End semester report on R & D Project (NU 302)

Academic Year- 2020-21

on

NOVELTY DETECTION

A dissertation

Submitted in partial fulfillment of the requirements for the award of the degree Bachelor of Technology

by

1.	Muskan Goel	BT18GCS188	C3
2.	Shreya Chauhan	BT18GCS154	C4
3.	Anoop Gupta	BT18GCS186	C3

Under supervision of

Prof. Sudip Sanyal

NIIT University, Neemrana, Rajasthan-301705

May 2021

DECLARATION BY STUDENT(S)

We hereby declare that the project report entitled **NOVELTY DETECTION** which is being submitted for the partial fulfilment of the Degree of Bachelor of Technology, at NIIT University, Neemrana, is an authentic record of my/our original work under the guidance of **Prof. Sudip Sanya**l. Due acknowledgements have been given in the project report to all other related work used. This has previously not formed the basis for the award of any degree, diploma, associate/fellowship or any other similar title or recognition in NIIT University or elsewhere.

Place: NIIT University ,Neemrana, Rajasthan
Date: 17th May'2021

Shreya Chauhan	BT18GCS154	Btech(CSE)	Signature
Muskan Goel	BT18GCS188	Btech(CSE)	Signature
Anoop Gupta	BT18GCS186	Btech(CSE)	Signature

CERTIFICATE BY SUPERVISOR(S)

This is to certify that the present R&D project entitled **Novelty Detection** being submitted to NIIT University, Neemrana, in partial fulfilment of the requirements for the award of the Degree of Bachelor of Technology, in the area of CSE, embodies faithful record of original research carried out by **Muskan Goel, Shreya Chauhan and Anoop Gupta**. They have worked under my guidance and supervision and that work has not been submitted, in part or full, for any other degree or diploma of NIIT or any other University.

Place: NIIT University ,Neemrana, Rajasthan
Name of the Supervisor(s) with signature: Prof. Sudip Sanyal
Date: 17th May'2021

ACKNOWLEDGEMENT

We have taken efforts in this project. However, it would not have been possible without the kind support and help of many individuals and organizations.We would like to extend our sincere thanks to all of them.

We are highly indebted to NIIT University for their guidance and constant supervision as well as for providing necessary information regarding the project & also for their support in completing the project.

We would like to express our gratitude towards our parents & Prof Sudip Sanyal for their kind co-operation and encouragement which helped us in completion of this project.

We would like to express our special gratitude and thanks to respected faculty members for introducing us to this project.

TABLE OF CONTENTS

INTRODUCTION

A data stream is an order sequence of unbounded (ever-growing) data that has a rapid change of flow over time. The evolving nature of data causes essentially the appearance of new concepts. These could be new normal patterns that the model or the system hasn't seen before or novel concepts that are considered abnormal. To detect these emerging changes, we need to continuously observe the movement of the streaming data.

In today's time, Novelty Detection is an essential problem in the world of data in different sectors such as fraud detection, machine failure, network security which needs to be dealt with with greater attention.

Novelty detection can be defined as the recognition of novelty, outlier or abnormality patterns inserted in a colossal dataset which is a very useful ability for the learning algorithms, where the system only learns about inputs that it has not seen or observed before. In general, novelty (anomaly, outlier, or exception) is a pattern in the data that does not obey the expected behavior. This is what our research mainly deals in.

Before we go any further, let's understand what is the crucial dissimilarity between outlier detection, anomaly detection, and novelty detection. Novelty, anomaly, and outlier detection have a strong mutual relationship between them. In some contexts, these three terms seem to have the same meaning and are used indifferently. In general, the terms namely anomaly and outliers are more similar and frequently used to express very close problems.

Novelty, anomaly, and outlier detection are terms related to finding patterns that are different from the normal or the usual ones. On one side the terms anomaly or outliers give the idea of an unexpected pattern, the term novelty indicates an arising a new concept that needs to be included into the normal pattern so that the model used can identify it as a new class, thereby making a separate class of it.

However, on the other hand, anomaly detection can be defined as the task of finding patterns in the datasets that do not follow expected behavior or pattern. These patterns or behaviors are also considered to be anomalies, outliers, exceptions, aberrations, or peculiarities.

However, Novelty Detection also aims to detect unobserved patterns in the given dataset. However, this term is distinguished from anomaly detection, since, in the first, the novel patterns are typically incorporated into the normal model after their detection.

The principal objective of novelty detection is to identify abnormal patterns or behaviors which are not about the normal state of a system. Novelty events do not occur frequently but can have very significant consequences to overall system operation.

Another area where novelty detection is especially applicable is where an important class is being under-represented or is not being fundamentally recognized in the dataset so that a classifier cannot be trained to reliably recognize that class. Some typical instances of this problem include medical diagnosis, where there are hundreds or maybe thousands of test results, however very few of them show the indications of a specific disease/s, and machine fault recognition, where there might be many hours of operation between failures.

The term novelty detection mainly deals with recognizing inputs that differ in some way from those that are usually seen, that is, that has not been recognized by the system. This indicates that the overall performance of the system will be poor for those particular sets of classes. In some circumstances, such as medical data and fraud detection, it is often minutely the class that is not recognized in the data, the disease or potential fraud, that the network should detect. In novelty detection systems the algorithm is only trained on the negative dataset examples where no other class is present, and then the algorithm detects inputs that do not suit the model that it has collected, that is, members of the novel class.

In this paper we propose and evaluate a technique for novelty detection in data streams, section 2 deals with the problem statement and objective whereas section 3 is about the observations made about various supervised learning algorithms used by other researchers to carefully try to find the best method, section 4 covers methods of novelty detection in supervised neural networks and the gaps or challenges that might occur with our proposed model, this section will also include some of the supervised learning methods depicting why we chose our proposed model.

PROBLEM STATEMENT AND OBJECTIVE

It has always been challenging to detect any novelty or unseen data while updating or modifying the model to make it efficient and stable, especially in high-dimensional data. Being a powerful and highly used method, supervised machine learning assumes that all classes are known a priori. The response of some input samples is already known. But this is not the case in streaming data. In the data streams, some novel classes may appear with time due to some changes in the statistical distribution of data. Now in this dynamic streaming environment, the new classes or

the new examples either belong to the target class or not. So the classes which are not explained by the model or are not available at the training phase are said to be a novelty. Here the data is not already tagged with the correct classes. This Novelty Detection is the problem of recognizing inputs that are not usually seen by the trained machine or differ in some way.

Given a set of training examples, our main challenge is to identify those novel concepts in the data streams, and for that, we need to identify the outlying classes in a continuous learning scenario. In this way, any time any abnormal or unusual data is detected, it is identified as a novelty. However, the known concepts may disappear or evolve with the arrival of new concepts. Since these concepts are rarely constant in this non-stationary environment, novelty detection in data streams has to deal with many challenges, including the presence of:

1. Concept Drift, which makes it difficult to differentiate between the unknown and known concepts due to continuous significant change in the data distribution.
2. Outliers, which are mixed up with the new concepts. Assuming that all the classes are previously defined and considering them as noise to the system, we ignore the fact that there might be an occurrence of new concepts.
3. Concept evolution, when there is an appearance of more problem classes (novel classes), then these novel concepts have to be incorporated into the decision model.

In the process of finding these novel classes, it is very important to find the best method that can deal with all these challenges mentioned above and evaluate the accuracy of the unusual data being driven to their respective classes. So to address this problem, the agenda or the purpose of this study would be to carry out a lot of experiments and to analyze and describe the methods and the algorithms for novelty detection in data stream scenarios. Besides, our main focus will be only on one of the concepts of Novelty Detection, i.e., the classification technique.

LITERATURE REVIEW

According to [3], Yuhua Li, et al., explained a detailed evaluation of both the Multilayer Perceptron (MLP) neural network and the RBF (Radial Basis Function), they evaluated the performance of these two on mainly 3 case studies or data set, first being the mathematical model case study followed by the diesel cooling and engine misfire case studies. The Study gives strong evidence regarding how both the RBF and the MLP provide a similar level of performance for basic classification problems i.e when it comes to unknown faults or novelty the

result indicates that either a well-trained MLP or RBF can be expected to perform well when classifying novel samples that fall in the range of the training data set only. However, the differences in these classifiers can be expected to have a greater impact on extrapolation performance(when samples to be classified are never seen before or outside the data set. They considered a two-dimensional measurement space, to explain the differences in the working of both the classifiers.

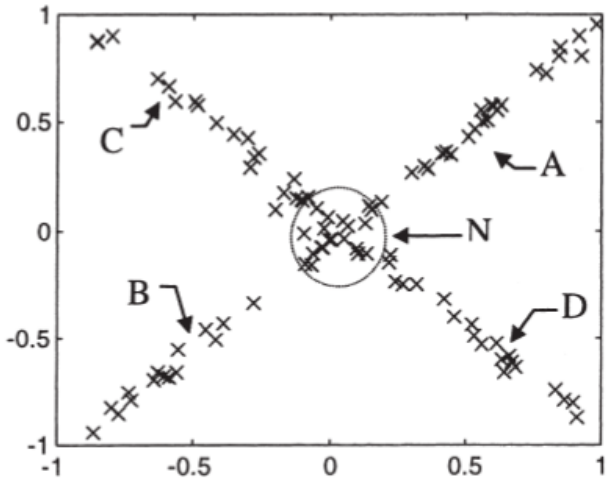

Fig 1: Distribution of three classes in two dimension

As shown in the figure within the measurement space, the distribution of samples of normal condition (region N)and fault conditions (region A and B for fault 1, region C, and D for fault 2) is supposed to be already known.

The networks for this fault classification problem had two input neurons and three output neurons and are considered to be trained using data streams already available. Once training is complete, the MLPs partition the input space into hyperplanes, its decision boundaries are unbounded. They observed the decision surfaces for each MLP class for example, for the decision surface for ‘fault 1’ in Figure 5 (a), the classifier will produce high values for samples not only in the region of the training samples but also for the samples which are some distance away: here ‘some distance’ implies maybe an infinitely far away in some directions. On the other hand, since RBFNs partition the input space using hyperspheres, their decision boundaries are bounded. The decision surfaces for each RBFN class are shown in Figure 6. From the figure, we can see that RBFN classifiers produce high values for samples only from the regions covered by training samples: Figure 6(d) clearly shows that the decision boundaries are closed.

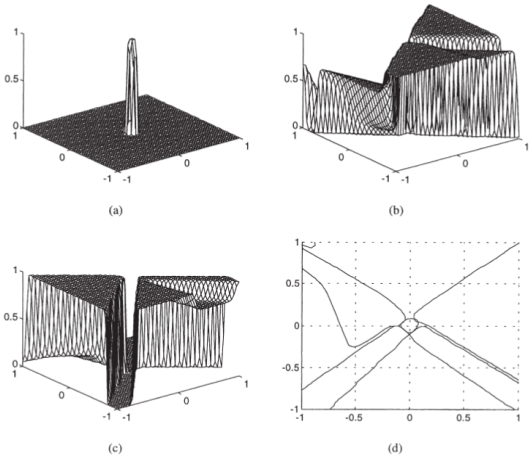

Fig 2: Decision surface of MLP.
(a)decision surface for “normal condition”
(b)decision surface for “fault1”
c)decision surface for “fault2”
(d)contour of decision surfaces.

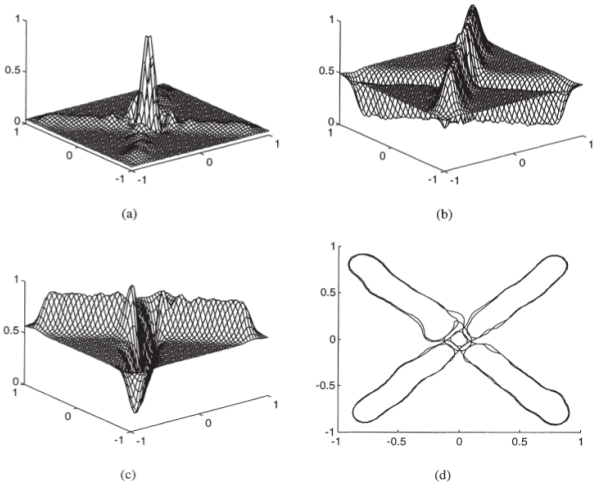

Fig 3: Decision surface of RBFN.
(a)decision surface for “normal condition”
(b)decision surface for “fault1”
(c)decision surface for “fault2”
(d)contour of decision surfaces.

Now it is clear from Figures 2 and 3 that both MLP and RBFN(Radial basis function network) could provide good classification results for the novel samples in the region of the original training data samples. Since the decision surfaces of the MLP are smoother than those of the RBFN, MLPs usually provide more accurate classification for these ‘in-range samples’: this performance has been verified by some investigators (Musavi et al., 1994). However, at the same time RBF plays a good role here that if a novel ‘out-of-range fault occurs in the diagnosed system, the unexpected samples will lie outside the regions N, A, B, C, or D: while the RBFN may still provide an ‘appropriate’ result for such faults/novel classes, the MLP is less likely to do so. Their study confirms the theoretical prediction that RBFNs can provide many accurate classification results for unknown faults if they are well separated from the known classes, but still performs poorly or not up to the mark for those unknown faults which overlap with known classes which are our main goal. On the other hand, MLPs always try to classify the samples of unknown faults into known classes since the decision boundaries of the known classes are unbounded.

During the past few decades, artificial neural networks (ANNs) have been majorly applied in many disciplines of engineering. Aref Hashemi Fath, et al., presented the two most powerful, intelligent, and reliable models for calculating the solution gas-oil ratio. The models are two different types of ANN algorithms- multilayer perceptron (MLP) and radial basis function (RBF) neural network.

They explained both MLP and RBF neural networks as a general class of neural networks called feed-forward neural networks, where information processing in the network structure follows just one direction from the input neurons to the output neurons. There are several important differences between MLP and RBF neural networks-

- RBF-NNs are simpler than MLP-NNs
- RBF-NNs are usually easier to be trained in compared to MLP-NNs because of the fixed three-layer structure.
- MLP-NNs work globally and the network outputs are decided by all the neurons. And on the other hand, RBF-NNs act as local approximation networks, and the network outputs are obtained by some specified hidden neurons in indefinite local accessible fields.
- MLP and RBF neural networks have different techniques of classification. In MLP-NNs, clusters are separated by hypersurfaces, while in RBF-NNs hyperspheres are used to separate clusters.

In [2], Approx 710 experimental data sets covering a very wide range of experimental conditions were collected to develop the MLP and RBF model from various literature sources including Glaso, Bello, et al., Mahmood, and Al-Marhoun, Dokla and Osman, Omar and Todd, and many more. Appropriate selection of the training set can heighten the efficiency and accuracy of the neural network in both testing and training phases. The training set is used to create the structure of the model and a test set is utilized to investigate the predictive validity and reliability of the proposed model. To develop these MLP and RBF models, 80% (568 data sets) of the original data sets were chosen as the training set and the remaining 20% (142 datasets) were considered as the test set.

Many statistical parameters were used to compare all the models including the Average Percent Relative Error (APRE), Average absolute percent relative error (AAPRE), Maximum absolute percent relative error (Max. APRE), Root mean square error (RMSE), Standard deviation (SD), and Correlation coefficient (R).

The figure below shows the comparison between the values of correlation coefficients of all the

proposed models which are calculated as -

$$R = \sqrt{1 - \frac{\sum_{i=1}^{n} \left(x_{iexp} - x_{ipred} \right)^2}{\sum_{i=1}^{n} \left(x_{ipred} - \bar{x} \right)^2}}$$

Where $\bar{x}$ refers to the average experimental value and can be written as:

$$\bar{x} = \frac{1}{n} \sum_{i=1}^{n} x_{iexp}$$

As we can see, MLP and RBF models hold the highest correlation coefficients(0.96953 and 0.97818, respectively). So the higher performance and accuracy of these 2 proposed models prove that they were successfully trained.

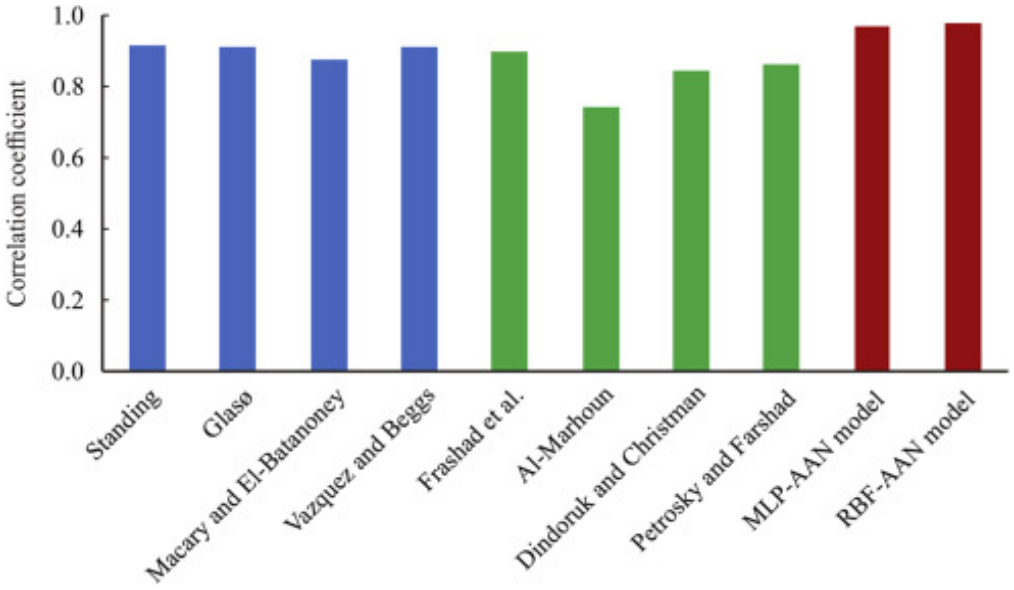

Fig 4: Performance of all models for correlation coefficients

The comparison between the values of the solution gas-oil ratio predicted by the developed models and several empirical correlations showed that the developed MLP and RBF models performed all of the studied empirical correlations providing a strong agreement between the experimental and predicted values. And the RBF model showed higher efficiency and accuracy compared to the proposed MLP model. Overall, the mathematical strategy proved that the developed MLP and RBF models are statistically valid and acceptable. The performance of the proposed MLP and RBF models confirmed their indication towards the fact that these models

can be fully utilized in the absence of experimental approaches for the accurate calculation of solution gas-oil ratio.

PROPOSED METHODOLOGY

- **Workflow**

Coming to the workflow, we started up by importing all the necessary libraries that would be used for classifying the classes:

1. **Numpy**: Since machine learning models are primarily concerned with data, a numpy library would be useful for manipulating and changing data. Numpy was chosen because it is exponentially fast and has a more comprehensive range of data manipulation capabilities. NumPy strives to have a 50-fold faster array object than standard Python collections. The array object in NumPy is called ndarray, and it comes with a slew of helper functions to make dealing with it an ease.
2. **Matplotlib.pyplot**: Matplotlib is a fantastic Python simulation library for 2D array plots. Matplotlib is a multi-platform data analysis library based on NumPy arrays and intended to work with the SciPy stack as a whole. Matplotlib has a large number of plots. Plots aid in the understanding of movements, patterns, and associations.They're usually used to make decisions based on numerical data. The Matplotlib.pyplot library is used to create graphs and charts, and we used it to create our plots because of its built-in functions and databases.
3. **Sklearn.svm**: In comparison to other classifiers such as logistic regression and decision trees, SVM has a very high accuracy. It is well-known for its nonlinear input space kernel trick. The classifier uses the hyperplane with the most margin to distinguish data points. An SVM classifier is also known as a discriminative classifier because of this. SVM determines the best hyperplane for classifying new data points.
4. **Scipy.stats**: SciPy (pronounced "Sigh Pie") is an open-source math, science, and engineering programming language. NumPy, which provides convenient and efficient N-dimensional array manipulation, is used by the SciPy library. The SciPy library is designed to deal with NumPy arrays, and it includes a number of user-friendly and effective numerical routines, such as numerical integration and optimization routines.This

module includes an increasing library of statistical functions as well as a vast number of probability distributions.

Once all the libraries had been imported our next step was to get the data, so to get the data we generated random values so as to clearly identify different classes since if we would have used the dataset, than all the classes would have been classified but our main aim of classifying novels would have been vanished because all the classes in a particular dataset are defined in such a way that they mostly belong to one set of particular class. So what we basically did was we defined 10% of the total data to belong to novel class. So if we take the data of 200 samples then there would be around 20 novel data present in our dataset.

Once done with that, we defined our classifier which would eventually help us in building and training our model using svm.oneclassSVM where we defined the parameters to be 'nu' (which is the upper bound on the fraction of the training errors and a lower bound of the fraction of support vectors) is set as 0.95*total_number_of_outlier_fraction + 0.05, then we defined 'kernel' of the classifier to be as RBF which is Radial Basis Function. Now we use the radial basis function because the feed-forward architecture of the RBF network is made up of a single hidden layer of locally tuned units that are entirely interconnected to an output layer. The input layer broadcasts the input vector's coordinates to each of the secret layer's units. After that, each unit in the hidden layer generates an activation depending on the RBF. Finally, the unit in the output layer computes a linear combination of the activations of hidden units.

After defining the classifier, we defined the training data, testing data and novel data variables and then we defined the variable 'ground_truth' where non-novel data values are represented by 1 and novel data values are represented by -1 which is again a necessary step because it will eventually be this variable that will differentiate novel classes from the normal classes. Starting with the loop we first set our data along with its output using the function 'np.r_' that takes the parameters as the upper bound, lower bound and size at which novel will be defined so as to clearly differentiate them.

Then we went on building the inner loop where we are actually differentiating the data using the classification.decision_function concatenating it with the result using numpy.c_ where this function is taking a contiguous flattened array of input data and the predicted data as input. At the last we were left with nothing but to just plot all the predicted values and for that we used

matplotlib library with all the parameters and henceforth we have our output where we could clearly distinguish between classes and novels. And that is how we were able to differentiate novel classes from the normal classes.

- **Technology**

Support Vector Machine (SVM) : "Support Vector Machine" (SVM) is a supervised machine learning algorithm which is mostly used for classification problems.In SVM algorithm, we represent each data item being a point in n-dimensional space (where n is number of features vectors you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well .So when new testing data is added, whatever side of the hyperplane it lands will decide the class that we assign to it.

Support vectors are the data points nearest to the hyperplane, the points of a data set that, if removed, would alter the position of the dividing hyperplane. Because of this, they can be considered the critical elements of a data set.

The distance between the hyperplane and the nearest data point from either set is known as the margin. The goal is to choose a hyperplane with the greatest possible margin between the hyperplane and any point within the training set, giving a greater chance of new data being classified correctly.In the SVM classifier, it is easy to have a linear hyper-plane between these two classes,the SVM algorithm has a technique called the kernel trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts non separable problem to separable problem. It is mostly useful in non-linear separation problems.For closed ,non linear decision boundary we will be using the rbf kernel in one class svm explained below:

Radial basis function kernel (RBF) / Gaussian Kernel:

Gaussian RBF(Radial Basis Function) is one of the popular Kernel methods of SVM. RBF kernel is a function whose value depends on the distance from the origin or from some point. Gaussian Kernel is of the following format:

$$K(X_1, X_2) = exponent(-\gamma||X_1 - X_2||^2)$$

Where ||X1 — X2 || = Euclidean distance between X1 & X2

Using the distance in the original space we calculate the dot product (similarity) of X1 & X2 where similarity is the angular distance between two points.

Parameters: γ -> Gamma

Behavior: As the value of ' γ ' increases the model gets overfits and as the value of ' γ ' decreases the model underfits.

RBF performs classification by measuring the input's similarity to the examples or the vectors from the training set. When we want to classify a new input, every neuron computes the Euclidean distance between the input and its prototype. Now, if the input more closely resembles the class A prototypes than the class B prototypes, it is classified as class A.

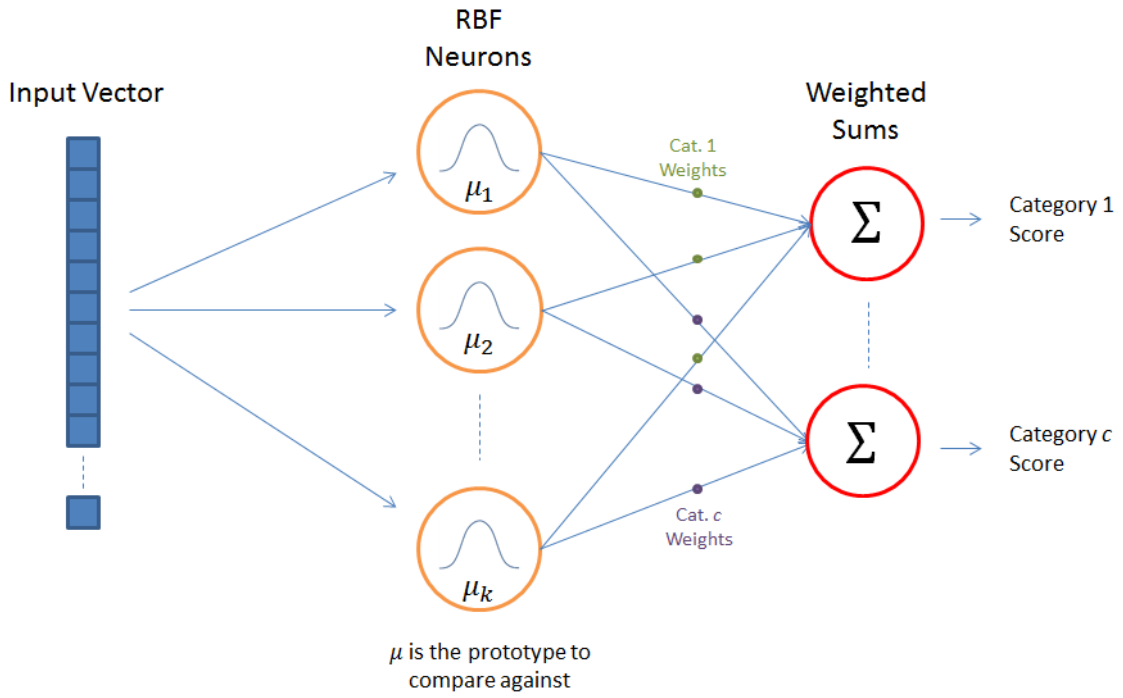

Fig 5: RBFNN Structure

Each neuron of RBF stores a "prototype", which is one of the vectors from the training set. Each neuron compares the input vector to its prototype and gives a value between 0 and 1 as an output, which is a measure of similarity. If the input and the prototype are equal, then the output of that RBF neuron will be 1. As the distance between the input and prototype increases, the output falls off exponentially towards 0. This RBF neuron activation function has a formula-

$$\varphi(x) = e^{-\gamma\|x-\mu\|^2}$$

Where x is the input, μ is the prototype vector, and γ is $\frac{1}{2\sigma^2}$ (σ is the standard deviation). For smaller values, the decision boundary is smooth but as the value of γ increases the mapping becomes less smooth, or more curves are observed which increases the accuracy of the model we want as it can quite precisely classify different classes and detect novel classes as well.

As shown in the figure below, we can see how the γ in the RBF kernel affects the decision boundary.

Now, since Radial Basis Function majorly creates a boundary around each class in such a way that it is easy to identify all the classes separately. However, in the case of Multi-Layer Perceptron, we just create a decision boundary where all the classes are segregated in a way that they could be distinguished but at the same time, all other novelties or anomalies are also taken into account. But that is not the case when it comes to Radial Basis Function, all other novelties or anomalies can be briefly distinguished from all other classes since it creates an enclosed boundary outside all other classes. In this way, we can identify what is the novelty in the data stream.

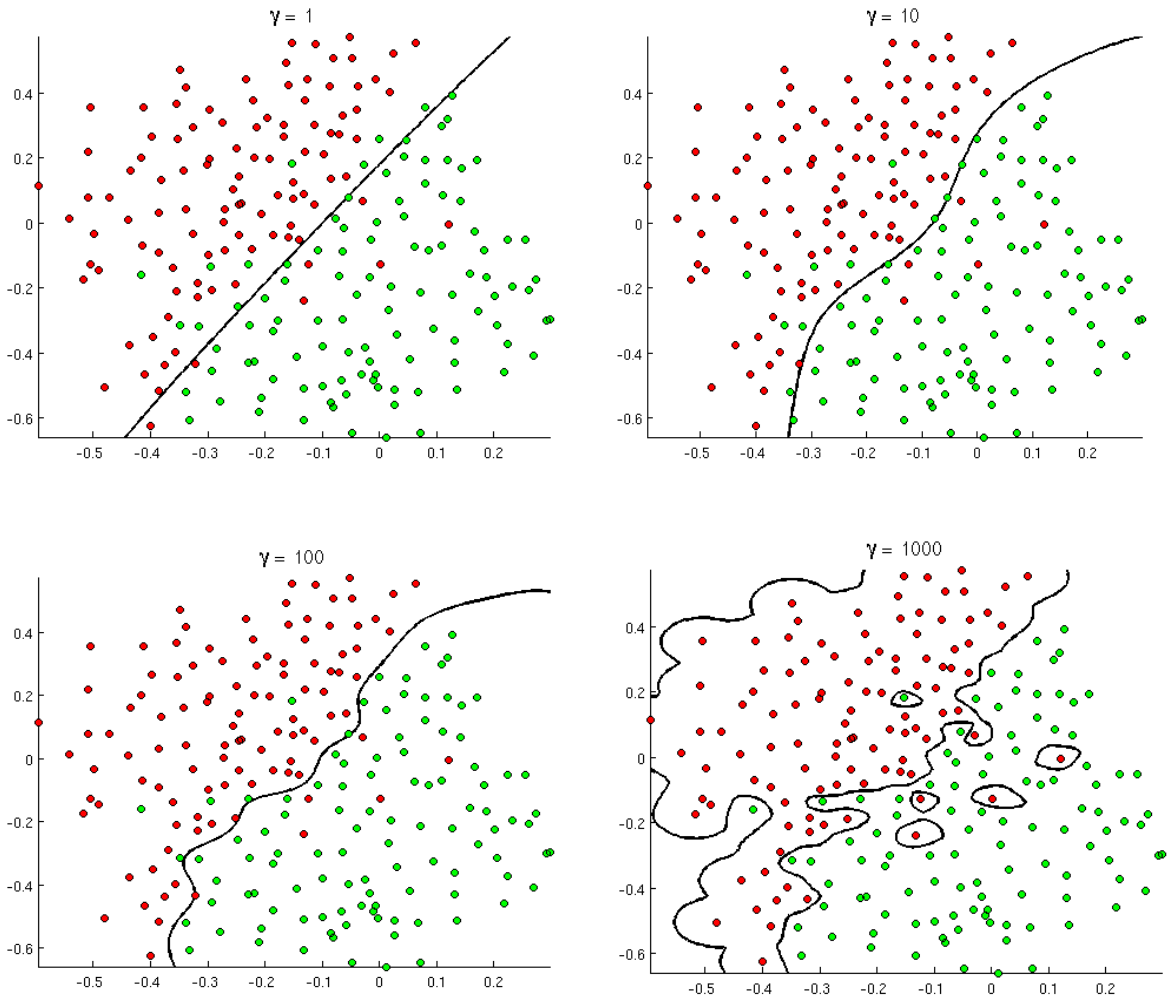

Fig 6: Change in decision boundary with respect to change in Gamma Value of RBF parameter

One Class SVM : One-class classification (OCC), also known as unary classification or class-modeling, attempts to classify objects with a given class among all objects by learning exclusively from a training set comprising only such objects. To classify suspicious findings, an OC-SVM calculates a distribution that covers the majority of the observations, and then identifies those that deviate from it by a fitting criterion as "suspicious." An OC-SVM solution is created by calculating a probability distribution function that makes the majority of the observed data more likely than the others, as well as a judgement rule that divides these observations by the widest margin possible.Since teaching an OC-SVM requires a quadratic programming problem, the learning process has a high computational complexity.

RESULT AND ANALYSIS

One class SVM has shown a very promising performance for this random data.From the results, we can observe that when there exist clear cluster structures in the dataset one-class SVM is able to capture the cluster structured with Gaussian kernels and detect novel classes efficiently,these gaussian rbf actually gives these non-linear and closed boundaries clusters .Cluster separation is measure which ensures proper separation of clusters , as this value increase better clusters are formed leading to more errors as now proper clustering of non-novel class items is done and the novel classes are identified better.We can see that when the cluster separation value is increased from 4(fig 7(a)) to 5 (fig 7(b)) the number of error increases.

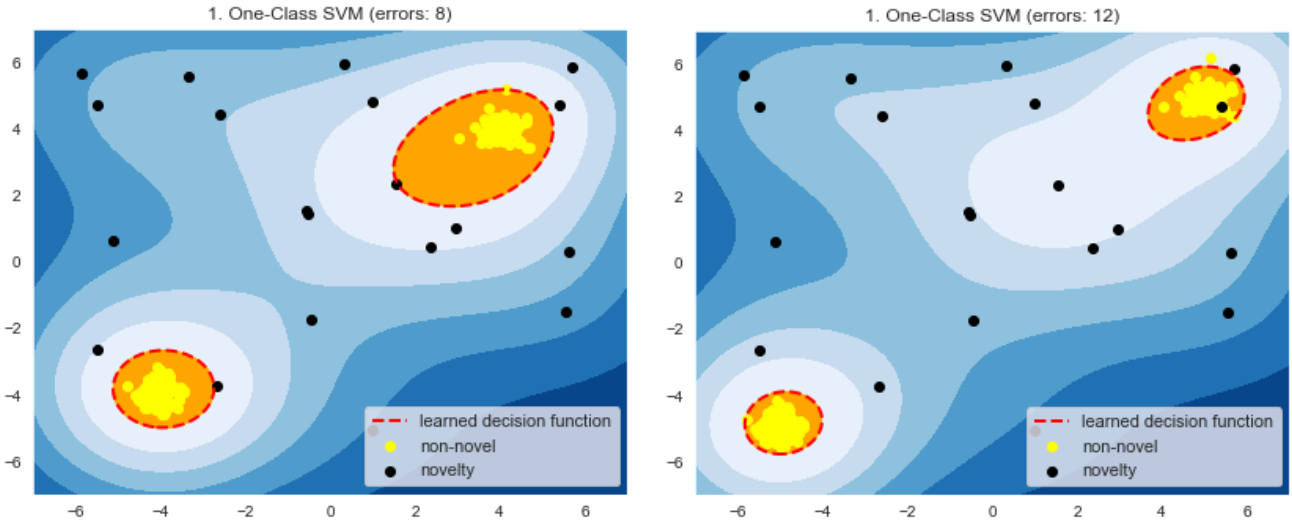

Fig 7(a) and Fig7(b) :Novelty Detection

CONCLUSION AND FUTURE SCOPE

Novelty detection, recognising that certain elements of a dataset do not fit into the underlying model of the data, is an important problem for learning systems – if data used as input to the system does not correspond with the data in the training set, then the performance of the system will suffer.

To sum up, RBF networks are much easier to be designed and trained than neural networks. From the point of generalization, RBF networks can respond well to patterns which are not used for training. RBF networks have strong tolerance to input noise, which enhances the stability of the designed systems. Therefore, it is reasonable to consider RBF for novelty detection .Rbf

functions as mentioned above provide closed,curved boundaries. But there are some shortcomings to it too:

1. Modeling normal objects and novel classes efficiently: Novelty detection depends highly on the modeling of the normal(non-novel) objects and the novelties. Often, building a comprehensive model for data normality is very challenging, if not impossible. This is because it's difficult to analyze all possible normal behaviors in an application.

2. Handling noise in Novelty Detection: Sometimes noise is unavoidable in the real data sets. It might be the missing values in the data set or deviations in the values. This can distort or deform the structural orientation of the data making it hard to differentiate the normal objects and the novelties which therefore can decrease the effectiveness of Novelty Detection systems.

3. More than 1 non-novel class :Even if we take datasets with more than 2 feature vectors they both would be considered as one non-novel class and classification would be done between the one non-novel class and the novel class.Hence we will be able to tell whether the class datapoint is novel class point or not,no further classification among non-novel class points is possible.

Novelty detection is extensively used in a wide variety of applications such as military surveillance for enemy activities to prevent attacks, intrusion detection in cyber security, fraud detection for credit cards, insurance or health care and fault detection in safety critical systems and in various kinds of images.

REFERENCES

[1] Radial Basis Function Network (RBFN) Tutorial, 15 Aug 2013, [Online], Available: https://mccormickml.com/2013/08/15/radial-basis-function-network-rbfn-tutorial/

[2] ArefHashemi, Farshid Madanifar, Et al. "Implementation of multilayer perceptron (MLP) and radial basis function (RBF) neural networks to predict solution gas-oil ratio of crude oil systems," *Petroleum*, volume 6, issue 1, March 2020, Pages 80-91

[3] Yuhua Li , Michael J. Pont ,Et al. "Applying MLP and RBF classifiers in embedded condition monitoring and fault diagnosis" , Dec 2001 , Pages 315-343

[4] Support Vector Machines (SVM), Ajay Yadav, 20 Oct, 2018, [Online], Available: https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

SOURCE CODE

Importing necessary libraries:

```
import numpy as np
from scipy import stats
import matplotlib.pyplot as plt
import matplotlib.font_manager
from sklearn import svm
```

Dataset settings:

```
rng = np.random.RandomState(42)

n_samples = 200
novel_frac = 0.1
clusters_separation = [4,5]
```

Define the novelty detection tool:

```
classifier = {
    "One-Class SVM": svm.OneClassSVM(nu=0.95 * novel_frac + 0.05,
                                     kernel="rbf", gamma=0.1)}


xx, yy = np.meshgrid(np.linspace(-7, 7, 500), np.linspace(-7, 7, 500))
non_novel = int((1. - novel_frac) * n_samples)
novel = int(novel_frac * n_samples)
ground_truth = np.ones(n_samples, dtype=int)
ground_truth[-novel:] = -1
```

Fit the problem with varying cluster separation:

```
for i, offset in enumerate(clusters_separation):
```

```
np.random.seed(42)

# Data generation
X1 = 0.3 * np.random.randn(non_novel // 2, 2) - offset
X2 = 0.3 * np.random.randn(non_novel // 2, 2) + offset
X = np.r_[X1, X2]
# Add novelty
X = np.r_[X, np.random.uniform(low=-6, high=6, size=(novel, 2))]

# Fit the model
plt.figure(figsize=(12, 5))
for i, (clf_name, clf) in enumerate(classifier.items()):
    # fit the data and tag novelty
    clf.fit(X)
    scores_pred = clf.decision_function(X)
    threshold = stats.scoreatpercentile(scores_pred,
                                        100 * novel_frac)
    y_pred = clf.predict(X)
    n_errors = (y_pred != ground_truth).sum()
    # plot the levels lines and the points
    Z = clf.decision_function(np.c_[xx.ravel(), yy.ravel()])
    Z = Z.reshape(xx.shape)
    subplot = plt.subplot(1, 2, i + 1)
    subplot.contourf(xx, yy, Z, levels=np.linspace(Z.min(), threshold, 7),
                     cmap=plt.cm.Blues_r)
    a = subplot.contour(xx, yy, Z, levels=[threshold],
                        linewidths=2, colors='red')
    subplot.contourf(xx, yy, Z, levels=[threshold, Z.max()],
                     colors='orange')
    b = subplot.scatter(X[:-novel, 0], X[:-novel, 1], c='yellow')
    c = subplot.scatter(X[-novel:, 0], X[-novel:, 1], c='black')
    subplot.axis('tight')
```

```
        subplot.legend(
            [a.collections[0], b, c],
            ['learned decision function', 'non-novel', 'novelty'],
            prop=matplotlib.font_manager.FontProperties(size=11),
            loc='lower right')
        subplot.set_title("%d. %s (errors: %d)" % (i + 1, clf_name, n_errors))
        subplot.set_xlim((-7, 7))
        subplot.set_ylim((-7, 7))
    plt.subplots_adjust(0.04, 0.1, 0.96, 0.92, 0.1, 0.26)
plt.show()
```