



Anomaly Detection through Density Matrices and Kernel Density Estimation (AD-DMKDE)

Oscar A. Bustos-Brinez 
National University of Colombia
Bogotá, Colombia
oabustosb@unal.edu.co

Joseph A. Gallego-Mejia 
National University of Colombia
Bogotá, Colombia
jagallegom@unal.edu.co

Fabio A. González 
National University of Colombia
Bogotá, Colombia
fagonzalezo@unal.edu.co

Abstract

This paper presents a novel anomaly detection method, called AD-DMKDE, based on the use of Kernel Density Estimation (KDE) along with density matrices (a powerful mathematical formalism from quantum mechanics) and Fourier features. The proposed method was systematically compared with eleven state-of-the-art anomaly detection methods on various data sets, and AD-DMKDE shows competitive performance. The method uses neural-network optimization to find the parameters of data embedding, and the prediction phase complexity of the proposed algorithm is constant relative to the training data size.

1 Introduction

An anomaly can be defined as an observation that deviates significantly from the patterns of the data set from which it originates. In most cases, data are generated by complex processes that depend on a great number of factors, so anomalies may contain valuable information about unexpected behaviors or elements that impact the generation or measurement of the data Aggarwal [2016]. Thus, recognizing anomalous data (which can be referred to as unusual, atypical, unexpected or malicious) and identifying the unusual processes that originate them are the main objectives of anomaly detection (AD) Blázquez-García et al. [2021], a field that has become quite important in recent years because the understanding of anomalous behavior is a crucial ability when making decisions and predictions Ruff et al. [2021]. The main mechanism used by AD 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. These algorithms are commonly used in applications like detecting anomalous readings in scientific experiments Flach et al. [2017], sensor monitoring in industry Denkena et al. [2020], network and information security Bouyeddou et al. [2021], among many others.

The main idea behind the method presented in this paper, called AD-DMKDE, is the combination of three key elements: random Fourier features Rahimi and Recht [2007] as an embedding that allows to approximate a Gaussian kernel centered in each training sample; a density matrix, that serves as an efficient and compact mechanism to summarize these kernels, and a density estimation process that uses the density matrix to estimate the density of new samples, so the ones whose density lie below a certain threshold are classified as anomalies. The method uses optimization to obtain suitable parameters of random Fourier feature embedding (a process called “Adaptive Fourier Features”), and is able to calculate the best threshold value using percentiles from a validation data set.

The outline of the paper is as follows: in Section 2, we present the baseline anomaly detection methods that were used to compare the proposed algorithm. In Section 3, we present the details of the novel method, explaining the stages of the algorithm and how it uses Fourier features and density matrices. In Section 4, we present the experimental setup we constructed and the results of the comparison between the proposed algorithm and the baseline methods. In Section 5, we state the conclusions of this work and sketch future research directions.

2 Anomaly Detection Baseline Methods

First of all, we selected five well-known methods based on classic, mathematical approaches to anomaly detection. These methods include One Class SVM Schölkopf et al. [2001], a kernel-based algorithm that builds a boundary that encloses normal data and leaves anomalies outside of it; Covariance Estimator Rousseeuw and Driessen [1999], that finds the smallest ellipsoid that wraps normal data; Isolation Forest Liu et al. [2008], that tries to separate points using decision trees, and those who are easier to isolate are the outliers; Local Outlier Factor (LOF) Breunig et al. [2000], based on a distance measure from each point to a set of its neighbors; and K-nearest neighbor (KNN) Ramaswamy et al. [2000], that makes an scoring based on the distance from only the k -th nearest neighbor.

We also included newer methods that do not use neural networks into their architectures, but instead take other approaches. These type of methods include SOS Janssens et al. [2012], that builds an affinity matrix between all points that acts as a similarity measure; COPOD Li et al. [2020], that builds univariate probability functions for each dimension and then joins them in a unified multivariate function that models the data distribution; and LODA Pevný [2016], that combines simple classifiers in low dimensions and uses histograms to detect anomalies.

Furthermore, we consider three additional baselines algorithms that rely on the use of neural networks as their central elements. These models include VAE-Bayes Kingma and Welling [2014], built around a variational autoencoder that maps data to a latent space where the probability distribution can be retrieved by using Bayesian assumptions; DSVDD Ruff et al. [2018], in where a neural network is used to transform data points into a latent space where normal data can be encompassed into a hypersphere; and LAKE Lv et al. [2020], that includes a variational autoencoder to reduce dimensionality in a way that preserves data distribution, and then performs KDE in this new space and separates anomalies using a threshold.

3 Anomaly Detection through Density Matrices and Kernel Density Estimation (AD-DMKDE)

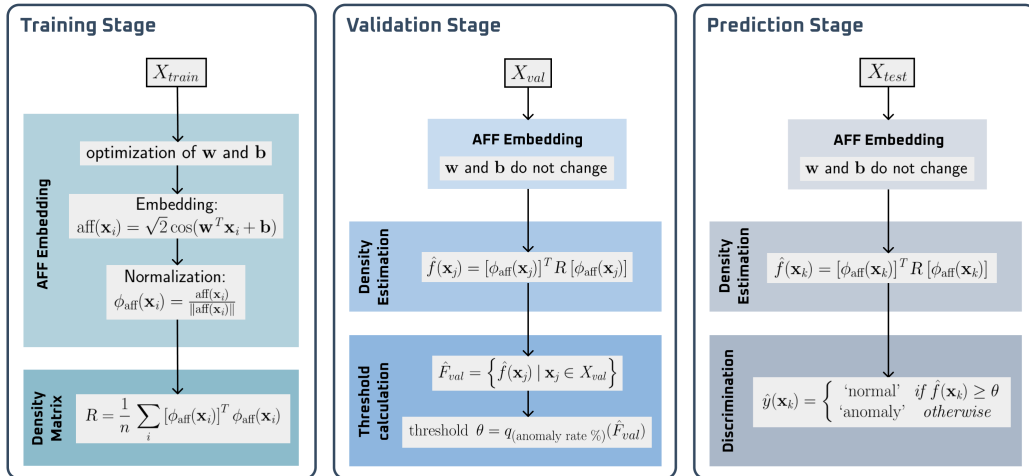


Figure 1: AD-DMKDE method architecture. The parameters of the embedding do not change after they are calculated.

González et al. Gallego and González [2022], González et al. [2022] proposed an algorithm intended for density estimation called “*Density matrix for Kernel Density Estimation*” (DMKDE), based on the usage of density matrices as an efficient way to approximate kernel density estimation. This algorithm is used as the foundation from which AD-DMKDE, the novel method presented here, is built. Figure 1 shows a summary of the method.

AD-DMKDE splits the dataset in three subsets: training set, validation set and test set. When training, each data point in the training data set is implicitly transformed into a higher dimension by applying the function $\text{rff}(\mathbf{x}_i) = \sqrt{2} \cos(\mathbf{w}^T \mathbf{x}_i + \mathbf{b})$ where \mathbf{w} and \mathbf{b} are vectors made of random samples. This mapping is able to approximate a Gaussian kernel, as stated in Rahimi and Recht [2007]. Also, AD-DMKDE enhances it through the use of optimization, that allows to find better values for the parameters of the mapping function. This process is called Adaptive Fourier features (AFF). After the application of the RFF formula with the optimized values of \mathbf{w} and \mathbf{b} , the obtained results are normalized to obtain the final embedding $\phi_{\text{aff}}(\mathbf{x}_i)$ that will be the representation of each \mathbf{x}_i .

The density matrix is built from the mappings of the training samples by using the formula

$$R = \frac{1}{n} \sum_{i=1}^n [\phi_{\text{aff}}(\mathbf{x}_i)]^T \phi_{\text{aff}}(\mathbf{x}_i)$$

Although R contains all the information of the embedded train samples, its size does not depend on the number of training samples n ; instead, it only depends on the size of the embedding. This is a major improvement over memory-based methods, such as KDE, that require the use of the entire training set to make a new prediction and thus can become prohibitive on large data sets.

With R we estimate the density for samples in validation and test sets. Given a validation sample \mathbf{x}_j , we need to transform it by using the exact same mapping and normalization we previously defined in order to obtain $\phi_{\text{aff}}(\mathbf{x}_j)$, and then calculate $\hat{f}(\mathbf{x}_j) = [\phi_{\text{aff}}(\mathbf{x}_j)]^T R [\phi_{\text{aff}}(\mathbf{x}_j)]$ as an estimate of the density of \mathbf{x}_j . This process is repeated in order to build the validation density set \hat{F}_{val} , that contains the estimates of every validation sample.

\hat{F}_{val} is required to obtain a threshold θ to discriminate samples as anomalies. We use the anomaly rate of the data set (either an a priori known value or the proportion of anomalies we expect to find), and calculate the percentile that corresponds to this anomaly rate: $\theta := q_{(\text{anomaly rate } \%)}(\hat{F}_{\text{val}})$. Finally, we estimate the density of the test samples (by mapping them and applying density estimation using R , similarly to the validation samples), and then use the threshold θ to discriminate. A test sample \mathbf{x}_k is labeled as ‘anomaly’ if $\hat{f}(\mathbf{x}_k) \leq \theta$, and as ‘normal’ otherwise.

4 Experimental Evaluation

For our experiments, we compared AD-DMKDE with all the baseline algorithms listed above. To run OneClassSVM, IsolationForest, Covariance and LOF, we used the Python implementation provided by Scikit-Learn library. KNN, the shallow methods, VAE-Bayes and DVSSD were run through the implementation provided by the PyOD Python library Zhao et al. [2019]. For the LAKE algorithm Lv et al. [2020], the implementation we used came from the Github repository of its authors.

The experimental setup applied over each algorithm (and the proposed method) consisted of eighteen public data sets, chosen due of their variety of characteristics, such as their size, dimensionality and anomaly rate, in order to test the performance of the algorithms in multiple scenarios. The source for all the selected data sets was the ODDS Library of Stony Brook University Rayana [2016]. Each 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. The anomalies in training set were ignored, using only normal samples in the AD-DMKDE process. The 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 a few parameters whose final values were decided through a parameter grid search. Also, the 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

Data Set	OCSVM	iForest	Cov.	LOF	KNN	SOS	COPOD	LODA	VAE-B	DSVDD	LAKE	AD-DMKDE
Arrhythmia	0.813	0.821	0.818	0.804	0.861	0.773	0.844	0.798	0.856	0.864	<u>0.909</u>	0.911
Cardio	<u>0.804</u>	0.752	0.756	0.702	0.753	0.739	0.750	0.717	0.783	0.735	0.772	0.831
Glass	0.916	0.931	0.931	0.925	0.900	0.907	0.916	0.848	0.900	0.907	1.000	<u>0.974</u>
Ionosphere	0.765	0.710	0.876	0.830	0.817	0.784	0.736	0.510	0.714	0.674	0.993	<u>0.959</u>
Letter	0.893	0.897	0.909	0.930	0.910	0.911	0.895	0.899	0.897	0.897	0.838	<u>0.927</u>
Lympho	0.934	1.000	0.934	1.000	1.000	0.934	0.962	0.923	1.000	1.000	1.000	1.000
MNIST	0.881	0.881	0.841	0.886	0.909	0.864	0.868	0.866	0.895	0.880	0.959	<u>0.911</u>
Musk	0.958	0.931	<u>0.997</u>	0.958	0.991	0.951	0.964	0.954	0.984	0.992	0.996	1.000
OptDigits	0.952	0.952	0.952	0.949	0.952	0.953	0.949	0.955	0.951	0.952	<u>0.976</u>	0.981
PenDigits	0.963	<u>0.967</u>	0.966	0.961	0.965	0.963	0.966	0.966	0.966	0.963	0.994	0.994
Pima	0.592	0.624	0.559	0.615	0.644	0.636	0.615	0.597	0.632	0.679	<u>0.740</u>	0.758
Satellite	0.681	0.757	0.813	0.634	0.716	0.597	0.732	0.709	0.761	0.761	<u>0.841</u>	0.845
SatImage	0.984	<u>0.998</u>	0.991	0.979	0.998	0.980	0.994	0.997	0.996	0.996	0.946	1.000
SpamBase	0.702	0.794	0.714	0.702	0.719	0.719	0.799	0.699	0.741	0.738	0.850	<u>0.816</u>
Thyroid	0.953	0.958	0.986	0.949	0.953	0.949	0.953	0.958	0.958	0.961	0.803	<u>0.967</u>
Vertebral	0.750	0.778	0.817	0.796	0.810	0.817	0.817	0.817	0.817	<u>0.819</u>	0.807	0.904
Vowels	0.950	0.942	0.941	0.951	0.969	0.954	0.943	0.929	0.952	0.943	1.000	<u>0.979</u>
WBC	0.942	0.941	0.949	0.957	0.942	0.913	0.970	0.947	0.957	0.956	0.953	<u>0.961</u>

Table 1: F1 Score for all classifiers over all data sets. The first and second best values are marked in bold and underlined, respectively.

the functions on which it is based remained the same as in its original configuration. The main metric we chose to determine the performance of the algorithms was the F1 score (with weighted average), a very common metric used when testing machine learning algorithms.

The results for all data sets and all algorithms (baseline methods and our proposed method) are shown in Table 1. The best value for each data set is highlighted in bold, and the second best value is underlined. A notable difference appears between most of the baseline methods and our proposed method, with LAKE being a notable exception; however, AD-DMKDE shows the best or second best performance in all the considered data sets, followed by LAKE and LOF methods.

To determine if the differences between all the methods are statistically significant, we performed the Friedman test, a well-known statistical method to compare different populations or groups. The test generates an specific p-value that acts as a measure of the total differentiation among the groups, so if this value is lower than a given confidence α (in our case, $\alpha = 0.05$), we can assure that there is statistically significant evidence supporting that the methods are different. When applying this test to the results in Table 1, we obtain a p-value of $1,031 \times 10^{-13}$, so it strongly assures that there is a difference between the performance of the methods. After that, we compared the algorithms two by two, using the Friedman-Nemenyi test, a variation of the former that informs us if there is a significant difference for every pair of methods. The result of this second test indicates that AD-DMKDE stands out, being different with respect to all other methods, with the sole exception of LAKE. Besides, LAKE differs only with other five methods, and the other methods do not differ significantly between them. In summary, AD-DMKDE is a method that can perform over the average state-of-the-art anomaly detection methods, standing out when affording data sets with various values in its number of dimensions, number of samples and outlier rates.

5 Conclusion

In this paper, we presented a novel method for anomaly detection using density matrices in combination with Kernel Density Estimation and random Fourier features. The new method AD-DMKDE was systematically compared against eleven different anomaly detection algorithms using F1 Score as main metric, and it showed better than state-of-the-art performance over eighteen anomaly detection data sets, being notably superior than classic algorithms and comparable to deep learning-based methods. AD-DMKDE does not have large memory requirements because the method builds a single density matrix throughout the training phase whose size is defined only by the embedding, allowing the summarizing of large data sets in relatively small matrices, showing an advantage in computational complexity with respect to KDE. In addition, the method allows for easy interpretation 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, building new algorithms based on the coupling of the method with deeper neural networks, such as autoencoders, whose strengths can contribute to improve the

AD-DMKDE method to allow it to handle more complex data sets like image data sets or time series data sets.

References

- C. C. Aggarwal. *Outlier Analysis. Second edition*. Springer Switzerland, 2016.
- A. Blázquez-García, A. Conde, U. Mori, and J. A. Lozano. A review on outlier/anomaly detection in time series data. *ACM Computing Surveys (CSUR)*, 54(3):1–33, 2021.
- B. Bouyeddou, F. Harrou, B. Kadri, and Y. Sun. Detecting network cyber-attacks using an integrated statistical approach. *Cluster Computing*, 24:1435–1453, 2021.
- M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. In *Proceedings of the 2000 ACM SIGMOD international conference on Management of data*, pages 93–104, 2000.
- B. Denkena, M.-A. Dittrich, H. Noske, and M. Witt. Statistical approaches for semi-supervised anomaly detection in machining. *Production Engineering*, 14(3):385–393, 2020.
- M. Flach, F. Gans, A. Brenning, J. Denzler, M. Reichstein, E. Rodner, S. Bathiany, P. Bodesheim, Y. Guaniche, S. Sippel, et al. Multivariate anomaly detection for earth observations: a comparison of algorithms and feature extraction techniques. *Earth System Dynamics*, 8(3):677–696, 2017.
- J. A. Gallego and F. A. González. Quantum adaptive fourier features for neural density estimation, 2022.
- F. A. González, A. Gallego, S. Toledo-Cortés, and V. Vargas-Calderón. Learning with density matrices and random features. *Quantum Machine Intelligence*, 4(2), 2022.
- J. Janssens, F. Huszar, E. Postma, and H. van den Herik. Stochastic outlier selection, 2012.
- D. P. Kingma and M. Welling. Auto-encoding variational bayes, 2014.
- Z. Li, Y. Zhao, N. Botta, C. Ionescu, and X. Hu. Copod: Copula-based outlier detection. pages 1118–1123, 11 2020.
- F. T. Liu, K. M. Ting, and Z.-H. Zhou. Isolation forest. In *2008 eighth ieee international conference on data mining*, pages 413–422. IEEE, 2008.
- P. Lv, Y. Yu, Y. Fan, X. Tang, and X. Tong. Layer-constrained variational autoencoding kernel density estimation model for anomaly detection. *Knowledge-Based Systems*, 196, 2020. ISSN 09507051.
- T. Pevný. Loda: Lightweight on-line detector of anomalies. *Machine Learning*, 102:275–304, 2016. ISSN 1573-0565.
- A. Rahimi and B. Recht. Random features for large-scale kernel machines. In *Proceedings of the 20th International Conference on Neural Information Processing Systems, NIPS’07*, page 1177–1184. Curran Associates Inc., 2007. ISBN 9781605603520.
- S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. page 427438. Association for Computing Machinery, 2000. ISBN 1581132174.
- S. Rayana. ODDS library, 2016. URL <http://odds.cs.stonybrook.edu>.
- P. J. Rousseeuw and K. V. Driessen. A fast algorithm for the minimum covariance determinant estimator. *Technometrics*, 41(3):212–223, 1999.
- L. Ruff, R. Vandermeulen, N. Goernitz, L. Deecke, S. A. Siddiqui, A. Binder, E. Müller, and M. Kloft. Deep one-class classification. volume 80. PMLR, 2018.
- L. Ruff, J. R. Kauffmann, R. A. Vandermeulen, G. Montavon, W. Samek, M. Kloft, T. G. Dietterich, and K.-R. Muller. A unifying review of deep and shallow anomaly detection. *Proceedings of the IEEE*, 109:756–795, 2021.
- B. Schölkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural computation*, 2001.
- Y. Zhao, Z. Nasrullah, and Z. Li. Pyod: A python toolbox for scalable outlier detection. *Journal of Machine Learning Research*, 20, 2019. ISSN 15337928.