

# Particle Swarm Optimisation for Outlier Detection

Ammar W Moheemmed, Mengjie Zhang, Will Browne  
School of Engineering and Computer Science, Victoria University of Wellington  
PO Box 600, Wellington, New Zealand

Technical Report: ECSTR10-07

## ABSTRACT

Outlier detection is an important problem as the underlying data points often contain crucial information, but identifying such points has multiple difficulties, e.g. noisy data, imprecise boundaries and lack of training examples. In this novel approach, the outlier detection problem is converted into an optimisation problem. A Particle Swarm Optimisation (PSO) based approach to outlier detection can then be applied, which expands the scope of PSO and enables new insights into outlier detection. Namely, PSO is used to automatically optimise the key distance measures instead of manually setting the distance parameters via trial and error, which is inefficient and often ineffective. The novel PSO approach is examined and compared with a commonly used detection method, Local Outlier Factor (LOF), on five real data sets. The results show that the new PSO method significantly outperforms the LOF methods for correctly detecting the outliers on the majority of the datasets and that the new PSO method is more efficient than the LOF method on the datasets tested.

## Categories and Subject Descriptors

I.5 [Pattern Recognition]; I.2 [Artificial Intelligence]

## General Terms

Algorithms, Design

## Keywords

Outlier detection, particle swarm optimisation

## 1. INTRODUCTION

An exact definition of an outlier often depends on the context of the application domain and the used detection method. Yet, Hawkins' definition is regarded general enough to cope with various applications and methods: *An outlier is an observation that deviates so much from other observations (considered normal) as to arouse suspicion that it was*

*generated by a different mechanism* [7]. The importance of outlier detection is due to the fact that outliers in data reveal hidden information that sometimes need swift action to avoid future wider harm or damage. For example, in finance, detecting outliers such as credit card frauds can trigger action to prevent more money loss to the customers and the banks. In network security, detecting suspicious intrusion behaviours as early as possible will prevent greater damage to the network and its components. Early fault diagnosis in machines (motors, generators and space shuttles, etc.) might spare human lives and avoid catastrophic damage in addition to many other applications.

The outlier detection problem, in its most general form, is difficult to solve due to a number of challenges [5]. The boundary between normal and outliers behaviour is often not precise. In many problem domains, normal behaviours keep evolving and the exact notion of an outlier varies with different tasks. It is often difficult to obtain sufficient outliers data for training and/or validation. The data often contain noise that tends to make normal observations appear similar to the actual outliers and vice versa.

Numerous techniques have been proposed to detect outliers for different applications. These techniques can be categorised into several approaches [5, 9, 3]: *statistical approach*, *clustering based approach* and *distance based approach*. In the distance based approach, a simple distance or similarity measure between every two instances/points in the data set is calculated and the points whose distances are longer than a particular radius (threshold) are considered outliers. Compared with the clustering based approach, the distance based approach is much more simple to use. Unlike the statistical based methods, the distance based methods make no prior assumptions about the data distribution model and are more suitable for multi-dimensional data sets. Due to these advantages, the distance based approach is widely used in outlier detection.

In the distance based approach, there are two fundamental methods on which many later techniques are based on [19, 20, 2, 17]. The first one builds upon the work of Knorr et al.[11]. In this work, a data point is defined as a Distance-Based outlier  $DB(\beta, r)$  if at least a fraction  $1 - \beta$  of the instances in the data set are further than  $r$  from it, where  $\beta$  is specified by the user based on the actual situation and  $r$  is the distance radius acting as an outlier threshold. While  $\beta$  is relatively easy to specify as outliers usually have a small neighbourhood, the value of  $r$  is usually very difficult to estimate and typically requires trial and error via hand-crafting of empirical search [19].

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

VUW ECSTR10-07 2010

Copyright 2010 VUW-ECS.

The second fundamental method develops the work from Breunig et al. [4]. This work uses local outlier factors (LOFs) rather than global distances [11]. A data point is given an outlier-ness score based on its relative density with respect to the nearest neighbour points. This method can detect outliers in data sets that have regions of varying densities, which can not be easily handled by the Knorr’s algorithm [11]. However, this method requires specifying the number of neighbourhood points (*MinPtn*) *a priori*, which typically needs hand-crafting and trial and error.

Another potential disadvantage for the distance based methods is the computational cost. For relatively small data sets this is not a problem. However, for larger data sets these methods typically require a large computational effort since the calculation of the distances between a large number of data instances/points is costly [9].

In the typical distance based outlier detection methods, the main task is to find good values of the important parameters, such as  $\beta$ ,  $r$  and *MinPtn* described earlier. This task can be naturally translated into an optimisation problem, which can be solved by some evolutionary computation paradigms, such as genetic algorithms and particle swarm optimisation. In recent years, there has been only a small amount of work applying evolutionary computation techniques to the outlier detection process, but these were mainly used for dimension reduction and feature selection [15, 1, 21].

This paper aims to convert the outlier detection problem to an optimisation problem and develop a particle swarm optimisation (PSO) approach for outlier detection using the distance based measures. Instead of using a manual hand-crafting, trial and error process, this approach will automatically evolve good values for the important parameters. To investigate this approach it will be examined and compared with a common distance based outlier detection method local outlier factor (LOF) on a sequence of outlier detection problems. Specifically, we will investigate:

- How the particles in the population can be encoded for the outlier detection task;
- How the individual particles are evaluated during the evolutionary process;
- Whether this approach outperforms the LOF method on a sequence of outlier detection tasks; and
- Whether this approach is more efficient than the LOF method particularly for large data sets.

## 2. BACKGROUND

### 2.1 The Outlier Detection Problem

Outlier detection does not have an agreed definition so far. In this paper, we use Hawkins’ definition, which is sufficiently general to cope with various applications and methods [7], as described in the introduction.

The outlier detection problem is similar, but different from binary classification. In binary classification, a classifier is trained on sufficient number of negative and positive examples of a data set to capture its characteristic and then used to classify unseen objects. The performance of a classifier is usually evaluated by its classification accuracy, error rate of the entire data set, or true positive fraction and false positive fraction for a particular class. In outlier detection, however, the task is a bit more fuzzy. The purpose is to identify those objects that are *different* from the rest of the

data that are considered normal or usual. However, it is not clear, how and to what extent the objects should be different to consider them as outliers. A common practice to detect outliers is to give an outlying score to the data points, then sort them to isolate the potential outliers on the top of the list. This is different from the typical binary classification tasks, where one data instance is considered either correct or incorrect toward a particular class.

However, due to the similarity between outlier detection and binary classification, many “benchmark outlier detection datasets” are built on some binary classification tasks with highly unbalanced data for the two classes as the test bed [8]. This paper will also use some of these data sets, which will be described in section 4.

### 2.2 Related Work

Various techniques have been proposed for outlier detections. Initially, these techniques were dominated by statistical methods [7, 3]. These methods assume a specific distribution model *a priori*, and are mostly uni-variant, i.e. they examine a single attribute to expose an outlier. These methods are not suitable for multivariate applications [11]. Clustering methods for outlier detections are based on the idea that the majority of the normal data belongs to one or more clusters while outliers either constitute a very small cluster or are far from the main formed clusters. However, clustering based approaches often find outliers as byproduct of the main process, so most of them are not optimised for detecting outliers [4].

The distance based approach proposed by Knorr [11] is widely referenced in outlier detection literature. Although it is based on a simple concept, it does not assume a specific distribution model and also can handle high dimensional data. Given a data set, a point is considered an outlier if a fraction  $1 - \beta$  of the data points is further than  $r$  (distance threshold) from that point. The two parameters  $r$  and  $\beta$  are specified by the user. Specifying  $r$  is not a straightforward, so trial and error is involved initially to determine a suitable value. Another problem with this approach is that it does not provide a means to rank the outliers, as it either categorises the point as an outlier or not.

Ramaswamy et al. propose an algorithm [19] that overcomes the requirement to specify  $r$  by the user above. The data points are given an outlying score based on the distance to the *MinPtn* - *th* nearest points. This algorithm first partitions the input dataset into disjoint subsets using clustering, then prunes those partitions determined on whether they contain outliers, leaving candidate partitions that might contain outliers. This step is to speed up the computation, especially for very large data sets, because many points will be eliminated so there is no need to find the *MinPtn* - *th* nearest points for those eliminated data points. Finally, outliers are computed from among the points in the candidate partitions. The DB based methods [11] might miss detecting outliers in datasets consisting of different density regions since they consider the data points globally.

Breunig et al. [4] propose a local based detection algorithm. The data points are scored using the Local Outlier Factor (LOF) that represents the degree of the outlying of the points depending on their local neighbourhood. For any given data point, the LOF score is equal to the ratio of average local density of the *MinPtn* nearest neighbours of the point and the local density of the data point itself. The lo-

cal density of a point, which is called *reachability density*, is defined in term of a reachability distance. The reachability distance is computed from the distance to the *MinPtn*-th nearest neighbourhood. For a normal data point lying in a dense region, its local reachability density will be similar to that of its neighbours and will have a low LOF, while for an outlier, its local density will be lower than that of its nearest neighbours, hence will get a higher LOF score. However, selecting *MinPtn* is non-trivial and the computation cost is directly related to the value of *MinPtn*.

The idea of LOF has been extended and improved in different ways. For example, the Connectivity-based Outlier Factor (COF) scheme [20] extends the LOF algorithm for detecting outliers in data patterns that are difficult to discover using LOF. The LSC-Mine [2] simplifies the computation of the local reachability density accompanied with pruning leading to a faster computation.

A similar technique called LOCI (Local Correlation Integral) is presented in [17]. LOCI addresses the difficulty of choosing *MinPtn* in the LOF technique by using a different definition for the neighbourhood. For each data point, the neighbourhood within different values of  $r$  is examined. A point is flagged as an outlier if a parameter called Multi-granularity deviation factor (MDEF) deviates three times from the standard deviation of MDEFs in a neighbourhood. The MDEF at radius  $r$  for a point is the relative deviation of its local neighbourhood density from the average local neighbourhood density in its  $r$  neighbourhood. Thus, an object whose neighbourhood density matches the average local neighbourhood density will have an MDEF of 0. In contrast, outliers will have MDEFs far from 0. However, the running cost of the algorithm is high due to the need to compute statistical values including the standard deviation.

Recently, Zhang et al. [22] proposed Local Distance-based Outlier Factor (LDOF) to measure the outlier-ness of objects in scattered datasets. LDOF uses the relative location of an object to its neighbours to determine the degree to which the object deviates from its neighbourhood. Kriegel et al. [12] proposes to use the variance in the angles between the difference vectors of a point to the other points to identify outliers. The intuition behind this method is that, for points within a cluster, the angles between difference vectors to pairs of other points differ widely, in contrast to outliers that will have small variance in the angles. To reduce the pairwise distance evaluations in distance and density based techniques, the method proposed by Yaling [18] defines a fixed set of reference points for ranking the data points and detecting the outliers.

Evolutionary algorithms, such as genetic algorithms (GAs), have been used in outliers detection. Kelly et al. [6] used GAs to search regression data for a subset of points that have the highest fitness/outlier-ness degree. The fitness function is a diagnostic test for outliers.

Example-based methods allow users to provide some outlier examples to the technique in order to find more objects that have similar outlier-ness characteristics to these examples. Yuan et al. [15] use GAs to find a low dimensional subspace where given user examples are isolated more significantly than in other subspaces, and then detecting  $DB(\beta, r)$  outliers in this subspace. The disadvantage of this algorithm is that there is still a need for the user to intervene to select the parameters for deciding the outliers.

Aggrawal et al. [1] defines a sparsity coefficient using GA

to find low dimensional cubes with the lowest data sparsity. The same algorithm is implemented by Dongyi et al. [21], but using PSO. The disadvantage of this algorithm is the assumption that the data points are uniformly distributed and that the sparsity factor is based on a normal distribution.

### 2.3 Particle Swarm Optimization

Particle Swarm Optimization (PSO) is a population based stochastic optimization tool inspired by social behaviour of flocks of birds (and schools of fish, etc), as developed by Kennedy and Eberhart in 1995 [10]. PSO consists of a population of particles that search for a solution within the search domain. The search for optimal position (solution) is performed by updating the velocity,  $V_i$ , and the position,  $X_i$ , of particle  $i$  according to the following two equations:

$$V_i = V_i + \phi_1 c_1 (X_i^{best} - X_i) + \phi_2 c_2 (X_i^{gbest} - X_i) \quad (1)$$

$$X_i = X_i + V_i \quad (2)$$

where  $\phi_1$  and  $\phi_2$  are positive constants, called *acceleration coefficients*,  $c_1$  and  $c_2$  are two independently generated random numbers in the range  $[0, 1]$ ,  $X_i^{best}$  is the best position of the  $i$ -th particle, and  $X_i^{gbest}$  is the best position found by the neighbourhood of the particle  $i$ . The swarm starts by initializing the particles' velocities and positions randomly with values constrained by the search domain. Then the swarm goes through a loop, during which the positions and velocities are updated according to the above equations. When the termination criterion is satisfied, the best particle (with its position) found so far is taken as the solution to the problem. To prevent the velocity from exploding, it is climbed to  $V^{max}$ .

Alternatively, Clerc et al. [16] proposed to add a constriction factor to prevent the velocities of overstepping the search domain limits. Thus, the velocity update equation is modified as follows:

$$V_i = \chi \times (V_i + \phi_1 c_1 (X_i^{best} - X_i) + \phi_2 c_2 (X_i^{gbest} - X_i)) \quad (3)$$

where  $\chi$  is the constriction factor and it is usually set to 0.729.

The neighborhood topology of the swarm determines how the particles are connected to each other and it effects how the information is exchanged among the particles. In the Ring topology, each particle is connected to two other neighbours to form a ring. In the Global topology, each particle is connected to all other particles. The advantage of the Ring topology over the Global topology is that the transfer of information is slow among the particles, which helps avoid the swarm of falling into a local optimal. Therefore, the ring topology is used in this paper.

### 3. A NEW PSO BASED APPROACH

Our PSO based approach uses the distance based measure. To illustrate which data points can be considered an outlier during PSO evolution, we use an example data set shown in Fig.1. In this example, the data point  $o$  is more likely to be an outlier because it is different and far from other points. An apparent difference lies in the number of nearby points that are within a certain radius/distance  $r$  from it. Assuming a circle of radius  $r$  centred at the data point, then  $o$  has the minimum  $\frac{k}{r}$  ratio, where  $k$  is the number of points enclosed by the circle of radius  $r$ . In other words, the ratio  $\frac{k}{r}$  can be used as a measure to detect outliers.

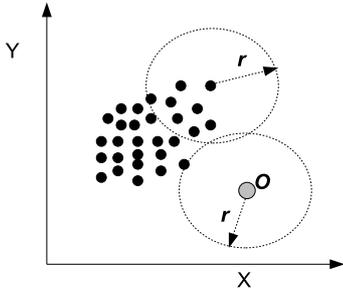


Figure 1: An examples dataset with ourliers.

Note that this idea is related to Knorr’s definition [11] of an outlier: a data point is an outlier if it has a fraction  $\beta$  or less of the total points within distance  $r$ . In fact,  $\beta$  in Knorr’s method is a function of  $k$  mentioned above, which varies with the value of  $r$ . The internal relationship makes manually setting good values for the two parameters even more difficult. Accordingly, instead of the user deciding the value of  $\beta$  and  $r$ , we propose an algorithm using PSO to automatically find suitable values from the data.

In this approach, PSO is used to find the point that has the minimum  $\frac{k}{r}$  ratio. The value of  $r$  that results in the minimum  $\frac{k}{r}$  is used to compute the ratio for the other points and rank them accordingly to identify the top outliers. For the simple example shown in Fig.1, a value of  $r$  can be set to the distance of the nearest neighbourhood to  $o$ , as this would be sufficient to score the data points and to put  $o$  to the top detected outlier. Unfortunately, most of the real world data sets include much more complex distributions that make finding  $r$  a very difficult task.

Intuitively, a PSO algorithm can be implemented to find the single parameter  $r$ . In this case, a particle can just encode one parameter,  $r$ . To evaluate the goodness of a particle, the ratio  $\frac{k}{r}$  is computed for each data point. Then the minimum value of the ratio  $\frac{k}{r}$  among the data points can be used as the fitness function. The data point with the best value of  $r$  that leads to the minimum value of the ratio  $\frac{k}{r}$  can be detected as a top outlier. However, computing the ratio  $\frac{k}{r}$  for all the data points every time it was needed to evaluate all the particles in the population could be very computationally inefficient. A better encoding scheme and corresponding fitness function had to be developed.

### 3.1 Particle Encoding

To get a more efficient encoding scheme, we consider a different way to exploit further the search capabilities of PSO. In addition to the radius parameter  $r$ , the particle’s encoding is now extended to include the index of the data point. So the particle encodes the tuple  $ID, r$ . The parameter  $ID$  is the index of the data points. It is an integer value, so the values returned by the particles are rounded up to the nearest integer value.  $r$  is the radius of the hyper-sphere centred on the data point  $ID$ .

The objective of the search here is to find the data point that has the minimum  $\frac{k}{r}$  and simultaneously to find the value of  $r$  that results in minimizing this ratio. By including the index of the point as another search variable in addition to  $r$ , it is not needed to compute the ratio  $\frac{k}{r}$  for all the points whenever a particle is evaluated. We expect PSO to automatically find a smaller group of “optimised” data points that have better potential as outliers.

## 3.2 The Fitness Function

### 3.2.1 Design Considerations

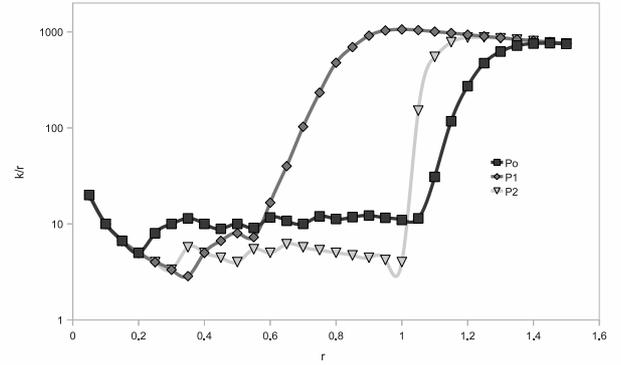


Figure 2: Change of  $\frac{k}{r}$  with  $r$  for 3 points from the yeast dataset, P0 is from the minority class, P1 and P2 from the majority class.

When designing the fitness function, there are a number of issues to be addressed. We analyse them through an example. Fig.2 shows the change in the ratio  $\frac{k}{r}$  for three points from the Yeast data set [14]. The original Yeast data set contains nine classes. Among them, the first three classes have 1136 examples in total, and the last class consists of only five data instances. To be used for outlier detection, we built a new dataset based on the original set. The new dataset uses the 1136 data points in the first three classes as the *normal* group, and the five data point in the last class as outliers that we wish to detect. The plot of the ratio  $\frac{k}{r}$  with respect to  $r$  for one of these considered outliers (P0) is shown in Fig.2. The other two points (P1 and P2) belong to the normal group.

The global minimum value of  $\frac{k}{r}$  is at  $r$  equals 0.35. However, this value is due to P1, which belongs to one of the normal group. Therefore, PSO will return P1 (and  $r = 0.3$ ) as an outlier point and it will be in a higher rank than P0, which is clearly not what we wanted. The minimum value of  $r$  for data point P0 is at 0.2, but P1 and P2 also have the same value at  $r = 0.2$  for , which does not assist in ranking P0 higher than P1 and P2. Importantly, too small a value of  $r$  is not good for detecting outliers even though the value of  $\frac{k}{r}$  is the smallest or very small. On the other hand, the value of  $r$  should not be so big to include the all points of the data set. In this case, the ratio  $\frac{k}{r}$  will be same for all the data points, which will not provide any help for ranking the data points either. Thus,  $r$  should have a lower bound and also an upper bound in order to identify the correct outliers. Accordingly, when designing the fitness function, we should consider this issue in addition to the key measure  $\frac{k}{r}$ .

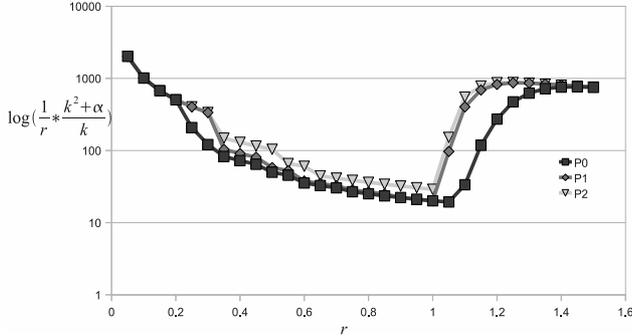
### 3.2.2 New Fitness Function

The fitness function consists of three terms as follows:

$$\frac{\alpha}{r * k} + \frac{k}{r} + \frac{k}{n - k} \quad (4)$$

where  $n$  is the size of the data set and  $\alpha$  is a constant. The first term  $\frac{\alpha}{r * k}$ , where  $\alpha$  is a constant to limit the lower bound of  $r$ . The value of  $r$  should be large enough to include some neighbouring points to be valid to compute the ratio  $\frac{\alpha}{r * k}$  and detect outliers. The second term  $\frac{k}{r}$  is the key measure

to optimize. These two terms are combined as  $\frac{1}{r} \frac{k^2 + \alpha}{k}$  and drawn with respect to  $r$  in Fig.3 for the three points of Fig.2. From this figure, it can be seen that, by adding the first term to the key measure (second term), the minimum (at  $r = 1.05$ ) is due to point P0, the true outlier. Thus, in the final ranked list P0 will be ahead of P1 and P2. The first term (and hence the constant  $\alpha$ ) will influence the minimum number of points to be included by  $r$ . By setting the value of  $\alpha$  to  $0.05 \times n$ , we will ensure that  $r$  for outliers will be large enough to include some neighbouring points.



**Figure 3:** The change of  $\frac{1}{r} \frac{k^2 + \alpha}{k}$  with  $r$  for the 3 points of Fig.2.

The third term  $\frac{k}{n-k}$  is to limit the upper bound value of  $r$ . Note that this term is effective when  $r$  is large so that it includes a large number of the data points. For example, when  $r$  is large so that  $k = n$ , the value of the term will become  $\infty$ , so the particles will be pushed away from those points which are very unlikely to indicate outliers. Hence,  $n$  should be usually much larger than  $k$  especially for the outliers points, in which case the value of the term is very small. Therefore, the first and the third terms are to limit the value of  $r$  to be within a useful range to detect outliers. Within this range,  $r$  is not too small that might lead to missing the outliers and not too large to include the too many false positive points from the normal group.

Please note that we could have added a coefficient acting as a weight factor to each of the three terms to further reflect the relative importance between the key measure of  $k/r$  and the two bounds of  $r$ . While this could potentially help improve the efficiency of the evolutionary process if good weight values can be found, searching for these coefficients will require further manual setup via trial and error. In fact, treating them as equally important (set all to 1.0 as in Equation 4) can achieve good results as PSO can automatically optimise  $r$ . Thus, we will use Equation 4 as the fitness function.

### 3.3 The Algorithm outPSO

Putting all the aspects together, the entire new PSO based algorithm for outlier detection is presented in Algorithm 1.

The algorithm starts with randomly initialising the particles with the first parameter,  $x_1$  initialised in the range  $[0, n]$ . The second parameter,  $x_2$ , is initialised in the range  $[0, maxdist]$ , where  $maxdist$  is the maximum possible distance between any two points. The maximum values of the velocities is set based on initial experimental results. The algorithms runs until a maximum number of iterations is finished.

---

#### Algorithm 1 Pseudocode for PSO based outlier detection outPSO

---

```

Set  $x_1^{min} = 0, x_1^{max} = n$  and  $x_2^{min} = 0, x_2^{max} = maxdist$ 
Set  $v_1^{min} = -10, v_1^{max} = +10$  and  $v_2^{min} = -1.0, v_2^{max} = +1.0$ 
{ $maxdist$  is the maximum distance between any two points}
for each particle do
   $x_1 = rand(x_1^{min}, x_1^{max})$  and  $x_2 = rand(x_2^{min}, x_2^{max})$ 
   $v_1 = rand(v_1^{min}, v_1^{max})$  and  $v_2 = rand(v_2^{min}, v_2^{max})$ 
end for
while iteration  $\leq$  MaxIterations do
  for each particle do
    Evaluate the particle
    1. Compute  $k$  for the data point of  $ID = x_1$ 
    2. Calculate the fitness function according to Eq.(4) using  $k$  and  $r = x_2$ 
    Update particle's best values  $x_1^{best}, x_2^{best}$ 
    Update swarm's best  $x_1^{gbest}, x_2^{gbest}$ 
  end for
  for for each particle do
    Calculate velocity and position according to Eq. (3) and Eq. (2)
  end for
end while
Using  $r = x_1^{gbest}$  compute  $\frac{k}{r}$  for all the data points
Sort the points

```

---

### 3.4 Discussion

Another important issue that outPSO faces is the fitness gradient of the search space. To motivate the particles to move toward promising regions where the best solutions are most likely to exist, the search domain should have gradient in the fitness landscape. A landscape with a flat level of fitness and including only spikes of high fitness will cause the PSO to continue to oscillate without settling down on a good solution, and thus difficult to converge.

The search space of the new outPSO algorithm is determined by the index of a point, which is an integer value, and  $r$ . Thus, outPSO is connected to the way the data are indexed. For the data sets to be tested, outPSO will not encounter a problem related to the gradient issue. In addition, it is not required to find exactly the optimal point that has the minimum  $\frac{k}{r}$ . Near optimal points will be sufficient because the points are ordered and the outliers will have the lowest  $\frac{k}{r}$  ratio even if the ordering of outliers could be improved. So we expect this algorithm to work well for outlier detection.

## 4. EXPERIMENTS

### 4.1 Dataset Selection

The evaluation of outlier detection methods poses a certain difficulty as there exists different definitions of an outlier and different domain experts often have different opinions on whether a detected case is a true outlier or not. In this paper, we use two ways to form the test datasets for the experiments. The first is to use widely accepted ‘‘benchmarks’’. The second is to use an existing unbalanced classification dataset, where some or all of the instances from the minority class are chosen as the outliers while the instances

from all or some of the instances from majority class(es) as the normal group. As there are so many conflicting “benchmarks” in this area, the second way is also widely used in the existing work [8]. However, in some data sets, some elements of the majority classes are so different that they can be detected as outliers.

Following the above two ways, we choose five datasets as the test bed: the *Hockey dataset*, the *Wisconsin Breast Cancer (Original)*, the *Wisconsin Breast Cancer (Diagnostic)*, the *Yeast dataset*, and the *Shuttle dataset* [14]

## 4.2 Experiment Configurations

The outPSO algorithm is evaluated with a population size set to 30 particles and the maximum number of iterations is 1000. Particles are connected in the Ring topology. The constriction factor  $\chi$  is set to 0.729,  $c_1$  and  $c_2$  in equation 1 are both set to 2.02. These parameter values are set based on the common setting [16] and quick empirical trial experiments. Each experiment is repeated for 100 independent runs and average results are reported.

We compare outPSO with the LOF algorithm with *MinPtn* set to 40.

The Euclidean distance measure is used for calculating the distance between any two data points/instances. The attribute values in all the data sets are normalized/scaled into [0, 1] according to the equation 5.

$$f_i = \frac{(f_i - f_i^{min})}{(f_i^{max} - f_i^{min})} \quad (5)$$

where  $f_i$  is the value of the attribute for a particular instance and  $f_i^{max}$  and  $f_i^{min}$  are the maximum and minimum values of that attribute across the entire dataset.

## 4.3 Results: The Hockey Dataset

The National Hockey League (NHL) data set has been used as a “benchmark” in several outlier detection papers [11, 18]. The statistics of NHL 2003-2004 obtained from NHL website [13] are used. The same statistics have been used in [18]. The data set contains 916 entries.

Two tests were conducted. The first test finds outliers based on the three attributes: *games played*, *goals scored* and *shooting percentage*. The results are shown in Table 1. This test is relatively easy as the outlier players are different from the other players (normal group) in terms of the median values of the three attributes. Our new outPSO method achieved the same rank of the outlier players as the LOF method [11] and also the Yaling method [18].

The second test is to detect outliers based on the three different attributes: *points scored*, *plus-minus statistic* and *penalty minutes*. The top three outliers are identical for outPSO and LOF algorithms, while the fourth outlier detected by outPSO came third in Yaling’s [18] rank.

Accordingly, the PSO approach achieved at least as good results as the LOF and Yaling methods on the two tests. In addition, PSO has the ability to automatically select important attributes from the datasets. Fig.4 visualises the two data sets with the outliers “important” (not all) attributes.

It is mentioned in section 3.4 that for PSO to locate good fitness regions, a fitness gradient is crucial. The two previous tests are carried out on the data sets without any prior ordering. We repeat the tests 100 times with the positions of the outliers changed randomly each time in order to change the fitness landscape and investigate whether outPSO is able

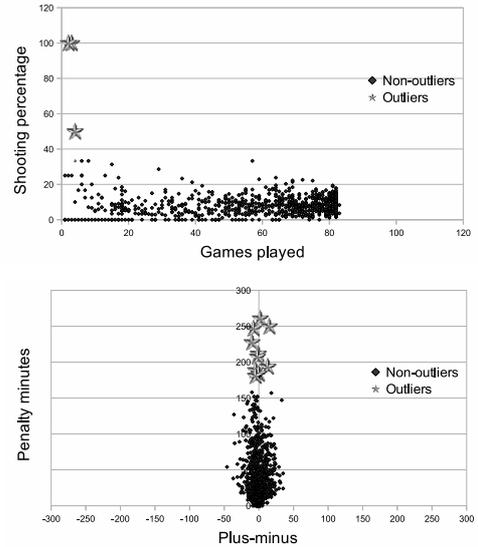


Figure 4: The outliers detected in NHL dataset.

to detect them again. For both tests, outPSO is able to find with a probability of 95% any of the three top outliers as the best point. When outPSO is run twice, after the first run the data points are ordered, outPSO is able to find any of the top outliers with probability of 100%. This is because ordering the data provides a better gradient to perform the search. In fact, even without ordering, the new PSO approach still performed well as the particle encoding scheme considered the fitness landscape.

## 4.4 Results: Wisconsin Breast Cancer Dataset (original) (WBCDO)

The WBCDO data set has 699 instances with nine attributes [14]. Each record is labeled as benign (458 or 65.5%) or malignant (241 or 34.5%). An outlier detection dataset is formed by choosing all the 458 benign records (as the normal group) and the first 10 malignant records (as outliers). The evaluation is based on the number of the malignant records occupying the top 10 positions.

The new outPSO algorithm is able to list 6 malignant records, while the LOF did not list any of the malignant records in the top 10 positions. To confirm these results, the experiment were repeated 100 times with the malignant examples selected randomly. The average number of outliers detected using outPSO was  $(5.85 \pm 1.17)$ . However, the LOF algorithm was not able to list any of the malignant examples in the 10 top positions.

But, when *MinPtn* increased from 40 to 80, the performance of LOF improved getting an average of  $(4.55 \pm 0.69)$ , although this is still statistically significantly worse than the outPSO method by a standard T/Z-test (at the 95% confidence level). However, increasing *MinPtn* is computationally expensive. In this case, LOF spent an average of 15 seconds to report the outliers, while the new outPSO method required less than 0.5 second of CPU time, which is 30 times shorter the LOF method.

## 4.5 Results: Wisconsin Breast Cancer Dataset (Diagnostic) (WBCDD)

This data set has 569 records with 30 attributes [14]. The number of instances for the *benign* class is 357, and the

**Table 1: Outlier detection on the Hockey data set, Test 1**

PSO Rank	LOF Rank [4]	Yaling Rank[18]	Player	Games played	Goals scored	Shooting percentage
1	1	1	Milan Michalek	2	1	100
2	2	2	Pat Kavanagh	3	1	100
3	3	3	Lubomir Sekeras	4	1	50
minimum				1	0	0
median				57	6.4	6.6
maximum				83	41	100

**Table 2: Outlier detection on the Hockey dataset, Test 2**

PSO Rank	LOF Rank	Yaling Rank[18]	Player	Points scored	Plus-minus	Penalty minutes
1	1	1	Sean Avery	28	2	261
2	2	2	Chris Simon	28	15	250
3	3	-	Krzysztof Oliwa	5	-8	247
4	6	3	Jody Shelley	6	-10	228
5	8	-	Donald Brashear	13	-1	212
minimum				0	-46	0
median				12	-1	26
maximum				94	35	261

number for the *malignant* class is 212. The outlier detection dataset in this case consists of the 357 instances from the *benign* class (as normal) and the first 10 instances from the *malignant* class as outliers. Of all the 100 independent experiment runs, the LOF algorithm correctly reported  $5.21 \pm 1.04$  outliers on average in the top 10 positions while outPSO reported  $5.23 \pm 0.95$  out of the 10 outliers. In this case, the new PSO algorithm is slightly better than the LOF method, but the difference is not statistically significant for the standard Z-test at the 95% confidence level.

#### 4.6 Results: The Yeast Dataset

This dataset consists of 1484 instances for nine classes [14]. Each instance has eight attributes. The ERL class has only five examples. We use this class as outliers against the first three classes (CYT, NUC, MIT) that consists of 1136 instances (normal group). The new outPSO algorithm successfully identified all the five outliers for the ERL class in the top five positions. However, these five members did not occupy any rank in the top five positions of the LOF list. Table 4.6 shows the top 16 outliers detected by outPSO and the corresponding ranking by LOF. Among the top 10 ranks of the LOF methods, only one of the five outlier (P5) was correctly detected and all other outliers (P1 – P4) were missing, which were ranked from 13–16 positions (incorrectly detected as belonging to the normal group). This suggests that the PSO algorithm outperforms the LOF method on this dataset.

To get an impression why those data points were incorrectly detected as outliers by LOF, Fig.5 shows the positions of these outliers with respect to other points (feature 5 vs feature 6, above) and (feature 1 vs feature 3, below). In these two subspaces, many of the non-outliers (P6–P16) were actually quite far from the normal points/instances and so are easy to confuse with the outliers. This is perhaps why the LOF method incorrectly detected many of them as outliers. The new PSO method, on the other hand, has the ability of automatically select important attributes/features, so has stronger ability to detect correct outliers from confusing cases.

**Table 3: The outliers detected in Yeast data set**

Points	Class	outPSO Rank	LOF Rank
P1	ERL	1	15
P2	ERL	2	16
P3	ERL	3	13
P4	ERL	4	12
P5	ERL	5	10
P6	NUC	6	14
P7	NUC	7	5
P8	MIT	8	6
P9	NUC	9	8
P10	CYT	10	9
P11	CYT	11	17
P12	CYT	12	18
P13	MIT	13	3
P14	CYT	14	4
P15	CYT	15	1
P16	CYT	16	2

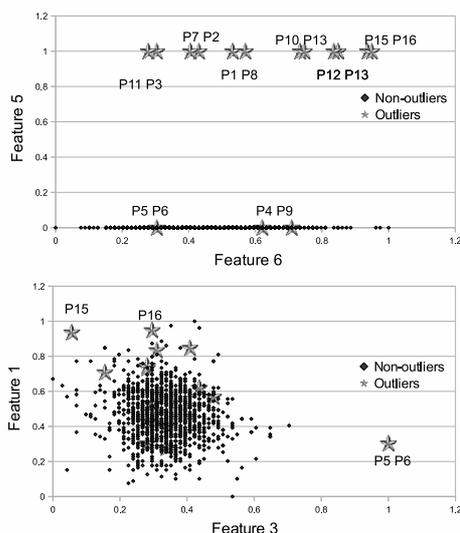
#### 4.7 The Shuttle Dataset

To compare the computational efficiency of the new outPSO algorithm and the LOF algorithm, we used a larger dataset with more examples as both algorithms can be fast on small datasets, which does not clearly distinguish the speed. The shuttle dataset was used here, with nine attributes [14]. The test set which has 14500 examples were used to perform outlier detection. Both algorithms detected the top outlier of  $ID = 11750$ .

Regarding the execution time, It took outPSO 170 seconds on average, in 100 independent experiment runs, while LOF spent 936 seconds on average to finish. Clearly, the new outPSO algorithm is much faster than the LOF on this dataset. This confirms our early hypothesis that the new PSO algorithm is more efficient as it only searches for promising regions rather than searching all neighbours.

### 5. CONCLUSIONS

The goal of this paper was to develop a new PSO based approach to outlier detection using the common distance



**Figure 5: Example false positive outliers in the Yeast dataset.**

based measures. XS This goal was successfully achieved. The results show that the new PSO based approach achieved significantly better performance than the LOF method on most of these datasets and comparable or slightly better performance on some datasets. In addition, the new PSO based method is more efficient than the common LOF method.

Compared with some distance based methods [11, 4], the new PSO method does not require users to manually specify any key distance measure parameters related to the datasets, although evolutionary parameters still need to be set.

The PSO based framework developed in this work is different from existing evolutionary solutions for outlier detection where PSO or GAs were used mainly for feature selection, and the selected features are used to detect outlier using other (non-evolutionary) methods. This new approach integrated the feature selection ability into the entire framework, which can *directly* and *automatically* detect the outliers from a particular dataset.

## 6. REFERENCES

- [1] C. C. Aggarwal and P. S. Yu. Outlier detection for high dimensional data. In *Proceedings of the 2001 ACM SIGMOD International Conference on Management of Data*, pages 37–46, 2001.
- [2] M. Agyemang and C. Ezeife. A robust outlier detection scheme for large data sets. In *6th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 6–8, 2001.
- [3] I. Ben-Gal. *Maimon O. and Rockach L.: Data Mining and Knowledge Discovery Handbook: A Complete Guide for Practitioners and Researchers*. Kluwer Academic Publishers, 2005.
- [4] M. M. Breunig, H.-P. Kriegel, R. T. Ng, and J. Sander. Lof: identifying density-based local outliers. *SIGMOD Rec.*, 29(2):93–104, 2000.
- [5] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: a survey. *ACM Comput. Surv.*, 41(15):1–58, July 2009.
- [6] K. D. Crawford and R. L. Wainwright. Applying genetic algorithms to outlier detection. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 546–550, 1995.
- [7] D. Hawkins. *Identification of Outliers*. Chapman and Hall, 1980.
- [8] S. Hawkins, H. He, G. Williams, and R. Baxter. Outlier detection using replicator neural networks. In *Proc. of the Fifth Int. Conf. and Data Warehousing and Knowledge Discovery (DaWaK02)*, pages 170–180, 2002.
- [9] V. Hodge and J. Austin. A survey of outlier detection methodologies. *Artif. Intell. Rev.*, 22(2):85–126, October 2004.
- [10] J. Kennedy and R. C. Eberhart. Particle swarm optimization. In *IEEE Int. Conf. on Neural Networks*, pages 1942–1948, 1995.
- [11] E. M. Knorr, R. T. Ng, and V. Tucakov. Distance-based outliers: Algorithms and applications. *VLDB Journal: Very Large Data Bases*, 8(3–4):237–253, 2000.
- [12] H.-P. Kriegel, M. S. Hubert, and A. Zimek. Angle-based outlier detection in high-dimensional data. In *Proceeding of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 444–452, 2008.
- [13] T. N. H. League. <http://www.nhl.com/>.
- [14] UCI Repository of Machine Learning Databases. <http://archive.ics.uci.edu/ml/datasets.html>
- [15] Y. Li and H. Kitagawa. Db-outlier detection by example in high dimensional datasets. In *SWOD '07: Proceedings of the 2007 IEEE International Workshop on Databases for Next Generation Researchers*, pages 73–78, 2007.
- [16] C. M. The swarm and queen: Towards a deterministic and adaptive particle swarm optimization. In *IEEE Congress on Evolutionary Computation*, volume 2, pages 1951–1957, 1999.
- [17] S. Papadimitriou, H. Kitagawa, P. B. Gibbons, and C. Faloutsos. Loci: fast outlier detection using the local correlation integral. In *Proceedings of 19th International Conference on Data Engineering*, pages 315–326, May 2003.
- [18] Y. Pei, O. Zaiane, and Y. Gao. An efficient reference-based approach to outlier detection in large datasets. In *Sixth International Conference on Data Mining*, pages 478–487, Dec. 2006.
- [19] S. Ramaswamy, R. Rastogi, and K. Shim. Efficient algorithms for mining outliers from large data sets. *SIGMOD Rec.*, 29(2):427–438, 2000.
- [20] J. Tang, Z. Chen, A. W.-C. Fu, and D. Cheung. Lsc-mine: Algorithm for mining local outliers. In *In 15th Information Resources Management Association (IRMA) International Conference*, May 2004.
- [21] D. Ye1 and Z. Chen. A new algorithm for high-dimensional outlier detection based on constrained particle swarm intelligence. *Lecture Notes in Computer Science*, 5009/2008:516–523, May 2008.
- [22] K. Zhang, M. Hutter, and H. Jin. Title: A new local distance-based outlier detection approach for scattered real-world data. In *Proc. 13th Pacific-Asia Conf. on Know. Discov. and Data Mining*, pp. 813–822, 2009.