AD-DMKDE: ANOMALY DETECTION THROUGH DENSITY MATRICES AND FOURIER FEATURES *

Oscar A. Bustos-Brinez ☑, Joseph A. Gallego-Mejia ☑ Fabio A. González ☑
MindLab
Universidad Nacional de Colombia
Bogotá, Colombia
{oabustosb, jagallegom, fagonzalezo}@unal.edu.co

ABSTRACT
This paper presents a novel density estimation method for anomaly detection using density matrices (a powerful mathematical formalism from quantum mechanics) and Fourier features. The method can be seen as an efficient approximation of Kernel Density Estimation (KDE). A systematic comparison of the proposed method with eleven state-of-the-art anomaly detection methods on various data sets is presented, showing competitive performance on different benchmark data sets. The method is trained efficiently and it uses optimization to find the parameters of data embedding. The prediction phase complexity of the proposed algorithm is constant relative to the training data size, and it performs well in data sets with different anomaly rates. Its architecture allows vectorization and can be implemented on GPU/TPU hardware.

Keywords density matrix · random features · anomaly detection · quantum machine learning.

1 Introduction
An anomaly can be broadly defined as an observation or datum that deviates significantly from the patterns of the data set from which it originates, in one or more features. In most cases, data are generated by complex processes and allow for different types of measurements, so an anomaly may contain valuable information about anomalous behaviors of the generative processes, or elements that are impacting the generation or measurement stages [1]. Then, recognizing this type of data (which can be referred to as unusual, atypical, unexpected, malicious or rare, depending on the scenario), discerning real, meaningful anomalies from normal noisy samples (known as ‘outliers’) and identifying the unusual processes that originate them are the main objectives of Anomaly Detection (AD) [2]. Methods and algorithms that perform AD are key in various applications such as bank fraud detection, identification of abrupt changes in sensors, medical diagnostics, natural sciences, network security intrusion detection, among many others [21].

However, classic AD methods face significant challenges that limit their performance and range of application [14]. In most cases, anomalies are much more sparse and scarce than normal data, and are not identified a priori as anomalies, which makes the use of supervised classifiers difficult. Furthermore, the boundaries between “normal” and “anomaly” regions are very subjective and depend on the particularities of each case. Data may also be contaminated by noise, so that both types of data may be intermingled, blurring the very idea of separation. Many of the AD algorithms address some of these difficulties, but at the cost of being vulnerable to others; however, the combination of classical methods with a good mathematical foundation and deep learning algorithms is one of the most promising paths [11], although it also presents shortcomings on the detection of complex types of anomalies and fewer possibilities of explanation.

The core idea of the proposed method is to use random Fourier features to approximate a Gaussian kernel centered in each training sample and then use a density matrix to summarize these kernels in an efficient and compact way. The density matrix is then used to estimate the density of new samples, and those whose density lies below a certain threshold are classified as anomalies. The method uses optimization to obtain the best random Fourier feature embedding (a

* Citation: Joseph et al., AD-DMKDE: Anomaly Detection through Density Matrices and Fourier Features.
AD-DMKDE: Anomaly Detection through Density Matrices and Fourier Features

process called “Adaptive Fourier Features”), and is able to calculate the best threshold using percentiles from a validation data set.

The outline of the paper is as follows: in Section 2, we describe anomaly detection and density estimation in more depth, and present the benchmark methods to which we will compare our algorithm. In Section 3, we present the novel method, explaining the stages of the algorithm and how it uses Fourier features and density matrices. In Section 4, we systematically compare the proposed algorithm with state-of-the-art anomaly detection algorithms. In Section 5, we state the conclusions of this work and sketch future research directions.

2 Background and Related Work

2.1 Anomaly Detection

The main mechanism of anomaly detection algorithms is the construction of a model that determines a degree of “normality” for the data points, and then detects anomalies as points that deviate from this model. The importance of anomaly detection lies in the fact that recognizing and understanding anomalous behavior allows one to make decisions and predictions. For example, anomalous traffic patterns on a network could mean that sensitive data is being transmitted across it, or that there is an intruder attempting to attack a system [3]. Anomalous user behavior in payment transactions could signify credit card fraud and give some clues as to how to avoid it in the future [23]. Anomaly detection should be distinguished from outlier detection, the purpose of which is data cleaning: to locate outliers (mainly normal data affected by noise) and subsequently remove them from the data set.

Let \( r \) be the ratio of anomaly samples with respect to all samples. When \( r \) is high, the most common approach to anomaly detection is to use a supervised classification method. However, when \( r \) is low (which is usually the case), the best approach is to use semi-supervised or unsupervised AD algorithms. According to [16], classical AD algorithms can be classified into six main types: classification-based approaches (such as the well-known one-class SVM), probabilistic models (such as histograms and density estimation methods like KDE [8]), clustering models, information-based methods, spectral analysis (where dimensionality reduction methods such as PCA are found), and distance-based methods (including isolation forest and nearest neighbors).

In recent years, these classical models have been progressively replaced by deep learning-based algorithms, showing high performance in anomaly detection tasks [25]. These algorithms can efficiently model the inherent features of complex data through their distributed and hierarchical architectures, i.e., they can perform implicit feature engineering, especially when a large amount of data is available [20]. For this reason, they are commonly used for anomaly detection in scenarios involving very large data sets with high-dimensional instances, such as speech recognition, image/video analysis, or natural language processing [13].

2.2 Anomaly Detection Baseline Methods

In this paper, we select 11 state-of-the-art anomaly detection methods that represent a complete sample of the most common types of anomaly detection methods. All algorithms consider the proportion of anomalies in the data as a necessary parameter for finding threshold values; other parameters for specific algorithms are mentioned in later sections.

We selected five well-known methods that are based on classic mathematical approaches to anomaly detection. These methods include One Class SVM [24], a kernel-based algorithm that encloses normal data in a boundary that leaves anomalies outside of it; Covariance Estimator, that finds the smallest ellipsoid that wraps normal data; Isolation Forest [12], that tries to separate points using decision trees, and those who are easier to isolate are the outliers; Local Outlier Factor (LOF) [4], based on a distance measure from each point to a set of its neighbors; and K-nearest neighbor (KNN) [18], that makes an scoring based on the distance from the \( k \)-th nearest neighbor.

We also decided to include methods developed in the last decade that do not use neural networks into their architectures, but instead take other approaches. These type of methods include SOS [7], that builds an affinity matrix between all points that acts as a similarity measure; COPOD [10], that builds univariant probability functions for each dimension and then joins them in a unified multivariant function that models the true distribution of the data; and LODA [15], that combines classifiers in low dimensions and uses histograms to detect anomalies.

Finally, we complete the overview of baseline methods with some of the proposals that are based on the use of neural networks as their central elements. These three models include VAE-Bayes [9], built around a variational autoencoder that builds a latent space from data, where the probability distribution of the data can be retrieved by using Bayesian assumptions; DSVDD [22], in where a neural network is used to transform data into a latent space where normal data can be encompassed into a hypersphere whose features are neurally optimized; and LAKE [13], that includes a
3 Anomaly Detection through Density Matrices and Fourier Features (AD-DMKDE)

González et al. [5] proposed a new algorithm based on the joining of density matrices and random Fourier features for neural density estimation, called “Density matrix for Kernel Density Estimation” (DMKDE), that has its core in kernel density estimation. The DMKDE algorithm consists of a random Fourier feature mapping, a density matrix calculation, and a quantum density estimation phases. This method, whose original purpose was only density estimation, is used as the foundation for the novel method presented here.

AD-DMKDE enhances the random Fourier feature mapping from DMKDE through the use of optimization, that allows to find better values for the parameters of the mapping function. This process, called Adaptive Fourier features, can be seen in more detail in subsection 3.1. After that, the density matrix is built from the mappings of the training samples (see subsection 3.2), and the density of new samples is calculated through a process known as Quantum Density Estimation, inspired in quantum measurement theory from quantum mechanics (see subsection 3.3). Then, as final step, a threshold is obtained through a cross-validation process over a validation data set. This threshold is calculated using the rate of the anomalies in the validation set and is explained in the Subsection 3.4. The test phase of the algorithm comprises applying the already-built Fourier feature mapping over the test set, Quantum Density Estimation and threshold comparison to make the decision whether each test point is anomalous or not. Figure 1 shows a summary of the method.

3.1 Random Fourier features and Adaptive Fourier features

In the training phase, each data point in the training data set is implicitly transformed into a higher dimension. The Random Fourier feature mapping, as first stated in [17], applies the following function $rff(x_i) = \sqrt{2} \cos(w^T x_i + b)$ where $w$ is a vector of iid from a $\mathcal{N}(0, I_D)$ and $b$ is a iid sample from a Uniform(0, 2$\pi$). According to Bochner’s theorem, the kernel $k$ can be approximated as an expectation, $k(x, y) \simeq E_{w}[\langle \phi_{rff}(y), \phi_{rff}(x) \rangle]$. So, since we expect to approximate a Gaussian kernel to perform KDE, the parameters of the kernel, in this case the $\sigma$ (standard deviation) parameter, is one of the major influences over the selection of $w$ and $b$.

This mapping is improved through AFF (Adaptive Fourier features), a mechanism that, before of the RFF mapping, performs an optimization of the values of $w$ and $b$ by building a neural network that adjust their values through a
learning process. After the application of the RFF formula with the optimized values of $w$ and $b$, the obtained results are normalized to obtain the final AFF embedding $\phi_{aff}(x_i)$ that will be the representation of each $x_i$.

### 3.2 Density matrix calculation

Once the AFF embedding has been applied to all training samples, the transformed training data correspond to the set \{ $\phi_{aff}(x_1), \phi_{aff}(x_2), \ldots, \phi_{aff}(x_n) \}$, being $n$ the number of train samples. For each sample, the algorithm computes its corresponding pure-state density matrix $R_i$, given by the outer product of each $\phi_{aff}(x_i)$:

$$R_i = [\phi_{aff}(x_i)]^T \phi_{aff}(x_i)$$

The mixed-state density matrix that comprehends all the training samples is calculated as $R = 1/n \sum_{i=1}^{n} R_i$. Although $R$ contains all the information of the embedded train samples, its size does not depend on $n$; instead, it only depends on the size of the embedding. This is a major improvement over other methods, making it possible to compactly represent (and store) very large data sets using only a single matrix whose size is much smaller in general than the training set. Other advantage of the use of the density matrix representation is the fact that the calculation of $R$ only has to occur once in the entire training phase.

### 3.3 Quantum Density Estimation

With the calculation of $R$, we can estimate the density for new samples, in particular those of the validation and test sets. Given a validation sample $x_j$, we need to transform it by using the exact same mapping and normalization we previously defined in order to obtain $\phi_{aff}(x_j)$, and then calculate

$$\hat{f}(x_j) = [\phi_{aff}(x_j)]^T R [\phi_{aff}(x_j)]$$

as an estimate of the density of $x_j$. This process is repeated for every validation sample, in order to build the validation density set $\hat{F}_{val} = \{ \hat{f}(x_j) \mid x_j \in X_{val} \}$.

### 3.4 Threshold calculation and prediction phase

To determine which of the samples are anomalies or not, we need to find a threshold $\theta$ that can be used as a sort of boundary between these two types of data points. Therefore, we use the anomaly rate of the data set (either an a priori known value or the proportion of anomalies we expect to find). We take the estimates from the validation set and find the percentile that corresponds to the anomaly rate we defined above to calculate the threshold $\theta := q_{(\text{anomaly rate \%})}(\hat{F}_{val})$.

As a final step, we estimate the density of the test samples (by mapping them and applying density estimation with $R$), and then use the threshold $\theta$ to discriminate them. So, the classification of a test sample $x_k$ is given by:

$$\hat{y}(x_k) = \begin{cases} \text{‘normal’} & \text{if } \hat{f}(x_k) \geq \theta \\ \text{‘anomaly’} & \text{otherwise} \end{cases}$$

### 4 Experimental Evaluation

#### 4.1 Experimental Setup

For our experiments, we compared our proposed method with all the baseline algorithms listed above (see section 2.2). To run the first four methods, we used the Python implementation provided by the scikit-learn library. KNN, the shallow methods, VAE and DVSDD were executed through the implementation provided by the PyOD Python library \cite{pyod}. For the LAKE algorithm \cite{lake}, the implementation we used comes from the Github repository of its authors, although we had to correct some issues in the original code, particularly the way to split the test data set to include both normal and anomalous samples.

To handle the inherent randomness that AD-DMKDE can present in some stages (particularly in AFF neural network training), we decided to select a unique, invariant value for every random seed that affect the behavior of the method.
Table 1: Main features of the data sets.

| Data Set     | Instances | Dimensions | Outlier Rate |
|--------------|-----------|------------|--------------|
| Arrhythmia   | 452       | 274        | 0.146        |
| Cardio       | 2060      | 22         | 0.2          |
| SpamBase     | 3485      | 58         | 0.2          |
| Thyroid      | 3772      | 36         | 0.0247       |
| KDDCUP       | 10000     | 118        | 0.1934       |
| Glass        | 214       | 9          | 0.042        |
| Lympho       | 148       | 18         | 0.04         |
| Ionosphere   | 351       | 33         | 0.359        |
| Letter       | 1600      | 32         | 0.0625       |
| MNIST        | 7603      | 100        | 0.092        |
| Musk         | 3062      | 166        | 0.0317       |
| OptDigits    | 5216      | 64         | 0.0288       |
| PenDigits    | 6870      | 16         | 0.0227       |
| Pima         | 768       | 8          | 0.349        |
| Satellite    | 6435      | 36         | 0.3164       |
| SatImage     | 5803      | 36         | 0.0122       |
| Shuttle      | 10000     | 9          | 0.0715       |
| Vertebral    | 240       | 6          | 0.125        |
| Vowels       | 1456      | 12         | 0.03434      |
| WBC          | 378       | 30         | 0.0556       |

All the experiments were carried out on a machine with a 2.1GHz Intel Xeon 64-Core processor with 128GB RAM and two NVIDIA RTX A5000 graphic processing units, that run Ubuntu 20.04.2 operating system.

4.1.1 Data sets.

We used twenty public data sets to evaluate the performance of our proposed algorithm in performing anomaly detection. We chose data sets with a wide variety of characteristics, such as their size or anomaly rate, to test the robustness of our algorithm in multiple scenarios. The main characteristics of all the data sets can be seen in Table 1. The source for all the data sets was the ODDS Library of Stony Brook University [19].

4.1.2 Experiment Configuration.

For each data set, the following configuration was chosen: the data set was split in a stratified way (keeping the same proportion of outliers in each subset) by randomly taking 30% of the samples as the test set, and from the remaining samples, again randomly taking 30% for validation and the remaining 70% as the training set. This splitting was performed only once per data set, so that all algorithms worked with exactly the same data partitions.

When implementing the Scikit-Learn and PyOD algorithms, the default configurations defined in these libraries were used, modifying only the parameters referred to the data set features mentioned in the previous section. The only exception to this rule was the VAE algorithm, whose architecture, based on autoencoders, needed to be slightly modified when using low-dimensional data sets, to ensure that the encoded representation was always of lower dimensionality than the original sample. Also, as mentioned above, LAKE algorithm required some corrections to the original code published by its authors; however, the basic internal structure, the internal logic of the algorithm and the functions on which it is based remained the same as in its original configuration.

Our AD-DMKDE algorithm was implemented from the original code of DMKDE, presented in [6], with some modifications. Since the main goal of DMKDE is density estimation, the separation of data as normal or outlier was performed by adding a function that calculates the classification threshold using the rate of anomalies in each data set, a known value in all cases (see table 1).

4.1.3 Evaluation Metrics.

The main metric we chose to determine the performance of the algorithms was the F1 score (with weighted average), a well-known and very common metric used when testing classification algorithms. However, the calculation of other common metrics, such as accuracy and the exclusive F1-Score to the anomalous class, was also implemented. For each algorithm, a set of parameters of interest was selected in order to perform a comprehensive search for the combinations of values that gave the best result for each data set. This search was performed by training only with the training set and reporting the results of the validation set. The selection of the best combination of parameters was performed by choosing the model with the best F1 score. Once the best parameters were found, these same values were used to run each algorithm once again, this time combining the training and validation sets for the training phase and reporting the results on the test set. These final results were the values we used to compare the overall performance of all algorithms.
When we consider the influence of the characteristics of the data set on the performance of our method, we see an
interesting pattern. The outlier rate of the data sets does not seem to have any influence over the performance of
AD-DMKDE, since the method performs well when there are a low number of anomalies (like in Musk or Shuttle), and
also when there are many of them (like in Cardio or Pima). The number of dimensions have a weak influence, since the
method has the best performance more frequently in data sets with less than twenty dimensions (such as in Vertebral,
Pima or Pendigits), but also in half of the data sets with more than 100 dimensions (Arrhythmia and Musk). Finally,
when we look at the number of samples, we see that AD-DMKDE stands out more for data sets with few instances (less
than 1000) , and also for many instances (more than 5000), but its performance is not as good in the intermediate data
sets (the ones that have between 1000 and 5000 instances). In summary, AD-DMKDE is a method that can perform
over the average state-of-the-art anomaly detection methods, standing out in many different scenarios and being capable
of affording data sets with both low and high values in its number of dimensions, number of samples and outlier rates.

5 Conclusion

In this paper, we presented a novel approach for anomaly detection using density matrices from quantum mechanics and
random Fourier features. The new method AD-DMKDE was systematically compared against eleven different anomaly
detection algorithms using F1 Score as main metric. AD-DMKDE show better than state-of-the-art performance over twenty anomaly detection data sets, being notably superior than classic algorithms and comparable to deep
learning-based methods. The performance of the method does not seem to be affected by the anomaly rate of the data
sets or the size of the data sets, but it seems to perform better for data sets with low dimensionality of the samples.
AD-DMKDE do not have huge memory requirements due to that the method constructs a single density matrix in all
training phase whose size is defined by the embedding. So it is possible to resume very large data sets in relatively small
matrices, which makes AD-DMKDE a scalable solution in those cases. Also, the method allows easy interpretability of
the results, because each data point labeled as “anomaly” can be understood as a sample lying in low density regions
with respect to normal data. As future work, we will continue to further develop the main concepts of AD-DMKDE,
buidling new algorithms based on the combination of Fourier features and density matrices with deeper neural networks,
such as autoencoders and variational autoencoders, whose reduction power can be coupled to the new method in order
to better handle data sets with higher dimensionalities.
References

[1] Aggarwal, C.C.: Outlier analysis second edition (2016)
[2] Blámez-García, A., Conde, A., Mori, U., Lozano, J.A.: A review on outlier/anomaly detection in time series data. ACM Computing Surveys (CSUR) 54(3), 1–33 (2021)
[3] Bouyeddou, B., Harrou, F., Kadri, B., Sun, Y.: Detecting network cyber-attacks using an integrated statistical approach. Cluster Computing 24, 1435–1453 (2021)
[4] Breunig, M.M., Kriegel, H.P., Ng, R.T., Sander, J.: Lof: identifying density-based local outliers. In: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. pp. 93–104 (2000)
[5] Gallego, J.A., González, F.A.: Quantum adaptive fourier features for neural density estimation (2022). https://doi.org/10.48550/ARXIV.2208.00564
[6] González, F.A., Gallego, A., Toledo-Cortés, S., Vargas-Calderón, V.: Learning with density matrices and random features. Quantum Machine Intelligence 4(2) (2022)
[7] Janssens, J., Huszar, F., Postma, E., van den Herik, H.: Stochastic outlier selection (2012)
[8] Kalair, K., Connaughton, C.: Anomaly detection and classification in traffic flow data from fluctuations in the flow–density relationship. Transportation Research Part C: Emerging Technologies 127 (2021), cited By 0
[9] Kingma, D.P., Welling, M.: Auto-encoding variational bayes (2014)
[10] Li, Z., Zhao, Y., Botta, N., Ionescu, C., Hu, X.: Copod: Copula-based outlier detection. pp. 1118–1123 (11 2020)
[11] Liu, F., Huang, X., Chen, Y., Suykens, J.A.K.: Random Features for Kernel Approximation: A Survey on Algorithms, Theory, and Beyond (apr 2020)
[12] Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 eighth ieee international conference on data mining. pp. 413–422. IEEE (2008)
[13] Lv, P., Yu, Y., Fan, Y., Tang, X., Tong, X.: Layer-constrained variational autoencoding kernel density estimation model for anomaly detection. Knowledge-Based Systems 196 (2020)
[14] Pang, G., Shen, C., Cao, L., Hengel, A.V.D.: Deep learning for anomaly detection: A review. ACM Computing Surveys (CSUR) 54(2), 1–38 (2021)
[15] Pevný, T.: Loda: Lightweight on-line detector of anomalies. Machine Learning 102, 275–304 (2016)
[16] Prasad, N.R., Almanza-Garcia, S., Lu, T.T.: Anomaly detection. Computers, Materials and Continua 14, 1–22 (2009)
[17] Rahimi, A., Recht, B.: Random features for large-scale kernel machines. In: Proceedings of the 20th International Conference on Neural Information Processing Systems. p. 1177–1184. NIPS’07, Curran Associates Inc. (2007)
[18] Ramaswamy, S., Rastogi, R., Shim, K.: Efficient algorithms for mining outliers from large data sets. p. 427438. Association for Computing Machinery (2000)
[19] Rayana, S.: ODDS library (2016), http://odds.cs.stonybrook.edu
[20] Rippel, O., Mertens, P., Konig, E., Merhof, D.: Gaussian anomaly detection by modeling the distribution of normal data in pretrained deep features. IEEE Transactions on Instrumentation and Measurement (2021)
[21] Ruff, L., Kauffmann, J.R., Vandermeulen, R.A., Montavon, G., Samek, W., Kloft, M., Dietterich, T.G., Muller, K.R.: A unifying review of deep and shallow anomaly detection. Proceedings of the IEEE 109, 756–795 (2021)
[22] Ruff, L., Vandermeulen, R., Goernitz, N., Deecke, L., Siddiqui, S.A., Binder, A., Müller, E., Kloft, M.: Deep one-class classification. vol. 80. PMLR (2018)
[23] Santosh, T., Ramesh, D.: Machine learning approach on apache spark for credit card fraud detection. Ingenierie des Systemes d’Information 25, 101–106 (2020)
[24] Schölkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural computation (2001)
[25] Zhang, C., Liu, J., Chen, W., Shi, J., Yao, M., Yan, X., Xu, N., Chen, D.: Unsupervised anomaly detection based on deep autoencoding and clustering. Security and Communication Networks 2021 (2021)
[26] Zhao, Y., Nasrullah, Z., Li, Z.: Pyod: A python toolbox for scalable outlier detection. Journal of Machine Learning Research 20 (2019)