Radial Basis Function Data Descriptor (RBFDD) Network: An Anomaly Detection Approach

Mehran H.Z. Bazargani
University College Dublin(UCD)
Dublin, Ireland
mehran.hosseinzadehbazarga@ucdconnect.ie

Brian Mac Namee
University College Dublin(UCD)
Dublin, Ireland
brian.macnamee@ucd.ie

ABSTRACT

In this paper, we propose a modification to the standard Radial Basis Function (RBF) network that transforms it into a one-class classifier suitable for anomaly detection. We name this new approach the *Radial Basis Function Data Descriptor (RBFDD) network*. The RBFDD network is of interest as it has inherent adaptability in its architecture making it suitable for domains in which concept drift is a concern. Also, features learned by an RBFDD network (i.e., centers and spreads of Gaussian kernels and associated weights) provide us with a level of interpretability that has potential to be quite informative in terms of understanding the model learned and the reasoning behind flagging anomalies. In a set of evaluation experiments we compare the performance of the RBFDD network with some state of the art algorithms for anomaly detection over a collection of benchmark anomaly detection datasets. The results show that the RBFDD network is a promising approach and suggest potential for more investigations and promising directions for future work. We also investigate how RBFDD networks can be interpreted.

CCS Concepts

•Computing methodologies → Kernel methods; Neural networks;

Keywords

Anomaly Detection; Radial Basis Function; Neural Networks

1. INTRODUCTION

Chandola & Kumar [3] define *anomaly detection* as "*the problem of finding patterns in data that do not conform to expected behavior*". The main challenge of anomaly detection is the fact that there is very limited, or sometimes no, access to anomalous patterns in training data. This is why anomaly detection is sometimes referred to as the *one-class classification* problem [17]. As a result, the goal when building an automated anomaly detector is to train a model to recognize normal data and flag an anomaly when something fails to be recognized as normal.

In this paper, we propose a modification on the Radial Basis Function (RBF) network [1, 2], to make it applicable to anomaly detection by adjusting its loss function. We have used the RBF network as the foundation of our proposal because it is inherently quite suitable for explaining the underlying distribution of the training data through its use of a collection of Gaussian kernels. Based on the results of our initial experiments, it seems that our modified approach, the Radial Basis Function Data Descriptor (RBFDD) network, has successfully learned the distribution of the normal data and it can be used for identifying anomalies.

The remainder of the paper is structured as follows: In Section 2, we describe the common approaches to anomaly detection, the RBF network and some previous work that uses RBF networks for anomaly detection. In Section 3, we describe the proposed RBFDD network approach and demonstrate how the model learned by an RBFDD can be interpreted. We perform an evaluation experiment to benchmark the performance of the proposed RBFDD network against common approaches to anomaly detection. The setup of these experiments is described in Section 4, and the results from these experiments are presented and discussed in Section 5. Finally, Section 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 tackle the anomaly detection problem. Then we describe the standard RBF network approach. Next, we will study the existing RBF-network-based solutions to anomaly detection and explain the motivation behind the proposed RBFDD approach.

2.1 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) [8] algorithm to work with only examples of a single class: One-Class SVM (OCSVM) [16]. In fact Khan & Madden [17] go so far as to say that one-class classification algorithms and methods should be divided into OCSVMs and non-OCSVMs. On the non-OCSVM side, Auto-Encoder networks (AENs) [5], and all their variations have been increasingly used for anomaly detection [22, 23, 20].

ODD v5.0: Outlier Detection De-constructed London, UK

DOI: 10.1145/1235

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 idea is to separate all the data points from the origin in the feature space using a hyper-plane, 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 choice of the kernel plays an important role into the destination feature space and the quality of the separation. There are a few choices: polynomial, linear, Gaussian, etc. However, OCSVM has performed best when using the Gaussian kernel [17]. The main disadvantage of the OCSVM is the assumption that the anomalous data instances are concentrated around the origin, and this is called the problem of the origin [17].

On the other hand an Auto-Encoder (AEN) is an Artificial Neural Network (ANN) used for unsupervised learning, which has very well been used for anomaly detection. An AEN and all its variations, are used to learn a representation of the input data in a feature space of lower dimensionality. An AEN does not learn a discriminative but a generative model of the input data and after transforming the input data into a representation with reduced dimensionality, it strives to reconstruct the original input data from the dimensionally-reduced representation feature space. The error function used in the AEN is called the reconstruction error, and it measures the quality of the reconstruction by measuring the distance between the original input vector and the output, which is the reconstructed input. In anomaly detection, the magnitude of the reconstruction error, is the indication for whether the input belongs to the normal or anomalous class. The AEN is trained on the normal data and learns features that could best reconstruct the normal data and, as a result, the reconstruction error for the normal data tends to be quite small. However, if an anomalous data point/pattern is inputted to the AEN, the model fails to reconstruct it well, hence the reconstruction error would be high. It is common practice to put a threshold on the reconstruction error, where if the error is higher than this threshold, then the input is classified as anomalous, and as normal otherwise.

2.2 Radial Basis Function Networks

A Radial Basis Function (RBF) network [1, 2] 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 very idea of locality, implies the need for a distance function that measures the similarity between a given input data X, with dimensionality d, and the center, m_h, of every kernel unit h. The common choice for this measure is the Euclidean distance, $\|X - m_h\|$. The response function for these local units, needs to have a maximum where $X = m_h$, and decrease as X and m_h get less similar. The most commonly used response function for the RBF units is the Gaussian function:

$$p_h = exp[-\frac{\|X - m_h\|^2}{2s_h^2}] \quad (1)$$

where m_h and s_h denote the center and the standard deviation (i.e., spread) of the local unit h, defining a radially symmetric basis function.

The idea behind using such local models is that if we assume that, in the training data, there are clusters of data points, then for each one of these clusters we define a basis function, p_h. This basis function gets activated if X belongs to cluster h. The closer X is to m_h, the higher the value of the activation, p_h, will be.

As a pre-training stage, it is common practice to apply an unsupervised step, namely, a statistical clustering method (e.g., K-means clustering), to come up with a nice set of initial positions (i.e., centers) for the Gaussians, as opposed to random initialization. A simple heuristic for initializing the spreads, once the centers are found by the pre-training stage, is to find the farthest data point covered by a cluster, and then consider half of that distance as the spread for that cluster [1]. One-third of that distance is also possible, but it is common practice to be more on the conservative side. After these cluster centers and spreads are found, the outputs of the response functions, p_h, act as the inputs to a perceptron:

$$y = f(\sum_{h=1}^{H} w_h p_h + w_0) \quad (2)$$

where $f()$ is the applied non-linearity function in the output unit. Also, H, is the number of basis functions (i.e., kernels), w_h is the weight associated with unit h, and w_0 is the bias term. This whole structure is called a Radial Basis Function (RBF) network. The structure of a standard RBF network is shown in Fig.1:

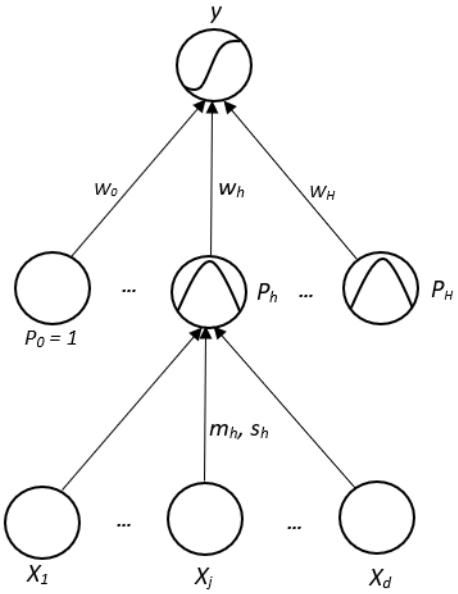

Figure 1: Standard Architecture of an RBF Network with H Gaussian Kernels

In Fig.1, X_j denotes the j^{th} feature in the input data X, which is a d-dimensional vector. Also, m_h, s_h, and p_h denote the mean, standard deviation, and the response (i.e., output) of the Gaussian kernel, h, with p_0 being the bias

unit. Finally, w_h denotes the weight of the connection between the Gaussian kernel, h, and the output neuron. The standard RBF network in Fig.1, is a supervised neural network. Provided a suitable non-linearity function (i.e., $f()$ in Eq.(2)), the RBF network should be able to classify the input X into its corresponding class. The hyperbolic $tanh()$ function is usually used for $f()$, and the error function to be minimized during training is usually the squared error function. This gives:

$$E = \frac{1}{2}(t-y)^2 \tag{3}$$

$$y = tanh(z) \tag{4}$$

$$z = \sum_{h=1}^{H} w_h p_h + w_0 \tag{5}$$

The learning, which takes place through the back-propagation algorithm [15], can be either batch-based or online. The learning update rules used for the online case are:

$$w_h = w_h - \eta(y-t)(1-y^2)p_h \tag{6}$$

$$w_0 = w_0 - \eta(y-t)(1-y^2) \tag{7}$$

$$m_h = m_h - \eta[(y-t)(1-y^2)w_h]p_h \frac{(X-m_h)}{s_h^2} \tag{8}$$

$$s_h = s_h - \eta[(y-t)(1-y^2)w_h]p_h \frac{(\|X-m_h\|)^2}{s_h^3} \tag{9}$$

where $\|X-m_h\|$ is the Euclidean distance between input data X and the center of unit h.

Depending on the problem at hand, there are other possible variations of the RBF network. It could have multiple output units, for the task of multivariate regression or a classic binary/multi-class classification task. The output activation function could be $tanh()$ or $sigmoid()$ for binary classification, and $softmax()$ for the case of multi-class classification. Last but not least, even though it is not customary for a standard RBF network to have more than one hidden layer, the RBF network could become deep and this might introduce more power to the model.

2.3 RBF Networks for Anomaly Detection

RBF networks have been applied to anomaly detection in two main ways. First, if examples of both normal and anomalous data are available during model training, the standard binary/multi-class classification RBF networks can be used. For example, in [12], the main focus of the work is on the optimization method for the task of anomaly detection. A hybrid optimization algorithm based on RBF networks has been proposed, which combines the traditional gradient descent algorithm with quantum-behaved particle swarm optimization to train the RBF network for anomaly detection. They have performed their experiments on the KDD Cup 1999 intrusion detection dataset, and during training, both normal and intrusion data have been used. Similarly in [14], 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, so the standard binary/multi-class classification approaches are no longer suitable. This problem can be addressed through modifying the dataset or through modifying the algorithm. As an example of the first type, in [13], an augmented training set is composed using the instances belonging to the normal class and random proxy anomalous patterns. Their work focuses on comparing the use of different classifiers, namely, Multi-Layer Perceptrons (MLPs), RBF networks, and the mixture of the two in terms of their performance. The task is to detect anomalies in time series data, by classifying every window of data as either normal or anomalous.

In [11], the RBF network algorithm is modified by combining it with the Support Vector Data Descriptor (SVDD) [19] algorithm. The hidden layer of a standard RBF network is used as a feature extractor and outputs for the subsequent transformed feature space are then used as inputs to the SVDD algorithm with a linear kernel. In their experiments, for each dataset, the instances of a particular class are considered as normal and everything else as abnormal. They have only used the instances belonging to the normal class during training and presented previously-unseen instances from both normal and anomalous classes during testing.

We have not found any work that has actually made modifications on the RBF network to convert it into a one-class classifier for anomaly detection. Our proposed approach (i.e., the RBFDD network), is targeting this gap, and in the next section we will go over its details.

3. THE RADIAL BASIS FUNCTION DATA DESCRIPTOR NETWORK

We have chosen RBF networks as the basis for building an anomaly detector due to the inherent capability of the RBF network to learn the underlying distribution of training data. An RBF network, however, is not capable of performing a one-class classification task, so we need to introduce some modifications to it to adapt it to this task. The intuition behind our approach is that we would like our kernels to gather around the subspace in the input space where the normal data reside, and then we would like for our learner to output a high value for any data appearing in that region. However, for any data point appearing in other regions of the input space, we would like for our learner to output a low value, thus distinguishing between the normal and anomalous data. A threshold should then be applied to this output to accomplish anomaly detection, where the input data resulting in an output higher than this threshold is classified as normal, and anomalous, otherwise.

We would like to make sure our kernels concentrate around the normal region of the input space, covering all of it, while excluding other regions. By setting an upper-bound on the spreads, s_h, of the Gaussians (i.e., kernels), we strive to converge to the *most compact* set of Gaussians that cover the normal region of the input space, while avoiding other unwanted regions where we believe the anomalous data to reside. 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.

In our proposal, the unsupervised pre-training phase has remained intact and identical to that of the standard RBF network (i.e., K-means clustering). The main contribution of this paper is the modifications that we have made to the error function in Eq.(3), to transform the standard RBF network into an anomaly detector (i.e., the RBFDD network). We propose the following cost function for the RBFDD network:

$$E = \frac{1}{2}[(1-y)^2 + \beta\|s\|_2^2 + \lambda\|w\|_2^2] \tag{10}$$

The cost function is a weighted summation of three main terms. Variable y in the first term, $(1-y)^2$, is the output of the RBFDD network. This first term in the cost function 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, $\|s\|_2^2$, is the L2-norm of the spreads of all the Gaussians. This term introduces our optimization criterion of having the most compact set of Gaussians possible to represent the normal data.

The third term, $\|w\|_2^2$, is the L2-norm of the weight vector connecting the RBF hidden layer units (i.e., Gaussians) to the output unit. This stops the weights from becoming so large that they would actually ignore the outputs from the hidden units. This also makes the RBFDD network robust to outliers in the training set [5]. In the output node of the network we use the hyperbolic $tanh()$ non-linear activation function proposed in [9] (i.e., $1.7159tanh(\frac{2x}{3})$), as it avoids saturation. We also remove bias terms output from the nodes in this network as we have found that they promote underfitting.

We argue that a gradient descent training process based on the minimization of Eq.(10) can result in the most 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). Here, we will explain how the RBFDD network can learn the normal class, using standard back-propagation, and gradient descent.

Let us consider Eq.(10), and the following:

$$y = 1.7159 \times tanh\left(\frac{2z}{3}\right) \tag{11}$$

$$z = \sum_{h=1}^{H} w_h p_h \tag{12}$$

where w_h is the weight connecting hidden unit h to the output unit and p_h is as defined in Eq.(1). Based on the application of the back-propagation of error algorithm using gradient descent the following update rules to learn the parameters of the RBFDD network (i.e., positions, and spreads of the Gaussians, and the weights) have been derived:

$$w_h = w_h - \eta[(1.1439)(y-1)(1-tanh(\frac{2z}{3}))^2)p_h + \lambda w_h] \tag{13}$$

$$m_h = m_h - \eta[(1.1439)(y-1)(1-tanh(\frac{2z}{3}))^2 (w_h p_h \frac{(X-m_h)}{s_h^2})] \tag{14}$$

$$s_h = s_h - \eta[(1.1439)(y-1)(1-tanh(\frac{2z}{3}))^2 (w_h p_h \frac{\|X-m_h\|^2}{s_h^3}) + \beta s_h] \tag{15}$$

As we explained earlier, the output of the RBFDD network, y, needs to be thresholded for the anomaly detection to take place. After training on the normal set is finalized, we make one last sweep over the entire training data, and measure the average output, $\bar{y}$, and the standard deviation of the output, σ_y, for these normal data. Finally, for a given test data sample, X, we would consider the following thresholding function to classify it as either normal or anomalous:

$$Class(X) = \begin{cases} \text{Normal} & \text{if } y \geq (\bar{y} - \sigma_y) \\ \text{Anomalous} & \text{if } y < (\bar{y} - \sigma_y) \end{cases} \tag{16}$$

3.1 Interpreting RBFDD Networks

It is useful to visualize the model learned by an RBFDD network to give an intuition about how it is partitioning the input space. To do this, we use the MNIST dataset of handwritten digits (described in detail in Section 4) using images of the digit 0 as the normal class with which an RBFDD network is trained. In order to be able to visualize the model learned, we have reduced the dimensionality of the input data to 2 using Principal Component Analysis (PCA) [1]. In the top left panel of Fig.2 we can see the digit 0 images transformed into 2-dimensions as blue dots. The centres of the 5 Gaussians learned by this RBFDD network are shown as red dots and their receptive fields with surrounding circles. The position and size of the Gaussians learned by the model illustrate the impact of the modifications made to the loss function to make the RBFDD network suitable for anomaly detection.

The panel to the top right of Fig.2 shows the output of the RBFDD network for the full feature space, where brighter colors correspond to higher outputs. We can see that the RBFDD network generates a high output for the area that is occupied by the normal data at the centre of the feature space, and that the strength of this output diminishes as inputs move away from this centre. The effect of the weights at the output layer can be seen in the interesting decision surface shapes towards the bottom of the image where multiple Gaussians are being combined.

Beyond these types of images, the learned features in an RBFDD network (i.e., the positions of the Gaussians, their spreads, and the learned weights) can give the model a further degree of interpretability. The Gaussian positions and spreads tell us something about the exemplars that the model has found to represent the normal class within the data. Moreover, the learned weight vector, can tell us about the importance of each Gaussian in explaining the underlying distribution of the normal data. As a result, we argue that all the learned features in the RBFDD network can give us a good level of tangible interpretability.

For comparison purposes, in the bottom row of Fig.2 we show the output of an OCSVM model (left) and AEN model (right) trained for the same problem. We can see the decision boundary of the OCSVM, which produces strong output in the middle of the feature space and gradually decreases towards the edges. On the right, the decision boundary of

the AEN is shown (the colours are reversed here as the AEN outputs reconstruction error where low errors refer to likely normal examples and high error to anomalies). We can see that the reconstruction error of the AEN gradually increases in value as we get away from the normal region.

In the next section, we will describe a set of experiments conducted to compare the performance of the proposed RBFDD network approach with that of the OCSVM and the AEN. We describe the experiment set up, the datasets used, the investigated scenarios, and the evaluation metrics used.

4. EXPERIMENTAL METHOD

In our experiments, we have trained RBFDD models as well as OCSVM and AEN models to perform anomaly detection tasks on a selection of datasets using only normal examples during training. We use fully labelled classification datasets in our experiments to allow calculation of performance metrics. For each dataset used, when possible, we have considered different anomaly detection scenarios (where we consider different normal or anomalous classes) to add more variety into our experiments. In this section we describe the datasets and the scenarios used in the experiments as well as the experimental method and performance metrics.

4.1 Datasets & Anomaly Detection Scenarios

We use four well-known datasets in our experiments:

- MNIST[1]: The MNIST database, has a training set of 60,000, and a test set of 10,000 gray-scale handwritten digit images of size 28x28. The task is to classify the digit present in each image [10].
- Fashion-MNIST[2]: Fashion-MNIST is a dataset of a training set of 60,000 and a test set of 10,000 greyscale images of 28x28 pixel images of clothing items. The 10 classes in this dataset are: t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot [21].
- KDD-CUP99[3]: A dataset used to build a network intrusion detector for distinguishing between "bad" connections (called intrusions or attacks) and "good" normal connections. We use a 10% sample of this dataset provided by its creators. This contains 494,021 rows. The 23 classes in this dataset represent different types of intrusion attacks [4].
- Wisconsin Breast Cancer[4]: The data (feature vectors of size 32) in this dataset are generated from digitized images of a fine needle aspirate of a breast mass. These features describe the characteristics of the nuclei in the cell that is in the image, which could relate to either a healthy or cancerous cell [4].

These datasets are made for classification problems, namely, binary and multi-class classification tasks. In order to use them for anomaly detection (i.e., one-class classification), in every dataset, we have considered the instances of a particular class as normal, and the instances from some other class as anomalous. In all cases, the models are trained using *only* the instances of the normal class, and during testing, we provide unseen samples from both normal and anomalous classes in order to measure the performance of the different models. The simulated scenarios per dataset are presented in Table 1.

We evaluate the performance of the proposed RBFDD approach on these 4 datasets and benchmark this against two well known state of the art methods, OCSVMs and AENs, which are discussed in Section 2.1. We have considered the OCSVM with Gaussian kernels, as this has proven to be the most successful kernel in the literature [17]. The hyper-parameters to search for in the case of OCSVM are ν, and γ. In the case of the AEN, we have considered a shallow AEN with one code layer. The hyper-parameters to search for in the case of the AEN are the number of neurons in the code layer (i.e., hidden layer), and the activation functions in the hidden layer (i.e., sigmoid or relu), and finally the output layer's non-linearity functions (i.e., sigmoid or relu). Finally, for the RBFDD network, the hyper-parameters that are needed to be tuned are the number of Gaussians in the hidden layer, and the coefficients of the cost function, β, and λ, whose values fall in the range of (0, 1].

4.2 The Evaluation Method

For a given algorithm (i.e., RBFDD network, OCSVM, and AEN), dataset and scenario, we use an evaluation method that is similar to bootstrapping [8]. For every combination of the hyper-parameters, we bootstrap a set of samples 20 times (with no replacement) from the whole normal portion of the dataset. Each time the bootstrap selects 80% of all the normal instances in the dataset (with no replacement). We use this sample for training the model. The remaining 20% is then mixed with all the anomalous data to constitute the test set, which is used to make predictions. As this makes the test set highly imbalanced, macro average F1-measure has been chosen as the appropriate evaluation metric. The F1-measure combines precision and recall scores for each class into one metric by obtaining the harmonic mean of both scores [6]. Moreover, macro averaging, as opposed to micro averaging, is used so the performance is not skewed towards the majority class (i.e., the normal class) [18].

5. RESULTS AND DISCUSSIONS

The experiment results are presented in Table 2. The tuned hyper-parameters used to generate these results, for each algorithm across all the datasets and scenarios, are presented in Table 3. To make the hyper-parameter table manageable in size, we used some acronyms to name the hyper-parameters. For the AEN, (#H, #EP, HA, OA) refer to (number of hidden units, number of training epochs, hidden layer activation function, and the output layer activation function). RBFDD almost shares the same terminology of hyper-parameters with the AEN, except we have η here, which is the learning rate, and the rest have been introduced before.

We have ranked the macro F1-measures in each column in Table 2, and averaged them across all the datasets, and presented the result in the last column.

While the RBFDD network receives the lowest average rank across the three techniques described, its performance

[1]http://yann.lecun.com/exdb/mnist/
[2]https://www.kaggle.com/zalando-research/fashionmnist
[3]http://kdd.ics.uci.edu/databases/kddcup99/kddcup99.html
[4]https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+(Diagnostic)

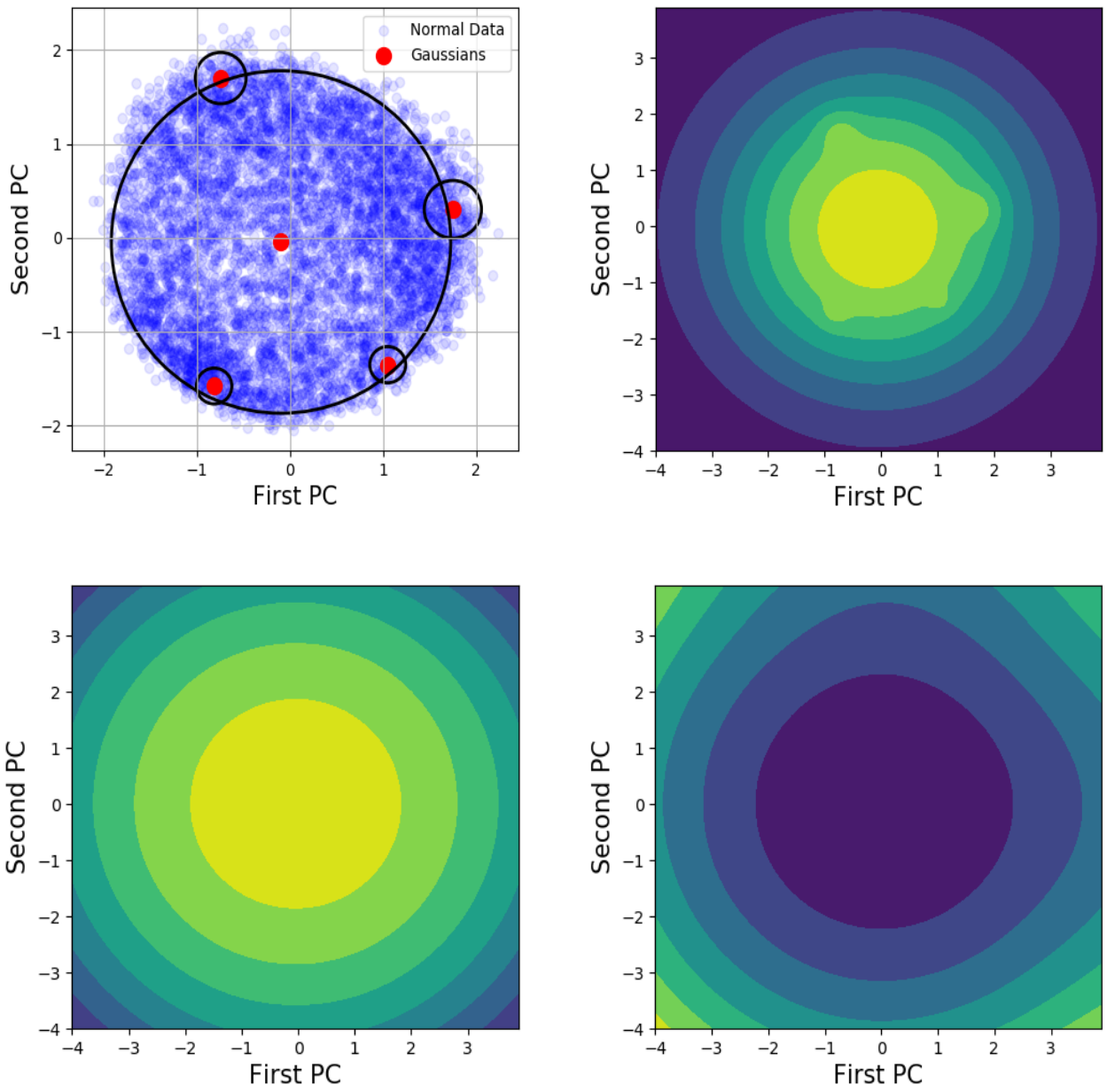

Figure 2: Considering images of digit 0 from the MNIST dataset as class "Normal" instances for training and using the first two principal components (i.e., First PC and Second PC), here are the visualizations of the trained RBFDD network (Top Left), the corresponding decision boundary of the same RBFDD network (Top Right), the decision boundary of the trained OCSVM (Bottom Left), and the decision boundary of the trained AEN (Bottom Right)

is in many cases competitive (the macro averaged F1-scores are very close) and it achieves the best performance in two cases. The two most challenging scenarios are the T-shirt versus Shirt scenario on the Fashion-MNIST dataset and the 7 versus 1 scenario on the MNIST dataset. All 3 algorithms perform poorly in these scenarios but the gap between the best performing approach, OCSVM, and the other two (i.e., AEN and RBFDD) is substantial.

However, overall we believe that these results show the value of the RBFDD proposal as an interesting line of research with advantages over other anomaly detection approaches and plenty of scope for improvement. We discuss both of these in the next section.

Datasets	Experimental Scenarios Normal	Anomalous
MNIST	Digit 0	Digit 1
	Digit 7	Digit 1
Fashion-MNIST	T-shirts/tops (T)	Ankle boots (B)
	T-shirts/tops (T)	Shirts (S)
Wisconsin Breast Cancer	Healthy cells (H)	Cancerous cells (C)
KDD-CUP99 (10%)	Normal Data (N)	Satan Attacks
	Normal Data (N)	Smurf Attacks

Table 1: The Experiment Scenarios for Each Dataset

	MNIST		Fashion-MNIST		KDD (10%)		Wisconsin	Avg.
	0 - 1	7 - 1	T - B	T - S	N - Satan	N - Smurf	H - C	Rank
OCSVM	0.957 (3)	0.946 (1)	0.949 (2)	0.605 (1)	0.971 (2)	0.987 (1)	0.971 (3)	1.857
AEN	0.959 (2)	0.827 (2)	0.959 (1)	0.505 (3)	0.967 (3)	0.984 (2)	0.978 (1)	2.140
RBFDD	0.987 (1)	0.769 (3)	0.943 (3)	0.545 (2)	0.987 (1)	0.982 (3)	0.975 (2)	2.285

Table 2: The Experiment Results per each Dataset and the Average Ranking

		OCSVM (ν, γ)	AEN (#H, #EP, HA, OA)	RBFDD (#H, #EP, η, β, λ)
MNIST	0 - 1	0.1 - 0.0001	392 - 10 - sigmoid - sigmoid	39 - 20 - 0.01 - 0.5 - 0.5
	7 - 1	0.001 - 0.001	196 - 20 - relu- sigmoid	392 -20 - 0.01 - 0.5 - 0.5
Fashion-MNIST	T- B	0.1 - 0.01	588 - 40 - sigmoid - sigmoid	5 - 5 - 0.001 - 0.01 - 0.01
	T- S	0.001 - 0.1	196 - 40 - relu - sigmoid	5 - 5 - 0.001 - 0.05 - 0.05
KDD (10%)	N - Satan	0.1 - 0.001	87 - 40 - relu - sigmoid	58 - 20 - 0.01 - 0.1 - 0.5
	N - Smurf	0.1 - 0.001	87 - 40 - relu - sigmoid	87 - 20 - 0.01 - 0.5 - 0.1
Wisconsin	H - C	0.01 - 0.0001	2 - 10 - sigmoid - relu	10 - 5 - 0.001 - 0.05 - 0.01

Table 3: Result of the Hyper-parameter Search for Each Algorithm per Experiment

6. CONCLUSIONS & FUTURE WORK

In this work, we have proposed a new method for anomaly detection, which is based on the standard RBF network. In order to modify an RBF network for one-class classification we have proposed a cost function that tries to maximize the output of the network, given the training set comprised of exclusively the normal class samples, while keeping the size of all Gaussians as small as possible. This should learn the most compact set of Gaussians possible to cover the normal input subspace and avoid any coverage of the non-normal regions of the input space. We have also regularized the weight vector to avoid over-fitting, and force better positioning of the Gaussians in the input space.

Based on the results of the experiments, it seems that these modifications have turned the standard RBF network into an anomaly detector, which is capable of one-class classification.

Based on the average ranking computed in Table.2, our proposed algorithm has the last position in the competition with the state of the art algorithms (i.e., OCSVM, and AEN). However, in most cases, the difference in the computed macro F1-measure is indeed negligible and even in a couple of scenarios, the RBFDD approach has ranked the first. We believe that the proposed RBFDD network, that is based on the standard RBF network, is still an attractive option for anomaly detection, due to the following reasons:

By nature, an RBF network strives to divide the input space, occupied by the training data, between a limited set of kernels. The RBFDD network takes advantage of this property, and learns the most compact set of Gaussians to cover the normal region and no where else. As a result, the features learned by the RBFDD network, provide us with a good level of interpretability. The learned positions of the kernels, can tell us about the underlying distribution of the training data. Last but not least, the magnitude of the weights connecting the output of each kernel to the output unit of the RBFDD network, can tell us about the importance of each kernel in explaining the underlying distribution of the normal data.

In addition, one of the main challenges in anomaly detection, is the changeability of what is considered to be normal, as a function of time (i.e., concept shift/concept drift [7]), which requires an agile adaptation by the anomaly detector. Adapting the architecture of the RBFDD network, in terms of updating the positions of the kernels, adding new or even removing certain kernels, is quite straightforward and relatively easy compared to the adaptation of state of the art algorithms (e.g., AEN, OCSVM) to the new definition of normal.

Even though, the results in Table.2, show that the proposed RBFDD network has the potential to be a strong anomaly detector there is plenty of room for improvement. The current RBFDD approach makes an assumption that the input features have equal variances, and no correlation with one another. Although this assumption offers the advantage of some computational efficiency, it is quite limiting as Gaussians are forced to always have a radial shape, with no flexibility to either elongate or even rotate. We are planning to introduce the assumption of unequal variances per input feature, and also correlation between the input features, which would dictate using a non-diagonal covariance matrix. This should make the model much more flexible as it

will no longer be constrained by radial kernels—kernels will be ellipsoids and capable of rotation. This property could lead to more interesting decision boundaries and better performance.

The current implementation relies on a simple thresholding of the output, produced by the RBFDD network to classify anomalies. We believe that a more sophisticated thresholding could considerably improve the performance. Similarly the current implementation is constrained to a single hidden layer connected to an output layer. In the future we will investigate deeper RBFDD network architectures. Finally, we will explicitly investigate the effectiveness of RBF networks in the presence of concept drift.

7. ACKNOWLEDGMENTS

This work was supported by Science Foundation Ireland under Grant No. 15/CDA/3520 and Grant No. 12/RC/2289. We would also like to thank our colleagues, Ellen Rushe and Arjun Pakrashi for their invaluable feed-backs and support.

References

[1] E. Alpaydin. *Introduction to Machine Learning*. MIT Press, 2004.

[2] C. M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2007.

[3] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM Computing Surveys (CSUR)*, 41(3), July 2009.

[4] D. Dheeru and E. Karra Taniskidou. UCI machine learning repository, 2017.

[5] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, Cambridge, Massachusetts, London, England, 2016.

[6] N. Japkowicz and M. Shah. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press, 2011.

[7] Y. Kadwe and V. Suryawanshi. A review on concept drift. *IOSR Journal of Computer Engineering (IOSR-JCE)*, 17:20–26, 2015.

[8] J. D. Kelleher, B. Mac Namee, and A. D'Arcy. *Machine Learning for Predictive Data Analytics*. MIT Press, 2015.

[9] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Muller. *Neural Networks: Tricks of the Trade*, chapter Efficient Backprop. Springer, 1998.

[10] Y. LeCun and C. Cortes. MNIST handwritten digit database. 2010.

[11] J. Li, Z. Xiao, and Y. Lu. Adapting radial basis function neural networks for one-class classification. In *Neural Networks, 2008. IJCNN 2008. (IEEE World Congress on Computational Intelligence). IEEE International Joint Conference on*, Hong Kong, China, September 2008. IEEE.

[12] R. Ma, Y. Liu, X. Lin, and Z. Wang. Network anomaly detection using rbf neural network with hybrid qpso. In *IEEE International Conference on Networking, Sensing and Control(ICNSC)*, Sanya, China, April 2008. IEEE.

[13] A. Oliveria, F. Neto, and S. Meira. Combining mlp and rbf neural networks for novelty detection in short time series. In *Lecture Notes in Computer Science (LNCS)*, volume 2972, pages 844–853. Springer, April 2004.

[14] A. Rapaka, A. Novokhodko, and D. Wunsch. 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.

[15] R. Rojas. *Neural Networks: A Systematic Introduction*. Springer, 1996.

[16] B. Scholkopf, J. C. Platt, J. Shawe-Taylor, A. J. Smola, and R. C. Williamson. Estimating the support of a high-dimensional distribution. *Neural Computation*, 13:1443–1471, 2001.

[17] S. S.Khan and M. G.Madden. One-class classification: Taxonomy of study and review of techniques. *The Knowledge Engineering Review*, pages 1–30, 2014.

[18] M. Sokolova and G. Lapalme. A systematic analysis of performance measures for classification tasks. *Information Processing & Management*, 45(4):427–437, 2009.

[19] D. M. J. Tax and R. P. W. Duin. Support vector data description. *Mach. Learn.*, 54(1):45–66, Jan. 2004.

[20] Y. Wang, W. Cai, and P. Wei. A deep learning approach for detecting malicious javascript code. *Security Comm. Network*, pages 1520–1534, 2016.

[21] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017.

[22] Y. Xiong and R. Zuo. Recognition of geochemical anomalies using a deep autoencoder network. *Computers Geosciences, Elsevier*, pages 75–82, 2016.

[23] W. Yan and L. Yu. On accurate and reliable anomaly detection for gas turbine combustors: A deep learning approach. In *Annual Conference of Prognostics and Health Management Society*, 2015.