube only, making it a reasonable approach that always leads to one rule.

A caveat that should be considered is the scenario in where some of the features are categorical not ordinal variables. In that case the approximation would be to extract a rule for each of the possible combinations of

categorical values among the data points not considered anomalous.

Considering again the 2 dimensional example previously mentioned but in this case the variable X being binary categorical, a dataset could look like the following image:

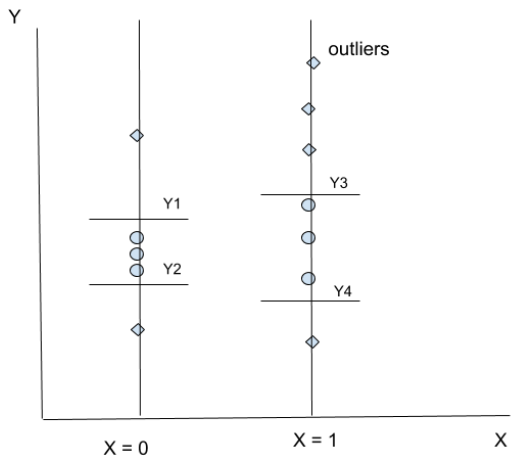

Fig 3. Rule extraction when there is a categorical variable.

In that case, the rules extracted would be two, one for each of the possible states of X:

- Rule 1: NOT OUTLIER... IF X = 0 AND Y >= Y2 AND Y <= Y1
- Rule 2: NOT OUTLIER... IF X = 1 AND Y>=Y4 AND Y<=Y3

So the number of rules extracted would be:

$$n^{o}\ of\ rules\ =\ 1\ +\sum_{i}^{N} n_i \times s_i$$

with n_i each of the categorical variables and s_i the number of states of that variable that are contained in the data labeled with +1 (not anomalous).

4. IMPLEMENTATION

4.1. CURRENT DEVELOPMENT

To be able to automatically infer the rules a library has been developed wrapping Python's Scikit-learn OneClass SVM class. This library can detect outliers and extract the rules explaining them using any data set that can include both numerical or categorical variables. One way to do that could be using a clustering method known as K-Prototypes that can deal with this scenario and that then computes the distance from the centroid using a distance measure that takes into account both the numerical and the categorical values.

Another simpler way is to create sub-hypecubes for the numerical values of each of the categorical states contained in the not anomalous points. In that way the centroid for that sub-state can be obtained using a distance measure such as Euclidean or Manhattan.

Documentation and code of this library can be found at [10].

The algorithm logic can be resumed in:

- Apply OneClass SVM to the dataset available
- Act accordingly depending on whether there are categorical variables, numerical or both:
    - Case 1. Numerical only: There is only one hypercube for all data
    - Case 2. Categorical only: The rules will correspond directly to the different value states contained in the dataset of not anomalous points. END
    - Case 3. Both numerical and categorical. In this case there would be a sub-hypercube for the numerical variables filtered in each of the sub-states of the categorical variables.

- For cases 1 and 3, calculate the centroid of the hypercube or sub-hypercube of that iteration.

- Identify the points farther away from that centroid and extract a number of the corresponding to the number of vertices that a hypercube has in that hyperspace (2*number of features)
- Use this vertices to obtain the boundaries of that hypercube and directly extract rules from the vertices.
- For case 1, END. For case 3 repeat for each of the categorical states.

The outcome of the system looks like this:

NOT anomaly...
Rule Nº 1: IF sex = 0 AND school = 0 AND studytime <= 4 AND G3 <= 15 AND studytime >= 1 AND G3 >= 8
Rule Nº 2: IF sex = 0 AND school = 1 AND studytime <= 2 AND G3 <= 0 AND studytime >= 2 AND G3 >= 0
Rule Nº 3: IF sex = 1 AND school = 0 AND studytime <= 4 AND G3 <= 13 AND studytime >= 2 AND G3 >= 8

4.2. POSSIBLE ALTERNATIVES

A question that can arise immediately is why not using directly the support vectors to obtain the rules. The support vectors could be identified directly thanks to some algorithms such as the OneClass SVM implementation of Scikit-Learn, as well as the distance to the decision frontier. However this approximation has one potential issue,

1. It depends on the availability of that information. Not every library gives that result. The implementation proposed in this article can deal with a more general scenario where the only information available is whether a data point is labeled as anomalous or not anomalous. That will be all the information needed to explain the rules for outlier detection.
2. The rule extraction logic would heavily depend on the kernel used. For example, with a linear kernel the support vectors would only lead to infer one bound (upper or lower) for the rest of the data points. However, with a RBF the support vectors would be potentially used to infer rules for both boundaries due to the fact that the decision frontier could surround the data points. On the contrary, the other method proposed is agnostic to the type of kernel because it will always create a hypercube for the data points of each class.

For all those reasons listed and in order to define a more general approximation it has been found preferable to use the implementation mentioned instead of using the support vectors.

5. CONCLUSION

In this article it's been shown the necessity of using FAIR AI techniques to contribute to the transparency of AI systems and in particular using rule extraction techniques to transform the traditional *black boxes* into solutions that are transparent. Because of that a set of simple rules can be produced explaining why an algorithm is performing a particular classification of a data point.

This has been done developing a library using the OneClass SVM outlier detection algorithm and extracting rules using the data points not labeled as anomalous. This implementation can work in datasets containing numerical variables, categorical or both.

6. REFERENCES

[1] R.Benjamins. *De los principios de IA a una IA responsable.* 2018. Retrieved from https://www.telefonica.com/es/web/public-policy/blog/articulo/-/blogs/de-los-principios-de-ia-a-ia-responsable

[2] H. Núñez, C. Angulo, and A. Català. Rule extraction from support vector machines. In

European Symposium on Artificial Neural Networks (ESANN), pages 107–112, 2002.

[3] D. Martens, J. Huysmans, R. Setiono, J. Vanthienen, and B. Baesens. Rule Extraction from Support Vector Machines: An Overview of Issues and Application in Credit Scoring. 2008.

[4] Huang, Z.: Extensions to the k-modes algorithm for clustering large data sets with categorical values, Data Mining and Knowledge Discovery 2(3), pp. 283-304, 1998.

[5]https://github.com/AlbertoBarbado/unsupervised-outlier-transparency

[6] Rachel K. E. Bellamy, Kuntal Dey, Michael Hind, Samuel C. Hoffman, Stephanie Houde, Kalapriya Kannan, Pranay Lohia, Jacquelyn Martino, Sameep Mehta, Aleksandra Mojsilovic, Seema Nagar, Karthikeyan Natesan Ramamurthy, John Richards, Diptikalyan Saha, Prasanna Sattigeri, Moninder Singh, Kush R. Varshney, Yunfeng Zhang. AI Fairness 360: An Extensible Toolkit for Detecting, Understanding, and Mitigating Unwanted Algorithmic Bias. 2018.

[7] M. Ward et al. *Audit-AI*. 2018. Retrieved from https://github.com/pymetrics/audit-ai

[8] Pedro Saleiro, Benedict Kuester, Abby Stevens, Ari Anisfeld, Loren Hinkson, Jesse London, Rayid Ghani, Aequitas: A Bias and Fairness Audit Toolkit, arXiv preprint arXiv:1811.05577. 2018.

[9] Bernhard Scholkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, John Platt. Support Vector Method for Novelty Detection. MIT Press (2000). S.A. Solla, T.K. Leen and K.-R. Müller (eds.), 582–588.

[10] A. Barbado. *Rules extraction in unsupervised outlier detection for algorithmic transparency*. 2018. Retrieved from https://github.com/AlbertoBarbado/unsupervised-outlier-transparency