The Elliptical Basis Function Data Descriptor (EBFDD) Network: A One-Class Classification Approach to Anomaly Detection

Mehran Hossein Zadeh Bazargani(✉) and Brian Mac Namee

The Insight Centre for Data Analytics, School of Computer Science, University College Dublin, Dublin, Ireland
mehran.hosseinzadehbazarga@ucdconnect.ie, brian.macnamee@ucd.ie

Abstract. This paper introduces the Elliptical Basis Function Data Descriptor (EBFDD) network, a one-class classification approach to anomaly detection based on Radial Basis Function (RBF) neural networks. The EBFDD network uses elliptical basis functions, which allows it to learn sophisticated decision boundaries while retaining the advantages of a shallow network. We have proposed a novel cost function, whose minimisation results in a trained anomaly detector that only requires examples of the *normal* class at training time. The paper includes a large benchmark experiment that evaluates the performance of EBFDD network and compares it to state of the art one-class classification algorithms including the One-Class Support Vector Machine and the Isolation Forest. The experiments show that, overall, the EBFDD network outperforms the state of the art approaches.

Keywords: Anomaly detection · Elliptical basis function · Neural networks

1 Introduction

Chandola and Kumar [4] define *anomaly detection* as "*the problem of finding patterns in data that do not conform to expected behavior*". Although anomaly detection is essentially a binary classification problem (i.e. instances are classified as either *normal* or *anomalous*), in anomaly detection scenarios the classes are highly imbalanced—there is very limited, or sometimes no access to anomalous instances during training, although there is usually an abundance

This work was supported by Science Foundation Ireland under Grant No. 15/CDA/3520 and Grant No. 12/RC/2289.

Electronic supplementary material The online version of this chapter (https://doi.org/10.1007/978-3-030-46150-8_7) contains supplementary material, which is available to authorized users.

U. Brefeld et al. (Eds.): ECML PKDD 2019, LNAI 11906, pp. 107–123, 2020.
https://doi.org/10.1007/978-3-030-46150-8_7

of normal instances. Anomaly detection approaches that do not use anomalous instances during training can be classified as *semi-supervised* machine learning approaches, and are often referred to as *one-class classifiers* [12]. Anomaly detection approaches have been used in a variety of applications including credit scoring [11], intrusion detection [2], forensics [13], medical applications [14], and computer network security [27].

The main contribution of this paper is to adapt the Radial Basis Function (RBF) network [3] for one-class classification through a novel cost function. We have named the resultant one-class neural network the Elliptical Basis Function Data Descriptor (EBFDD) network. Coupled with our novel cost function, the EBFDD network is a semi-supervised one-class classification approach that utilizes elliptical kernels to learn sophisticated decision boundaries, while retaining the advantages of a shallow network, which include: easy retraining, interpretability, reduced data requirements, and shorter training time.

The remainder of the paper is structured as follows: Sect. 2 reviews related work, and briefly explains the motivation behind the EBFDD network approach. In Sect. 3 the EBFDD network approach is explained in detail. Section 4 describes the design of an evaluation experiment that compares the performance of the EBFDD network to state of the art algorithms across a number of benchmark datasets. Section 5 presents the results of the experiment, which then discusses their implications. Finally, Sect. 6 concludes the paper and suggests directions for future work.

2 Related Work

In this section we first describe some of the common approaches to anomaly detection. Then we describe the standard RBF network before explaining how it has been adapted for anomaly detection in the EBFDD network.

2.1 Common Approaches to Anomaly Detection

Semi-supervised machine learning approaches to anomaly detection are dominated by a family of algorithms that are modifications of the Support Vector Machine (SVM) algorithm [24], designed to work with only examples of a single class: One-Class SVM (OCSVM) [23]. In fact Khan and Madden [12] go so far as to say that one-class classification algorithms and methods should be divided into OCSVMs and non-OCSVMs.

The idea underpinning OCSVM is quite similar to the standard SVM method, and the *kernel trick* is still a key part of the OCSVM for the transformation of the input space into a feature space of higher dimensionality. The OCSVM approach finds a hyper-plane that separates all of the normal data points in a training set from the origin, while maximizing the distance between the origin and this hyper-plane. This results in a binary function, whose output is 1 for the regions of the input space belonging to the normal data, and -1 anywhere else. The main disadvantage of the OCSVM is the assumption that the anomalous data instances are concentrated around the origin [12]. Variations of the OCSVM

approach include the Support Vector Machine Data Description (SVDD) [26], which uses hyper-spheres rather than hyper-planes to achieve separation. Interestingly, the authors in [22] propose the idea of a deep SVDD, where the SVDD is mixed with deep learning to accomplish anomaly detection.

On the non-OCSVM side, Auto-Encoder networks (AENs) [7], and all their variations have been increasingly used for anomaly detection [29–31]. An AEN does not learn a discriminative model, but rather a generative model of the input data. After transforming the input data into a representation with reduced dimensionality, it learns to reconstruct the original input data from the dimensionally-reduced representation. The error between the input and the reconstruction of the input, referred to as the reconstruction error, can be used as an anomaly detection signal—normal instances should be accurately reproduced leading to low reconstruction error, while anomalous instances should be poorly reproduced leading to large reconstruction errors. The AEN is trained on normal data only and learns features that could best reconstruct the normal data.

Isolation Forest (iForest) [28] is another interesting non-OCSVM approach to anomaly detection. The iForest tries to isolate individual data points in the training set by splitting the space randomly and repeatedly. The intuition behind this approach is that less splits should be required to isolate anomalous instances. An anomaly score can be calculated based on the number of splits required to isolate a data point.

Gaussian Mixture Models (GMMs) [1] are also used for anomaly detection. GMMs assume that normal data is generated by a collection of H Gaussians, which are placed randomly in the input space in the beginning of training. Then using the Expectation Maximization (EM) [1], the means and covariance matrices of the Gaussians are learned such that the likelihood of observing the training data is maximised. During testing, one can measure the likelihood of the test data, and if it is below a certain threshold it could be labeled as anomalous, and normal otherwise.

2.2 Radial Basis Function Networks

A Radial Basis Function (RBF) network [3] is a local-representation learning technique, which divides the input space among local kernels. For every input data point, depending on where in the input space it appears, a fraction of these locally-tuned kernel units get activated. It is as if these local units have divided the input space among themselves and each one takes responsibility for a subspace. The idea of locality, inherently implies the need for a distance function that measures the similarity between a given input data instance X, with dimensionality D, and the center, μ_h, of every kernel unit h. The common choice for this measure is the Euclidean distance, $\|X - \mu_h\|$. The response function for these local units, should have a maximum when $X = \mu_h$, and decrease as X and μ_h get less similar. The most commonly used response function for the RBF units is the Gaussian function. Not only have the RBF networks been used for traditional classification [18] and regression [25] problems, they have also been applied to anomaly detection tasks, which we will explore below.

RBF networks have been applied to anomaly detection in two main ways. First, if examples of both normal and anomalous data are available during training, the standard binary/multi-class classification RBF networks can be used with modifications. For example, in [19], a hybrid optimisation algorithm based on RBF networks, which combines gradient descent with quantum-behaved particle swarm optimisation is used to train the RBF network for anomaly detection. Similarly, in [21], an RBF network is trained for the task of intrusion detection. The model keeps adding hidden units to the architecture until a certain performance goal is met.

In the second approach to using RBF networks for anomaly detection, only examples of the normal class are used at training time. This can be achieved by modifying the dataset or modifying the algorithm. As an example of the first type, in [20] an augmented training set is composed using the instances belonging to the normal class as random proxy anomalous patterns and used to train an anomaly detection model for time series data. As an example of the second type, in [17], the RBF network algorithm is combined with the Support Vector Data Descriptor (SVDD) [26] algorithm, resulting in a hybrid model. The hidden layer of the RBF network is used as a feature extractor and the outputs for the subsequent transformed feature space are then used as inputs to the SVDD algorithm with a linear kernel.

The EBFDD network approach modifies the cost function used to train an RBF network to ensure that it learns a compact set of Gaussian kernels that concentrate around the normal region of the input space, covering all of it, while excluding other regions. During testing, for a given input data, X, the output of the model would be high, if X belongs to the normal region, and low, otherwise.

3 The Elliptical Basis Function Data Descriptor (EBFDD) Network

The EBFDD network is a semi-supervised one-class classifier, which is based on the Radial Basis Function (RBF) network [3]. Figure 1 illustrates the architecture of the EBFDD network. This is a shallow network containing one hidden layer and one output layer with a single node. There are H hidden nodes in the hidden layer, each of which uses a Gaussian activation function. The activation, $p_h(X)$, of the h^{th} Gaussian node is defined as:

$$p_h(X) = exp\left[-\frac{1}{2}\left(X-\mu_h\right)^T {\Sigma_h}^{-1}\left(X-\mu_h\right)\right] \tag{1}$$

where X is the input vector of length D; x_d is the value of the d^{th} dimension of X; and the parameters μ_h and Σ_h are the mean vector (which is D-dimensional) and the covariance matrix (which is a $D \times D$ matrix) of the h^{th} Gaussian kernel.

The outputs of the nodes in the hidden layer are connected to a single output node via a weight vector W, where w_h is the weight parameter connecting the

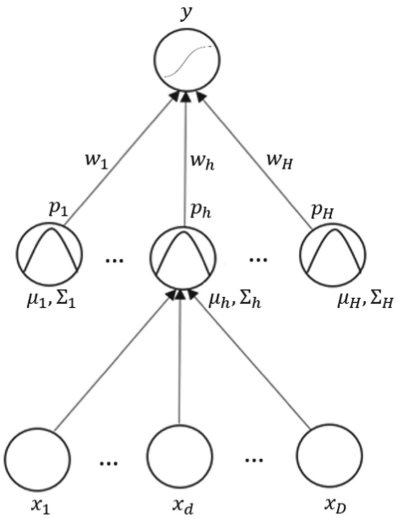


Fig. 1. The Architecture of the EBFDD network.

h^{th} hidden node to the output node. In the output node the modified hyperbolic tangent, $tanh$, activation function proposed in [15] is used as it avoids saturation:

$$y = 1.7159 \times tanh\left(\frac{2z(X)}{3}\right) \tag{2}$$

where $z(X)$ is the weighted sum of the outputs of the hidden layer, when X is the input vector:

$$z(X) = \sum_{h=1}^{H} w_h \times p_h\,(X) \tag{3}$$

The intuition behind the EBFDD network is that, it can be trained to learn a compact set of elliptical kernels that gather around the region in the input space where the normal data resides. This means that a trained model will output a high value for any *normal* query data point that falls within this region, and a low value for any *anomalous* query data point that falls outside this region. Thresholding this value will allow accurate anomaly detection.

EBFDD training begins with a pre-training phase that provides an initial set of positions for the H Gaussian kernels at the hidden layer. This phase uses the k-means [1] clustering algorithm, using Euclidean distance, applied to the training dataset. The resulting cluster centres are used to initialise the Gaussian kernel centres, μ_h, and covariance matrices, Σ_h, for each of the H hidden nodes. Initially these kernels are radial and assume equal variance in all directions. The number of hidden nodes in the network, H, (which is also the number of clusters found by k-means) is a hyper-parameter set before training starts.

The parameter values in the network (μ_h, Σ_h, and w_h) for each node in the network are then optimised using mini-batch gradient descent [1] (mini-batch

sizes are always set to 32) and the back-propagation of error algorithm [1]. The cost function to be minimised in this process is:

$$E = \frac{1}{2}\left[(1-y)^2 + \beta R_{\Sigma} + \lambda R_W\right] \tag{4}$$

where y is the output of the network; R_{Σ} and R_W are regularisation terms defined below in Eq. (5) and Eq. (6); and β and λ are hyper-parameters that control the influence of the regularisation terms.

This cost function is a weighted sum of three main terms that ensure the network learns a compact set of Gaussians that cover just the normal training data. The first term in this summation, $(1-y)^2$ encourages the network training process to learn a model that outputs a value as close as possible to 1 for instances belonging to the normal class.

The second term in the cost function, $R(\Sigma)$, regularises the variances of the Gaussian kernels in the hidden layer of the network. This term introduces the optimisation criterion of having the most compact set of Gaussians possible to represent the normal data. As the size of a Gaussian ellipsoid is dictated by the diagonal elements (i.e., the variances) of its covariance matrix, off-diagonal elements are excluded from the regularisation. R_{Σ} is defined as:

$$R_{\Sigma} = \sum_{h=1}^{H}\sum_{d=1}^{D}\left(\Sigma_h\left[d,d\right]\right)^2 \tag{5}$$

which is the squared L-2 norm [7] of the variances (i.e., the diagonal elements in each of the H Gaussian covariance matrices). Here, D denotes the dimensionality of the input space and $\Sigma_h\left[d,d\right]$ refers to the d^{th} diagonal element of the convariance matrix for the h^{th} hidden node.

The third term in the cost function, R_W, is the squared L-2 norm of the weight vector, W, connecting the hidden layer nodes to the output node. This discourages the weights from becoming so large that they would actually ignore the outputs from the hidden nodes. It also makes the EBFDD network robust to outliers in the training set [7]. R_W is defined as:

$$R_W = \sum_{h=1}^{H} w_h^2 \tag{6}$$

where w_h is the weight connecting the h^{th} hidden node to the output node.

We argue that a gradient descent training process based on the minimisation of the cost function in Eq. (4) will find a compact set of Gaussians, whose collective output is still high for the normal region of the input space and low, anywhere else (i.e., where we believe the anomalies would appear).

Based on the application of the back-propagation of error algorithm using gradient descent, the following update rules for learning the parameters of the EBFDD network have been derived in Eq. (7), Eq. (8) and Eq. (9) (the update rules are described for a single training instance for ease of reading, but are

easily expandable to a mini-batch scenario by summing across the instances in the mini-batch). Below, we have derived the learning rules for all 3 learnable parameters of the EBFDD network. The general structure of these learning rules is $a = a - \eta \times \frac{\partial E}{\partial a}$, where a is the learnable parameter, and E is the cost function defined in Eq. (4). Each of the weights, w_h, connecting a hidden unit to the output unit is updated using:

$$\begin{aligned} w_h &= w_h - \eta \times \frac{\partial E}{\partial w_h} \\ &= w_h - \eta \left[\frac{\partial \frac{1}{2}(1-y)^2}{\partial w_h} + \frac{\partial \frac{1}{2}\lambda R_W}{\partial w_h} \right] \\ &= w_h - \eta \left[\frac{\partial \frac{1}{2}(1-y)^2}{\partial y} \times \frac{\partial y}{\partial z_h(X)} \times \frac{\partial z_h(X)}{\partial w_h} + \frac{\partial \frac{1}{2}\lambda R_W}{\partial w_h} \right] \\ &= w_h - \eta[(y-1) \times [1.1439(1 - tanh(\frac{2z_h(X)}{3})^2)] \times p_h(X) + \lambda w_h] \end{aligned} \tag{7}$$

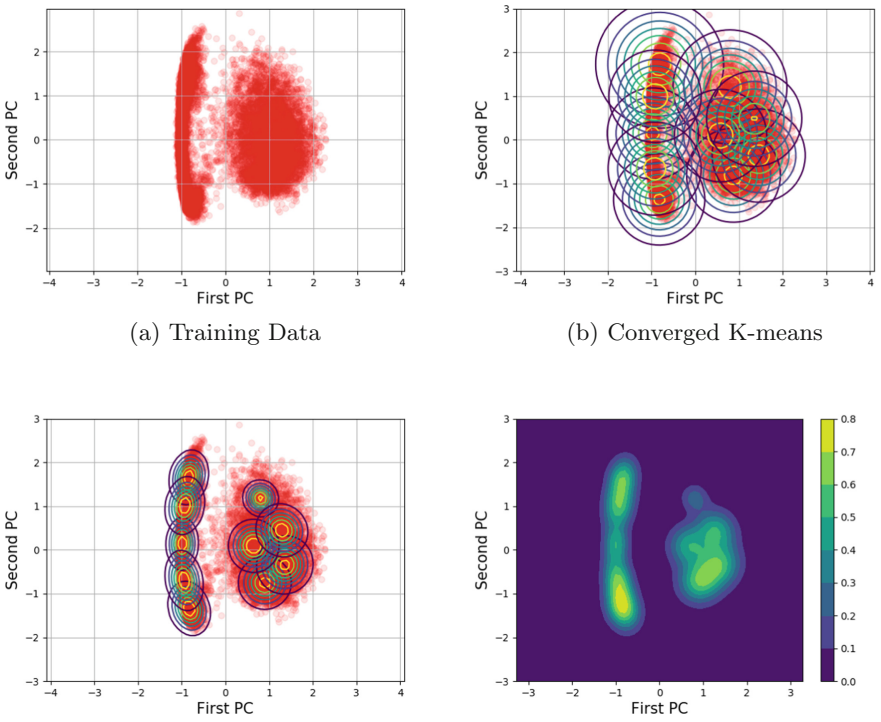
where η is the learning rate, and $z_h(X)$ is defined as in Eq. 3 (although we omit the summation because only the h^{th} hidden unit is relevant). All other terms are defined as before. Each of the kernel centres, μ_h, is updated using:

$$\begin{aligned} \mu_h &= \mu_h - \eta \times \frac{\partial E}{\partial \mu_h} \\ &= \mu_h - \eta \left[\frac{\partial \frac{1}{2}(1-y)^2}{\partial \mu_h} \right] \\ &= \mu_h - \eta \left[\frac{\partial \frac{1}{2}(1-y)^2}{\partial y} \times \frac{\partial y}{\partial z_h(X)} \times \frac{\partial z_h(X)}{\partial p_h(X)} \times \frac{\partial p_h(X)}{\partial \mu_h} \right] \\ &= \mu_h - \eta[(y-1) \times [1.1439(1 - tanh(\frac{2z_h(X)}{3})^2)] \times w_h \\ &\qquad \times [p_h(X)\Sigma_h^{-1}(X - \mu_h)]] \end{aligned} \tag{8}$$

And for the covariance matrix of the h^{th} Gaussian, Σ_h, the learning rule is:

$$\begin{aligned} \Sigma_h &= \Sigma_h - \eta \times \frac{\partial E}{\partial \Sigma_h} \\ &= \Sigma_h - \eta \times \left[\frac{\partial \frac{1}{2}(1-y)^2}{\partial \Sigma_h} + \frac{\partial \frac{1}{2}\beta R_\Sigma}{\partial \Sigma_h} \right] \\ &= \Sigma_h - \eta \times \left[\frac{\partial \frac{1}{2}(1-y)^2}{\partial y} \times \frac{\partial y}{\partial z_h(X)} \times \frac{\partial z_h(X)}{\partial p_h(X)} \times \frac{\partial p_h(X)}{\partial \Sigma_h} + \frac{\partial \frac{1}{2}\beta R_\Sigma}{\partial \Sigma_h} \right] \\ &= \Sigma_h - \eta \times [(y-1) \times [1.1439(1 - tanh(\frac{2z_h(X)}{3})^2)] \times w_h \\ &\qquad \times [p_h(X)(-\Sigma_h^{-T}(X - \mu_h)(X - \mu_h)^T \Sigma_h^{-T})] + \beta Diag(\Sigma_h)] \end{aligned} \tag{9}$$

where the function $Diag()$ diagonalises the covariance matrix Σ_h by turning all non-diagonal elements to 0.



(a) Training Data (b) Converged K-means

(c) Converged EBFDD (d) Decision Boundary

Fig. 2. Considering images of digits 0 and 1 from the MNIST dataset as class *normal* instances for training and using the first two principal components, starting from left to right, we have the visualizations of the transformed training data after applying PCA in red circles, the Gaussians after K-means pre-training, trained EBFDD network, and the decision surface of the trained EBFDD network. (Color figure online)

When Eq. 9 is applied, the resulting covariance matrix my not be invertible. This is a problem, as the inverse of the covariance matrices are used in both forward and backward propagation in the network. In such cases, we replace the covariance matrix generated with its closest positive semi-definite matrix found using the method proposed by Higham [8].

As explained earlier, the output of the EBFDD network, y, needs to be thresholded for the anomaly detection to take place. Any of the common thresholding schemes used with other anomaly detection algorithms [4] can be used with EBFDD networks. The thresholding method is not the focus of this paper as we have used the Area Under the Curve (AUC) of the ROC curve as the metric of evaluation. This gives us a good estimate of the performance of the algorithms assuming that we have a method of choosing the best threshold on the outputs for each one of the algorithms used in our experiments.

To illustrate the behaviour of an EBFDD network we present a simple example based on the MNIST dataset of handwritten digits [16], and we have used 10 Gaussians in this illustration. We have also used images of digits 0 and 1 as the normal class. So that the behaviour of the network can be visualised, the dimensionality of the input data is reduced to 2 using Principal Component Analysis (PCA) [1]. Figure 2a shows the training dataset of normal instances (images of digits 0 and 1). Figure 2b shows the output of the k-means algorithm after the pre-training phase, which provides the EBFDD network with an initial set of well positioned Gaussian kernels. Figure 2c shows the compact set of kernels covering the normal region after the EBFDD network is trained. Finally, Fig. 2d shows the decision surface that has been learned, where brighter colors correspond to higher outputs by the EBFDD network.

4 Experimental Method

This section describes the design of an experiment conducted to measure the performance of the EBFDD network and compare it with a number of state of the art anomaly detection approaches. In these experiments, we have trained EBFDD networks with 2 kernel options: (1) Elliptical Gaussian kernels and (2) Radial Gaussian Kernels. In the case of the latter, we are hoping that a less computationally expensive kernel, the radial kernel, would bring us an advantage. The state of the art algorithms in our experiments are the One-Class SVM (OCSVM) models [23], Auto-Encoders (AEN) [7], Gaussian Mixture Models (GMM) [1], and the Isolation Forests (iForest) [28]). These algorithms have performed anomaly detection tasks on a selection of datasets using *only normal* examples during training. The remainder of this section describes the datasets and performance metrics used in the experiments.

4.1 Benchmark Datasets and Anomaly Detection Scenarios

Following Emmott, et al. [5] we use fully labelled classification datasets to simulate anomaly detection scenarios so that performance metrics can be calculated. For each dataset used, where possible, we have considered different anomaly detection scenarios, where we consider different normal and anomalous classes to add more variety into our experiments.

We have chosen a random subset of the datasets used in [5] for our experiments[1]. All of these datasets come from the UCI repository[2]. From these datasets a number of different anomaly detection scenarios can be generated. These scenarios can be divided into two main groups:

[1] Since every dataset leads to multiple experiments (One vs All/ All vs One/ difficult scenarios in [5]) we chose a subset of the datasets available to reduce the computation required to run the complete set of experiments.

[2] https://archive.ics.uci.edu/ml/datasets.html.

– Binary Classification Datasets: In this case, the instances in each dataset belong to either of 2 classes. The normal class is selected based on the recommendations in [5], and only examples from this class are used for training.
– Multi-Class Classification Datasets: In this case, we have followed 3 approaches. First, is the one-vs-all approach where we select instances in a particular class as normal and treat all other classes as anomalous. Only normal instances are used during training. This is interesting, because it explores scenarios where anomalous instances might come from a variety of different distributions, but a compact distribution defines the normal class. So, if a dataset has K classes, we define K one-vs-all experiments. The second approach is the all-vs-one approach, where we choose instances of one class as anomalous and the instances of the other classes as normal. Only normal instances are used during training. This method explores scenarios where we might have a variety of definitions of the normal data coming from different distributions but a well-defined compact distribution generating the anomalous instances. Similarly, if a dataset has K classes, we define K all-vs-one experiments. The last type are more difficult scenarios recommended by Emmott, et al. [5] where an analysis is performed to find the most challenging partition of classes, in terms of separability, into normal and anomalous groups for the datasets they use in their experiments. We use these partitions to define the third set of scenarios in our experiments.

Table 1. The datasets used in our experiments

Dataset name	Number of rows	Number of classes	Number of features	Number of generated scenarios
Magic gamma telesacope	19021	2	10	1
Spambase	4602	2	57	1
Skin segmentation	245058	2	3	1
Steel plates faults	1941	7	27	15
Image segmentation	2311	7	18	15
Page blocks classification	5473	5	10	11
Statlog (Landsat satellite)	6436	6	36	13
Waveform database generator (Version 1)	5000	3	21	7

Table 1 summarises the datasets that have been used in our experiments. The number of rows, classes, and features in each one, as well as the number of anomaly detection scenarios extracted are shown. For all datasets feature values have been normalised to $[0, 1]$ using range normalisation.

4.2 Experimental Method and Performance Metrics

For each algorithm, dataset and scenario combination we use an experimental design that is similar to bootstrapping [10]. We extract an 80% sample (with no replacement) from the whole normal portion of a dataset to use to train a model. The remaining 20% is then mixed with all the anomalous data to constitute the test set. This is repeated 10 times and average performance results are reported. For every algorithm a grid search is performed to find the best set of hyper-parameters and the results of the best performing set of hyper-parameters are reported. All of our code and scripts used in our experiments are available on GitHub[3].

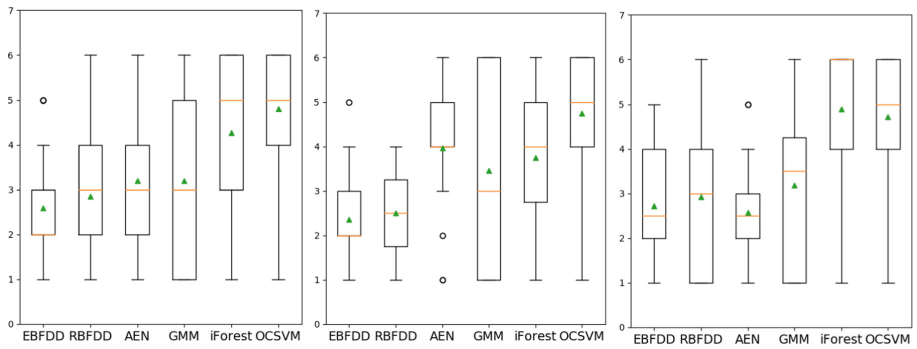
The tunable hyper-parameters used to generate these results, for each algorithm across all the datasets and scenarios, are presented in Table 2. It is important to mention that the values for H in the hyper-parameter grid search are determined by the dimensionality of each dataset, and in our experiments they do not exceed the dimensionality of the input. The range tested for H starts with 1 and then increases in steps of 5 all the way up to the dimensionality of the data, D. In the case of the Page Blocks Classification, and Magic Gamma Telescope datasets we reduce the step size to 2, and for the Skin Segmentation dataset the step size of 1, as the dimensionality of these datasets is rather low. As a result, the EBFDD/RBFDD, and AEN always tend to compress the input data in their hidden representation and similarly, the number of Gaussians in the GMM is also bounded by the dimensionality of the input, D. However, in the case of the Isolation Forest, we have chosen the same number of estimators across all the datasets. In the case of OCSVM, ν is the upper bound on the fraction of training errors and a lower bound of the fraction of support vectors, and γ is the kernel coefficient.

The area under the ROC curve [9] has been chosen as the evaluation metric in these experiments. This allows us to avoid defining a specific thresholding function to use with each model, but still measures the ability of a model to distinguish between normal and anomalous classes. The ROC curves are generated from the raw output of each model type.

[3] https://github.com/MLDawn/EBFDD.

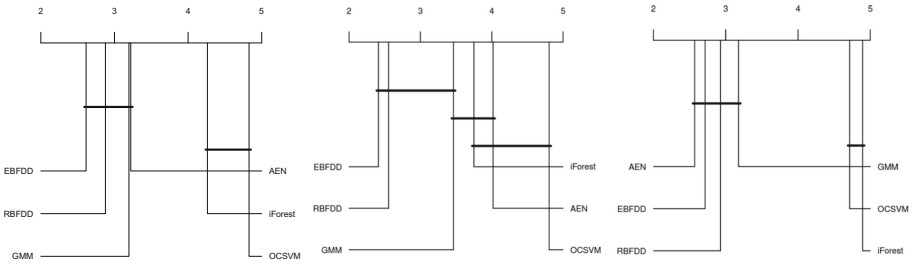
Table 2. Hyper-parameter ranges for each algorithm

Algorithms	Hyper-parameters	Investigated hyper-parameters
EBFDD	Number of kernels (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	λ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
RBFDD	Number of kernels (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	β	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	λ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
OCSVM	ν	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
	γ	[0.9, 0.5, 0.1, 0.05, 0.01, 0.001, 0.0001]
AEN	Number of hidden units (H)	[1, 5, 10, ..., D]
	η	[0.01, 0.001, 0.0001]
	Hidden layer activation functions	[sigmoid, relu]
	Output layer activation functions	[sigmoid, relu, linear]
	Error functions	[mean squared error, cross entropy]
GMM	Number of kernels (H)	[1, 5, 10, ..., D]
Isolation Forest	Number of estimators	[100, 200, 500, 800, 1000]



(a) All Scenarios (b) One vs All (c) All vs One

Fig. 3. The average rank of the algorithms across the experiments. On each box plot the median ranks are shown via the horizontal orange lines and the mean ranks are shown using green triangles. (Color figure online)



(a) All Scenarios (b) One vs All (c) All vs One

Fig. 4. Friedman's aligned test results where the statistical significance is $\alpha = 0.05$

5 Results and Discussion

The performances of each algorithm on each anomaly detection scenario for each dataset were compared and ranked. The distribution of the ranks for each algorithm for all anomaly detection scenarios are shown in Fig. 3a, and the average rank scores are summarised in Table 3[4]. The average ranks in Table 3 show that the EBFDD network has the lowest average rank over all experiments. It is also clear from Fig. 3a that the ranks of EBFDD have low variance indicating its consistent strong performance.

Table 3. Average rank across all experiments

	EBFDD	RBFDD	AEN	GMM	iForest	OCSVM
Average rank	2.59	2.85	3.20	3.20	4.26	4.80

Figure 3 also shows the distribution of ranks of the different approaches on the one-vs-all and all-vs-one scenarios (distributions for the other scenarios are not included as there are too few results to generate meaningful distributions). While the overall pattern is largely the same for these subsets as for the overall results, it is worth noting that there is a marked difference in the performance of the AEN models on the one-vs-all scenarios—performance is relatively poor—and on the all-vs-one scenarios—performance is quite good.

A Friedman test [6] with a significance level of $\alpha = 0.05$ was performed on the rank data. This showed a statistically significant difference in the performance of the different algorithms and so, following the recommendations of [6], a post-hoc Friedman aligned rank test, with $\alpha = 0.05$, was performed to further investigate the pairwise differences between the performance of the algorithms. Figure 4 summarises the results of these tests (tests were performed for all scenarios and independently for the one-vs-all and all-vs-one scenarios)[5].

[4] The full tables of results are provided in the supplementary material.

[5] The Win/Loss/Draw tables for the Friedman aligned rank test for $\alpha = 0.1$, 0.05, and 0.01 are provided in the supplementary material.

In all cases the performance of a group of the best performing algorithms cannot be separated in a statistically significant way. This group includes EBFDD, RBFDD, GMM, and (except for the one-vs-all scenario subset) AEN. The poorer performances of the OCSVM and iForest approaches are statistically significantly different to the performance of the other approaches. The poor performance of OCSVM is somewhat surprising, however, it is in line with previous benchmarks, for example [5].

Both the EBFDD and RBFDD algorithms accomplish their training through minimizing our proposed cost function in Eq. (4). We believe that the EBFDD out-performs RBFDD because of the more flexible nature of the decision boundaries that it is able to learn because of the use of elliptical kernels.

It is worth noting, however, that the cost of the good performance of the EBFDD algorithm is the very large number of parameters that must be learned in this model. For an EBFDD network, as the full covariance matrices are learnable, the number of trainable parameters is quite high:

$$(H \times D) + \left(H \times \frac{D(D+1)}{2}\right) + H \tag{10}$$

where H is the number of hidden nodes, and D is the dimensionality of the input. Moreover, if we consider radial kernels for the EBFDD networks, i.e. an RBFDD network, the number of trainable parameters is reduced to:

$$(H \times D) + (H \times D) + H \tag{11}$$

This means that training EBFDD networks can take longer than training equivalently sized RBFDD networks.

There is a minor difference in the number of trainable parameters in a GMM compared to an EBFDD network. As the weights used to combine distributions in a GMM sum to 1, for H Gaussians only $(H-1)$ weights need to be learned as the last weight can be computed by subtracting the sum of those weights from 1. As a result, the number of trainable parameters in a GMM is equal to:

$$(H \times D) + \left(H \times \frac{D(D+1)}{2}\right) + (H-1) \tag{12}$$

In addition the number of trainable parameters for the AEN is:

$$(H \times D) + H \tag{13}$$

It is not possible to talk about the number of trainable parameters for the iForest and OCSVM models in a meaningful way and so these are not compared. Moreover, as the implementations of algorithms used in these experiments come from different packages with differing levels of optimisation we do not believe that detailed comparisons of training times are appropriate.

As observed in Table 3, the empirical results show that the added complexity of the EBFDD network has made it a stronger anomaly detector compared to its simpler version that is the RBFDD network. In addition, even though the

EBFDD network and GMM have almost the same number of trainable parameters, it seems that the optimisation of the proposed cost function in the EBFDD network, which utilises gradient descent, allows it to find better solutions than the expectation maximisation approach used in a GMM.

6 Conclusions and Future Work

This paper presents a novel cost function, whose minimisation can adapt the Radial Basis Function (RBF) network into a one-class classifier. We have named the resultant anomaly detector the Elliptical Basis Function Data Descriptor (EBFDD) network. EBFDD utilises elliptical kernels that can elongate and rotate to allow it to learn sophisticated decision surfaces. An evaluation experiment conducted using a large set of datasets compared the EBFDD network with state of the art anomaly detection algorithms. Although statistical significance is not shown in all cases, the empirical results show that the EBFDD network has a better overall performance across all the experiments.

In future work, we plan to add recurrent connections to the EBFDD network architecture to allow contextual anomalies within streams to be identified, as well as the point anomalies identified by the current architecture. Moreover, we would like to investigate the idea of building a deep architecture where the EBFDD network and a deep network would be trained in an end to end fashion, the motivation being that the backpropagation signal for training the EBFDD network might push the deep network into learning better features for the Gaussian kernels of the EBFDD network to work with. An alternative would be to train the deep architecture separately and use the extracted features to train the EBFDD network, and this will also be explored. Finally, we will investigate how EBFDD can be adapted to handle concept drift scenarios in which the characteristics of what constitutes normal change over time.

References

1. Alpaydin, E.: Introduction to Machine Learning. MIT Press, Cambridge (2004)
2. Balupari, R., Tjaden, B., Ostermann, S., Bykova, M., Mitchell, A.: Real-time Network-based Anomaly Intrusion Detection. Nova Science Publishers Inc., New York (2003)
3. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer, Heidelberg (2007)
4. Chandola, V., Banerjee, A., Kumar, V.: Anomaly detection: a survey. ACM Comput. Surv. (CSUR) **41**(3), 1–58 (2009)
5. Emmott, A.F., Das, S., Dietterich, T., Fern, A., Wong, W.K.: Systematic construction of anomaly detection benchmarks from real data. In: Proceedings of the ACM SIGKDD workshop on outlier detection and description, pp. 16–21. ACM (2013)

6. Garcia, S., Fernandez, A., Luengo, J., Herrera, F.: Advanced nonparametric tests for multiple comparisons in the design of experiments in computational intelligence and data mining: Experimental analysis of power. Information Sciences **180**(10), 2044–2064 (2010). https://doi.org/10.1016/j.ins.2009.12.010. http://www.sciencedirect.com/science/article/pii/S0020025509005404, special Issue on Intelligent Distributed Information Systems
7. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. The MIT Press, Cambridge (2016)
8. Higham, N.J.: Computing a nearest symmetric positive semidefinite matrix. Linear Algebra Appl. **103**, 103–118 (1988). https://doi.org/10.1016/0024-3795(88)90223-6
9. Japkowicz, N., Shah, M.: Evaluating Learning Algorithms: A Classification Perspective. Cambridge University Press, Cambridge (2011)
10. Kelleher, J.D., Mac Namee, B., D'Arcy, A.: Machine Learning for Predictive Data Analytics. MIT Press, Cambridge (2015)
11. Kennedy, K., Namee, B.M., Delany, S.J.: Using semi-supervised classifiers for credit scoring. J. Oper. Res. Soc. **64**(4), 513–529 (2013)
12. Khan, S.S., Madden, M.G.: One-class classification: Taxonomy of study and review of techniques. CoRR abs/1312.0049 (2013). http://arxiv.org/abs/1312.0049
13. Kruegel, C., Vigna, G.: Anomaly detection of web-based attacks. In: Proceedings of the 10th ACM Conference on Computer and Communications Security., CCS '03, pp. 251–261. ACM, New York (2003). https://doi.org/10.1145/948109.948144
14. Laurikkala, J., Juhola, M., Kentala, E., Lavrac, N., Miksch, S., Kavsek, B.: Informal identification of outliers in medical data. In: Fifth International Workshop on Intelligent Data Analysis in Medicine and Pharmacology, vol. 1, pp. 20–24 (2000)
15. LeCun, Y.A., Bottou, L., Orr, G.B., Müller, K.-R.: Efficient BackProp. In: Montavon, G., Orr, G.B., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade. LNCS, vol. 7700, pp. 9–48. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-35289-8_3
16. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010). http://yann.lecun.com/exdb/mnist/
17. Li, J., Xiao, Z., Lu, Y.: Adapting radial basis function neural networks for one-class classification. In: IEEE International Joint Conference on Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE, Hong Kong, September 2008
18. Lin, W.M., Yang, C.D., Lin, J.H., Tsay, M.T.: A fault classification method by RBF neural network with OLS learning procedure. IEEE Trans. Power Deliv. **16**(4), 473–477 (2001). https://doi.org/10.1109/61.956723
19. Ma, R., Liu, Y., Lin, X., Wang, Z.: Network anomaly detection using RBF neural network with hybrid QPSO. In: IEEE International Conference on Networking, Sensing and Control (ICNSC). IEEE, Sanya, April 2008
20. Oliveira, A.L.I., Neto, F.B.L., Meira, S.R.L.: Combining MLP and RBF neural networks for novelty detection in short time series. In: Monroy, R., Arroyo-Figueroa, G., Sucar, L.E., Sossa, H. (eds.) MICAI 2004. LNCS (LNAI), vol. 2972, pp. 844–853. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24694-7_87
21. Rapaka, A., Novokhodko, A., Wunsch, D.: Intrusion detection using radial basis function network on sequences of system calls. In: Proceedings of the International Joint Conference on Neural Networks. IEEE, August 2003
22. Ruff, L., et al.: Deep one-class classification. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 4393–4402. PMLR (2018). http://proceedings.mlr.press/v80/ruff18a.html

23. Scholkopf, B., Platt, J.C., Shawe-Taylor, J., Smola, A.J., Williamson, R.C.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**, 1443–1471 (2001)
24. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT press, Cambridge (2001)
25. Staiano, A., Tagliaferri, R., Pedrycz, W.: Improving rbf networks performance in regression tasks by means of a supervised fuzzy clustering. Neurocomputing **69**(13–15), 1570–1581 (2006)
26. Tax, D.M.J., Duin, R.P.W.: Support vector data description. Mach. Learn. **54**(1), 45–66 (2004). https://doi.org/10.1023/B:MACH.0000008084.60811.49
27. Thottan, M., Ji, C.: Anomaly detection in ip networks. IEEE Trans. Signal Process. **51**(8), 2191–2204 (2003). https://doi.org/10.1109/TSP.2003.814797
28. Ting, K.M., Liu, F.T., Zhou, Z.: Isolation Forest. In: 2008 Eighth IEEE International Conference on Data Mining (ICDM), vol. 00, pp. 413–422 (2008). https://doi.org/10.1109/ICDM.2008.17
29. Wang, Y., Cai, W., Wei, P.: A deep learning approach for detecting malicious javascript code. Secur. Commun. Netw. **9**(11), 1520–1534 (2016)
30. Xiong, Y., Zuo, R.: Recognition of geochemical anomalies using a deep autoencoder network. Comput. Geosci. **86**, 75–82 (2016). Elsevier
31. Yan, W., Yu, L.: On accurate and reliable anomaly detection for gas turbine combustors: a deep learning approach. In: Annual Conference of Prognostics and Health Management Society (2015)