OUTLIER ANALYSIS

Submitted by :-
Nishant Raj
2012JE0847
III Semester
Dept. of Computer Science and Engg.

Acknowledgements

It is my great privilege to express my deepest and most sincere thanks to **Dr. Rajendra Pamula** , for his invaluable suggestions and guidance during the course of this project, without which the project would not have been successful.

I am also grateful to my our friends with whom we were able to discuss my doubts.

Introduction

Finding outlier is a collection of pattern is a very well known problem in data mining field.
Data objects which are grossly different from or inconsistent with the remaining set of data are called outliers.

Depending upon application domain outlier are of particular interest.In other cases outliers are the center of interest as in case of intrusuion detection system , credit card fraud detection system.

For example outlier may be generated due to measurement impairments,rare normal events exibiting entirely different characteristics,deliberate action etc.

there are number of methods for proposed in literature of outlier and are mainly of three type.

1. Distance based
2. Density based
3. Nearest neighbor based

1. Distance based

these technique counts the number of pattern falling under the selected threshold distance r from a point x in data sheet.
If the count is more than a preset number then x is considered as normal otherwise an outlier.

2. Density Based

These techniques measures density of a point x within a small region by counting number of points within neighbourhood region . Local Density of a point x depends on it k nearest neighbour points.

3. Nearest neighbor based

These outlier detection techniques compare the distance of point x with k nearest neighbor. If x has a short distance to its k neighbors it is considered as normal otherwise it is considered as outlier. The distance measured use is largely domain and attribute dependent.

Different types of algorithms used in outlier detection are

1. Index-based algorithm

2. Nested-loop algorithm

3. Cell-based algorithm

4. N DoT

N *DoT*

We introduce a term *Nearest Neighbor Factor* (**NNF)** measure the degree of outlierness of a point.

If *Nearest Neighbor Factor* of the point w.r.t majority of its neighbor is more than a threshold then the point is declared as a potential outlier.

BASIC TERMINOLOGIES

1. K Nearest Neighbor (knn) Set

2. Average knn distance

3. Nearest Neighbor Factor

Proposed outlier detection technique : N DoT

In this section we studied the formal outlier detection techniques by understanding the above mentioned terminologies and then developing an algorithm for the propsed method.

Finally we develop a C++ program to implement the technique

K Nearest Neighbor (knn) Set

Let *D* be a dataset of and *x* be a point in *D*.

For a natural number k and a distance function d, a set Nnk(x)= {q1 Є D|d(x,q1) < d(x,q2), q2 Є D} is called knn of x if the following two conditions hold.

(1) |Nnk| > k if q2 is not unique in D or |Nnk|=k otherwise.

(2) |Nnk \ Nq2|=k-1, where Nq2 is the set of all q2 point(s).

Average knn Distance

Let NNk be the knn of a point x in dataset D. Average knn distance of x is the average of distances between x and q belongs to Nnk ,i.e.,

Average knn distance(x)= ∑q d(x,q| qЄNnk)/|NNk|

Average knn distance of a point x is the average of distances between x and its knn.

If Average knn distance of x is less as compared to other point y, it indicates that *x*'s neighborhood is more densed compared to that of y.

Nearest Neighbor Factor(N N F)

Let x be a point in D and Nnk(x) be the knn of x .

The NNF of x with respect to q ∈ NNk(x) is the ratio of d(x,q) and Average knn distance of q.

NNF(x,q)=d(x,q)/Average knn distance(q).

How does it Work

Given a dataset D, it calculates knn and Average knn distance for all points in D.

In the next step, it computes *Nearest Neighbor Factor* for all points in the dataset using the previously calculated *knn* and *Average knn Distance.*

NDoT decides whether x is an outlier or not based on a voting mechanism.

Votes are counted based on the generated NNF values with respect to all its k nearest neighbors.

If NNF(x,q | q ∈ Nnk(x)) is more than a threshold value(=1.5 in most experiments), x is considered as an outlier with respect to q.

Subsequently, a vote is counted for x being an outlier point. If the number of votes are at least 2/3 of the number of nearest neighbors then x is declared as an outlier point.

Algorithm For N Dot(D,k)

For each x ∈ D do

    calculate knn set Nnk(x) of x.

    calculate average distance of x.

end for

for each x ∈ D do

    Count=0   /*Count counts the number of votes for x */

    for each q ∈ Nnk(x) do

        if NNF(x,q)>= 1.5 then

            Count=Count +1

        end if

    end for

    if Count >=2/3*|NNk(x)| then

        output x as an outlier in D

    end if

end for

Complexity: Time and space requirement of N DoT are as follows

1. Finding knn set and average knn distance of all point takes time of O(n^2) whre n is size of the dataset and space requirement of step is O(n).

2. Deciding a point x to be outlier or not take time O(|NNK(x)|)=O(k) . For whole dataset step takes time of O(n*k)=O(n) as k is a small constant.

C++ program to show the implementation of NDot

```
#include <iostream>

#include <stdio.h>

#include <math.h>

#include <algorithm>


int main()

{

	int k,i,j,t,n,ch;

	printf("Enter the number of points in the dataset\n===>");

	scanf("%d",&n);

	float x[n],y[n],dis[n],avgdist=0.0,cnt=0.0,nnf,x1,y1,disq[n],discpy[n],xa[n],ya[n];

	printf("Enter the coordinates of the points\n");

	for(i=0;i<n;i++)
```

```
{
        printf("===>");
        scanf("%f%f",&x[i],&y[i]);
}
while(1)
{
        printf("Enter your choice ?\n1. Detect the outlierness of a point\n2. Exit\n===>");
        scanf("%d",&ch);
        if(ch==1)
        {
                t=0;avgdist=0.0;cnt=0.0;
                printf("Enter the coordinate of the point whose outlierness is to be detected\n===>");
                scanf("%f%f",&x1,&y1);
                printf("Enter the number of elements in the Nearest Neighbour(knn) Set\n===>");
                scanf("%d",&k);
                //calculates the distances between the points
                for(i=0;i<n;i++)
                {
                        if(x1==x[i] && y1==y[i])
```

```
                discpy[i]=dis[i]=0;
            else
            {
                dis[i]=sqrt((x1-x[i])*(x1-x[i])+(y1-
y[i])*(y1-y[i]));
                discpy[i]=dis[i];
            }
        }
        std::sort(dis,dis+t);
        //calculating the knn set for the coordinate
        for(i=1;i<k;i++)
        {
            for(j=0;j<n;j++)
            {
                if(dis[i]==discpy[j])
                {
                    xa[i]=x[j];
                    ya[i]=y[j];
                }
            }
        }
```

```
//calculates the distance for each of the
//elements in knn and finds the value of nnf
for(i=1;i<k;i++)
{
    avgdist=0;
    t=0;
    for(j=0;j<n;j++)
    {
        if(xa[i]==x[j] && ya[i]==y[j])
            continue;
        else
        {
            disq[t]=sqrt((xa[i]-x[j])*(xa[i]-x[j])+
(ya[i]-y[j])*(ya[i]-y[j]));
            t++;
        }
    }
    std::sort(disq,disq+t);
    k--;
    for(j=0;j<k;j++)
    {
```

```
                avgdist=avgdist+disq[i];
            }
            avgdist=avgdist/k;
            nnf=dis[i]/avgdist;
            if(nnf>=1.5)
                cnt=cnt+1.0;
        }
        if(cnt>=(0.66666*(float)k))
            printf("Point (%.3f,%.3f) is an
outlier\n",x1,y1);
        else
            printf("Point (%.3f,%.3f) is not an
outlier\n",x1,y1);
        }
        else
            break;
    }
    return 0;
}
```