Dydaktyka Informatyki 20(2025)
ISSN 2083-3156; e-ISSN 2543-9847
DOI: 10.15584/di.2025.20.16
https://www.journals.ur.edu.pl/di

Uniwersytet Rzeszowski

Jacek WOŁOSZYN[1], **Michał WOŁOSZYN**[2]

[1] *ORCID: 0000-0003-4340-9853. Dr inż., Uniwersytet Radomski, Wydział Informatyki i Matematyki, Katedra Informatyki, ul. Malczewskiego 20A, 26-600 Radom, e-mail: jacek.woloszyn@uthrad.pl*
[2] *BSc, student,Goldsmiths, University of London, 8 Lewisham Way, London SE 14 6NW, e-mail: mwolo001@gold.ac.uk*

data złożenia tekstu do Redakcji DI: 18.05.2025; data wstępnej oceny artykułu: 27.05.2025

THEORETICAL CONSIDERATIONS ON ARTIFICIAL INTELLIGENCE AND CYBERSECURITY, ONE-CLASS SVM FOR ANOMALY DETECTION IN NETWORK TRAFFIC

ROZWAŻANIA TEORETYCZNE NA TEMAT SZTUCZNEJ INTELIGENCJI I CYBERBEZPIECZEŃSTWA, ONE-CLASS SVM DO WYKRYWANIA ANOMALII W RUCHU SIECIOWYM

Keywords: Python, artificial intelligence, cybersecurity, One-Class SVM.
Słowa kluczowe: Python, sztuczna inteligencja, cyberbezpieczeństwo, One-Class SVM.

Abstract

In this article, One-Class SVM has been highlighted as a particularly valuable approach, as it does not require collecting large datasets of labeled attack samples. Instead, it effectively models normal behavior and identifies significant deviations from the expected pattern.

Both theoretical considerations and a Python code example have been presented, demonstrating how such a model can be trained on real or synthetic network data and subsequently used to detect potential anomalies.

Additionally, the text includes guidelines for data preparation, covering collection, cleaning, normalization, and potential dimensionality reduction, as well as hyperparameter optimization (including nu and gamma).

Furthermore, a mathematical perspective is provided, explaining the role of the weight vector (w), the threshold value (R), and the kernel function, which enables nonlinear mapping and the separation of normal samples from outliers.

Streszczenie

W niniejszym artykule skupiono się na One-Class SVM jako szczególnie wartościowym podejściu, gdyż nie wymaga ono gromadzenia dużych zbiorów etykietowanych próbek ataków, natomiast potrafi opisać normalne zachowania i rozpoznać każde istotne odchylenie od wzorca.

Przedstawiono rozważania teoretyczne i zaprezentowano przykład kodu w Pythonie demonstrujący, w jaki sposób można w praktyce trenować taki model na rzeczywistych lub syntetycznych danych sieciowych, a następnie wykorzystywać go do oznaczania potencjalnych anomalii.

W tekście zawarto ponadto wskazówki dotyczące przygotowania danych (od zbierania poprzez czyszczenie, normalizację i ewentualną redukcję wymiarowości), jak również optymalizacji hiperparametrów (m.in. nu, gamma).

Omówiono też, w ujęciu matematycznym, na czym polega rola wektora w, wartości progowej R oraz funkcji jądra (kernel), która umożliwia nieliniowe odwzorowanie i oddzielenie normalnych próbek od odstających.

Introduction

In the article "Advanced Artificial Intelligence Methods in Cybersecurity, Threat and Anomaly Detection Using Unsupervised Learning Techniques", the importance of anomaly detection in the context of cybersecurity is discussed, along with various approaches and algorithms, including unsupervised learning techniques such as One-Class SVM and Isolation Forest. It is highlighted that these tools can be particularly effective when there is an insufficient number of attack samples available or when an attack has a completely new, previously unknown nature (zero-day attacks).

This text presents the transition from theory to practice by demonstrating a detailed example focusing on One-Class SVM as an anomaly detection model for network traffic.

One-Class SVM has gained popularity in network security applications because it does not require access to labeled attack samples. It only needs a sample of "normal" network traffic, and the model learns the characteristic behavioral patterns. Subsequently, anything that deviates from the boundary established by the algorithm is marked as a potential anomaly.

This approach is particularly effective in environments[1] where many resources and processes undergo continuous evolution, and attacks take on ever-changing forms.

[1] Hu, Jing, et al., *A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks*, IEEE Access, Vol. 6, 2018; Ngo, Quang-Hung, et al., *Unsupervised Deep Learning for Real-Time Cyber Anomaly Detection*, IEEE Access, Vol. 7, 2019.

1. Environment and Data Preparation

Environment and data preparation[2] begins with selecting the appropriate tool for experiments and providing a representative dataset of network traffic information. Typically, work is conducted in a Python 3 environment with support from popular libraries such as NumPy, Pandas, and Scikit-learn, as they offer comprehensive functionalities for data loading, preprocessing, and analysis, as well as implementing machine learning algorithms.

The first step is ensuring that all necessary packages are installed. This can be done using the pip package manager or Anaconda, which helps maintain consistent library versions and minimizes conflicts between dependencies.

On the data side, the initial task is to collect or acquire a dataset describing both normal network traffic and potential anomalies. The easiest approach is to use publicly available datasets, such as KDD Cup '99, UNSW-NB15, or other benchmark datasets containing recorded traffic traces with annotations indicating normal or attack-related activity. If public datasets cannot be used, or if there is a need to train the algorithm on a specific environment, it is necessary to generate custom logs and metadata from network monitoring systems like Zeek (formerly Bro), Suricata, or firewall log analysis tools.

Once data is gathered, it should be standardized, which includes format conversion—usually to CSV or Parquet, where each column represents a single feature. At this stage, attention should be paid to the range and scale of attributes, such as packet count per flow, session duration, or protocol type. It is often useful to handle missing values or remove erroneous records (e.g., those containing invalid source port values).

After eliminating undesirable entries, preprocessing is performed, applying normalization or standardization techniques such as MinMaxScaler or StandardScaler to ensure that features have similar value ranges and prevent the model from being biased toward attributes with large scales. The next step may involve dimensionality reduction (e.g., PCA), especially if the dataset contains dozens or hundreds of columns, which can complicate One-Class SVM training.

For large datasets, sampling techniques are often applied, selecting a random subset of observations to accelerate training and facilitate concept validation. Finally, the dataset should be split into training, validation, and test

[2] I. Goodfellow, P. McDaniel, N. Papernot, *Making Machine Learning Robust Against Adversarial Inputs*, Communications of the ACM, Vol. 61, No. 7, 2018; M. López-Martín, B. Carro, A. Sánchez-Esguevillas, J. Lloret, *Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things*, IEEE Access, Vol. 5, 2017 (updated online in 2018); I. Sarker, et al., *IntrudTree: A Machine Learning Based Cyber Security Intrusion Detection Model*, "Symmetry" 2021, Vol. 13, No. 4.

sets, enabling both optimal parameter selection for One-Class SVM[3] and evaluating its effectiveness on unseen samples.

This logical sequence of steps, from environment setup, data collection, and preprocessing to proper segmentation, provides a solid foundation for further work on anomaly detection in network traffic.

2. Model One-Class

One-Class SVM is designed as an anomaly detection method in which we learn only from "normal" examples, and treat each new, abnormal observation as a potential anomaly. Its basis is the idea of determining the boundary in the space of features in such a way as to best separate normal data from the rest (in a sense from the "emptiness"). Mathematically, given the training set, we want to find such a vector $\{x_i\}_{i=1}^{n}$w and a threshold value R that the values are at least equal to $w \cdot \Phi(xi)$R for as many normal samples as possible (here $\Phi(\cdot)$

Φ(·) means a mapping to the most often nonlinear feature space, which can be multivariate. In the classic formulation, it is usually written as follows:

$$\min_{w,\varepsilon,R} \frac{1}{2}||w||^2 - R + \frac{1}{vn}\sum_{i=1}^{n} \varepsilon_i$$

with conditions:

$$w\, \emptyset(x_i) \geq R - \varepsilon_i, \varepsilon_i \geq 0, i = 1, \ldots, n$$

The expression ξi are the so-called slack variables, which allow for some acceptable "errors" in matching, and ν is an important hyperparameter that controls the proportion of outliers that the model may consider an anomaly. In practice, ν determines the (approximate) percentage of samples that the model can "reject" (i.e. consider outliers), and at the same time affects the flexibility of the boundary that fits the normal data. The larger the value of ν, the more the model is inclined to determine a smaller area of "normality" and thus increases the number of potential anomalies.

To solve a problem in a space of very high dimensionality (or even infinite), the kernel trick is used. Thanks to it, we do not have to explicitly use the Φ projection. Instead, we introduce a kernel function $K(xi, xj) = \langle \Phi(xi), \Phi(xj) \rangle$

[3] B. Al-Duwairi, A. Razaque, *A Survey of AI-Based Intrusion Detection Systems for Cloud Computing Environments*, "International Journal of Cloud Applications and Computing" (IJCAC), 2019, Vol. 9, No. 3, R.S. Sutton, A.G. Barto, *Reinforcement Learning: An Introduction* (2nd Edition), MIT Press, 2018.

which indirectly expresses the dot product in the feature space.[4] The most commonly adopted kernel is the RBF (Radial Basis Function), defined as

$$K(xi, xj) = exp(-\gamma \parallel xi - xj \parallel 2)$$

where γ is a parameter that determines the "range" of the influence of individual samples on the shape of the boundary. At a higher γ value, the boundary may become more jagged (more closely aligned with local data irregularities), and at too small, it may become too smooth, which may not capture subtle anomalies.

The key is that after training the One-Class SVM, we get the decision function f(x) = w·Φ(x)−R. If f(x) is positive, the point x is considered to be. If the function value is positive, the instance is classified as "normal", whereas if it is negative, it is classified as "anomalous". From a cybersecurity perspective, this model is trained exclusively on those segments of network traffic or log records that can be reliably assumed to represent normal behavior. As a result, any new traffic that deviates from the learned patterns produces a negative decision function value and is flagged as a potential threat.

This approach has several practical advantages. It does not require large labeled attack datasets[5] (which are often rare or difficult to collect), while still being able to effectively identify previously unseen situations that did not occur during the system's normal operation period. In the real world, this means that One-Class SVM can often detect a wide range of previously unknown attacks (zero-day attacks), whose signatures cannot be predefined in advance.

Of course, the model is not without limitations. If ν, γ, or the kernel type is not chosen appropriately, it may either generate too many false alarms or fail to detect subtle anomalies. Nevertheless, One-Class SVM has become one of the most popular anomaly detection tools in network environments, offering a useful combination of conceptual simplicity, reliable performance in many cases, and broad support in machine learning libraries such as Scikit-learn.

3. Example Illustrating the Basic Calculations in One-Class SVM on Two Points in a 2-Dimensional Space

This is only an overview diagram that shows what the calculations might look like and how to confirm them in Python. In practice, One-Class SVM usually operates on a larger number of samples and in higher dimensionality and

[4] N. Kshetri, *Artificial Intelligence in Cyber Security*, "IT Professional" 2018, Vol. 20, No. 3.

[5] Yi Chen, Wang Ding, *Isolation Forest for Anomaly Detection*, Mathematical Problems in Engineering, 2018; S. Shalev-Shwartz, S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, 2019.

additionally introduces the so-called slack variables (ξ) and more extensive optimization conditions. Nevertheless, this example will allow you to see "in a nutshell" where some values come from.

Assumptions and data

Let us assume that we have 2 points considered "normal"

$$x1 = (0,0), x2 = (1,1)$$

We choose the RBF (Radial Basis Function) kernel function with γ = 1, ν = 0.5.

In One-Class SVM (in the version with the dual optimization form), we aim to determine, for m.in example, the αi coefficients and the threshold value ρ

In simplified conditions (ignoring slack ξ) we have, among others,

$$\sum_{i=1}^{n} \alpha_i = \frac{1}{vn}$$

n=2 (number of points), and ν=0.5, so

$$\sum_{i=1}^{n} \alpha_i = \frac{1}{0.5 * 2} = 1$$

To solve the problem of minimization (or maximization in dual form), we also need to use the kernel matrix

Kernel matrix calculations

For γ=1

1 $K(x1, x1) = exp(-1 \times \| (0,0) - (0,0) \| 2) = exp(0) = 1$

2 $K(x1, x2) = \exp(-1 \times \| (0,0) - (1,1) \| 2) = \exp(-1 \times 2) = \exp(-2)$

3 $K(x2, x1) = \exp(-1 \times \| (1,1) - (0,0) \| 2) = \exp(-2)$

4 $K(x2, x2) = \exp(-1 \times \| (1,1) - (1,1) \| 2) = \exp(0) = 1$

Thus, the kernel matrix K (2×2) looks like this

$$K = \begin{bmatrix} 1 & e^{-2} \\ e^{-2} & 1 \end{bmatrix}$$

4. Estimating

In the full formulation of the One-Class SVM, we are looking for a solution (in the dual version) to the problem

$$max\alpha - \frac{1}{2}\sum_{i=1}^{n} \quad \sum_{j=1}^{n} \alpha_i \alpha_j \, K(x_i, x_j)$$

Under the conditions of

$$0 \leq \alpha \leq \frac{1}{nv}, \sum_{i=1}^{n} \alpha_i = \frac{1}{vn}$$

For the mini-example under consideration and ν=0.5, it is common (but not always) to obtain a solution in which α1=α2 When α1+α2=1α, the natural candidate is α1=0.5 and α2 =0.5. Then, based on αi, we determine ρ. After omitting the slacks and simplifying the conditions, typically ρ is equal to αi×K(xi,xi) for those xithat lie on the boundary. In practice, however, it may turn out that ρ requires minimal corrections, especially if the boundary conditions αiare violated. However, in such a small example, where the data is perfectly "clean", a solution like this is possible.

After finding αiand ρ, the decision function takes the form:

$$f(x) = \sum_{i=1}^{2} \alpha_i K(x_i, x) - \rho$$

If f(x) ≥ 0, the point x x is considered "normal" (the model says: +1), and if f(x)< 0 is considered an anomaly (-1).

5. Example Implementation

Below is a Python code that confirms that in practice (for such a small set) One-Class SVM will train quickly and give results indicating that both points are "normal". The script will also display the value of dual_coef_ (corresponds to α coefficients) and intercept_ (corresponds to – ρin the decision function) calculated by the library.

Code

```
import numpy as np
from sklearn.svm import OneClassSVM

# Nasze dwie próbki
X = np.array([
    [0.0, 0.0],
    [1.0, 1.0]
])

# Inicjalizujemy One-Class SVM
```

```
# Ustawiamy gamma=1 i nu=0.5 (tak jak w założeniach)
model = OneClassSVM(kernel='rbf', gamma=1.0, nu=0.5)

# Uczymy model na podstawie wyłącznie tych 2 punktów
model.fit(X)

# Podglądamy kluczowe atrybuty
print("Alfa (dual_coef_):", model.dual_coef_)
print("Intercept_ (odpowiada -rho):", model.intercept_)

# Sprawdzamy predykcję dla obu punktów
predictions = model.predict(X)
print("Predykcje:", predictions)
```

Code Result

```
Alfa (dual_coef_): [[0.5 0.5]]
Intercept_ (odpowiada -rho): [-0.56766764]
Predykcje: [1 1]
```

Explanation of the most important elements

model.dual_coef_ – a matrix containing αi from a dual solution. In scikit-learn, it is written as a 2D array, usually [1 x n_samples].

model.intercept_ is a free term in the decision-making function, corresponding to −ρ.

model.predict(X) – for each point in X, it returns +1 (normal) or -1 (anomalous).

With such a small example (only two points), one would expect both to be considered normal, since science proceeds solely on these two observations. As a rule, dual_coef_ will indicate that α1 +α2α1 = 1/(ν·n)

The exact values may differ slightly from the intuitive 0.5 / 0.5, due to implementation details and possible regularization terms in the solver.

The diagram above shows how the main ideas of One-Class SVM work on a microscale

The kernel matrix determines the similarity between points and allows for a non-linear demarcation of the area of normality.

The α coefficients affect how much each point “pushes” the border in its vicinity.

The ν parameter controls how much of the samples can be considered outliers, and

γ in RBF determines how local (or global) the match will be.

In real-world applications (with hundreds or thousands of points), solving equations is too complex for manual calculation, but it is based on the same principles. The implementation in scikit-learn allows you to quickly check what values αand ρ have been determined by the solver, as well as verify which samples will be considered anomalous.

Summary

The presented considerations lead to the conclusion that One-Class SVM algorithms, complemented by additional methods such as Isolation Forest or autoencoders, form the foundation of a modern approach to threat detection in computer networks. They are particularly effective in situations where traditional signature-based systems do not provide suitable attack patterns or when cybercriminals employ unique, previously unknown techniques.

However, the preparation of high-quality data is crucial, ensuring that it is both representative of normal traffic and, if possible, includes various examples of suspicious behavior to verify and calibrate the model (although One-Class SVM itself learns from normal samples).

Deploying such solutions in a production environment requires careful integration with SIEM systems, continuous performance monitoring (especially in terms of false alarm rates), and periodic parameter tuning, which helps adapt to changing network traffic patterns.

The article concludes by emphasizing that, in an era of rapidly evolving threats and highly dynamic IT systems, combining machine learning with good network security practices can significantly enhance protection levels and enable a more proactive defense against attacks.

Bibliography

Al-Duwairi B., Razaque A., *A Survey of AI-Based Intrusion Detection Systems for Cloud Computing Environments*, "International Journal of Cloud Applications and Computing" (IJCAC) 2019, Vol. 9, No. 3.

Chen Yi, Ding Wang, *Isolation Forest for Anomaly Detection*, Mathematical Problems in Engineering, 2018.

Goodfellow I., McDaniel P., Papernot N., *Making Machine Learning Robust Against Adversarial Inputs*, "Communications of the ACM" 2018, Vol. 61, No. 7.

Hu Jing, et al., *A Deep Learning Approach for Intrusion Detection Using Recurrent Neural Networks*, IEEE Access, Vol. 6, 2018.

Kshetri N., *Artificial Intelligence in Cyber Security*, "IT Professional" 2018, Vol. 20, No. 3.

López-Martín M., Carro B., Sánchez-Esguevillas A., Lloret J., *Network Traffic Classifier With Convolutional and Recurrent Neural Networks for Internet of Things*, IEEE Access, Vol. 5, 2017 (updated online in 2018).

Ngo Quang-Hung, et al., *Unsupervised Deep Learning for Real-Time Cyber Anomaly Detection*, IEEE Access, Vol. 7, 2019.

Sarker I., et al., *IntrudTree: A Machine Learning Based Cyber Security Intrusion Detection Model*, "Symmetry" 2021, Vol. 13, No. 4.

Shalev-Shwartz S., Ben-David S., *Understanding Machine Learning: From Theory to Algorithms*, Cambridge University Press, 2019.

Sutton R.S., Barto A.G., *Reinforcement Learning: An Introduction* (2nd Edition), MIT Press, 2018.