 Natural Nearest Neighbor 1

I. Algorithm 1

II. Source Code 1

III. Data 1

 EXPERIMENT 2

I. EXPERIMENTAL OBJECTIVE 2

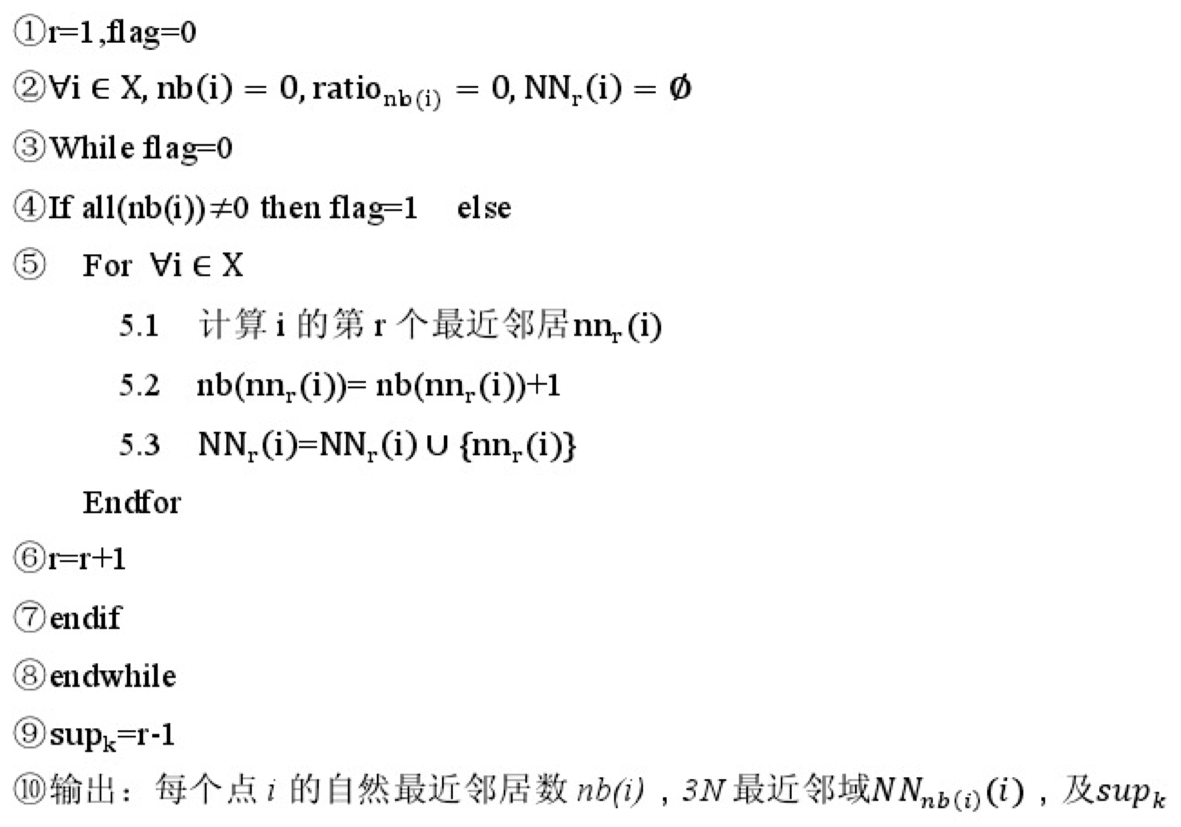
II. EXPERIMENT PRINCIPLE 2

III. EXPERIMENTAL PROCEDURE 2

APPENDIX 4

## Natural Nearest Neighbor

### Algorithm



~~表示点y的第r 最近邻~~表示元素的第r 个最近邻居，表示元素在其它点的r邻域中出现的次数，为数据集的平均邻居点数。

### Source Code**[[1]](#endnote-1)**

### Data[[2]](#endnote-2)

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Hubs与错误率的实验分析 | | | | | | | | | |
| 实验目的 | 检测具有的点是否拥有与之相高的错误率 | | | | | | | | |
| 实验数据 | **Name** |  | | **n** | | **d** |  | |  |
| **Two Moons** |  | | 14977 | | 2 | 2 | |  |
| **Abalone** | UCI | | 4177 | | 8 | 29 | |  |
| **Ionosphere** | UCI | | 351 | | 34 | 2 | |  |
| 实验方法 | * 利用自然最近邻算法求出 * 根据原始的类标签，计算每个点作为其它点最近邻的错误率 * 检测拥有的点是否拥有与之相高的错误率 | | | | | | | | |
| 实验结果 | **Name** |  | **Hubs** | | **SNK** |  | | **Rate** | |
| **Two Moons** | **10** | **0** | | **-0.2** | **False** | | **0** | |
| **Abalone** | **19** | **2%** | |  | **False** | | **0** | |
| **Ionosphere** | **14** | **12%** | | **1.54** | **True** | | **65.9%** | |
| 实验结论 | 只有在高维度的数据集中， 拥有的点才能拥有与之相高的错误率 | | | | | | | | |

## IMPROVEMENT EXPERIMENT

### EXPERIMENTAL OBJECTIVE

To verify the accuracy of spectral clustering similarity matrices with hubs in high-dimension.

### EXPERIMENT PRINCIPLE

* The main objectives of (distance-based) clustering algorithms are to minimize intra-cluster distance and maximize inter-cluster distance. The skewness of k-occurrences in high-dimensional data influences both objectives.
* Intra-cluster distance may be increased due to points with low k-occurrences. Nevertheless, the skewness of suggests that in high-dimensional data outliers are also expected due to inherent properties of vector space.
* Inter-cluster distance may be reduced due to points with high k-occurrences, that is, hubs. Like outliers, hubs do not cluster well, but for a different reason: they have low inter-cluster distance, because they are close to many points, thus also to points from other clusters.

### EXPERIMENTAL PROCEDURE

* **STEP1**

Select as hubs those points with , that is, more than two standard deviations higher than the mean. Let be the number of hubs selected.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ionosphere Data Set** | | | |
| Data Set Characteristics: | Multivariate | Number of Instances: | 351 |
| Attribute Characteristics: | Integer, Real | Number of Attributes: | 34 |

* [[3]](#endnote-3)作为其它结点最近邻的次数

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| hub | 71 | 68 | 65 | 61 | 59 | 57 | 55 | 50 | 47 | 45 | 42 | 41 | 40 | 39 | 37 | 36 | 35 | 34 | 33 | 32 | 31 | 30 | 29 |
| index | 1 | 1 | 2 | 3 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 3 | 1 | 2 | 5 | 3 | 2 | 1 |

* **STEP2**

Each hub has been considered as several points’ nearest neighbor.

Begin with calculating of training instances that consider hub as their nearest neighbor for each dimension. These calculations are performed according to Eq. (1);

(1)

The average and standard deviation of the error is and respectively.

* **STEP3:**

Calculate the normalized error and the weights of these instances according to Eqs. (2) and (3) as follows:

* **STEP4**

According to Eq. (4) the corrected hub is the average of those instances that have as their nearest neighbor.

* **STEP5**

Use the weights recalculate their similarities according to Eq. (5).

### EXPERIMENTAL CONCLUSION

逆近邻错分的次数对比

* 原始数据

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 错分次数 | 20 | 17 | 15 | 14 | 13 | 12 | 11 | 9 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 结点数 | 2 | 1 | 3 | 2 | 2 | 3 | 1 | 1 | 1 | 5 | 1 | 9 | 14 | 44 | 85 | 177 |

* 修改后的数据

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 错分次数 | 16 | 14 | 12 | 10 | 7 | 6 | 5 | 4 | 3 | 2 | 1 | 0 |
| 结点数 | 1 | 2 | 2 | 6 | 2 | 1 | 8 | 3 | 11 | 33 | 83 | 199 |

## APPENDIX

1. Source Code

   NearestNeighbor

   |  |
   | --- |
   | function NN( dataSet,dataSetName )    fprintf('Nearest Neighbor Begin...\n');    [N, dim]=size(dataSet);    %Compute Rejoint Matrix  tic;  dist = squareform(pdist(dataSet));  toc    %  [sdist,index]=sort(dist,2);      %???????  r=1;  flag=0;  nb=zeros(1,N); %??????  NNN=zeros(N,N); %????????  count=0; %???????????????????  count1=0; %???????????????  count2=0; %??????????????    %????????  while flag==0  for i=1:N  k=index(i,r+1);  nb(k)=nb(k)+1;  NNN(k,nb(k))=i;  end  r=r+1;  count2=0;  for i=1:N  if nb(i)==0  count2=count2+1;  end  end  %??nb(i)=0?????????????  if count1==count2  count=count+1;  else  count=1;  end  if count2==0 || (r>2 && count>=2) %????????  flag=1;  end  count1=count2;  end    %?????????????  SUPk=r-1; %??K??????????????  K=SUPk;  %disp(K);  fprintf('K=%d\n',K);  savePath=strcat('Result/',dataSetName,'/');  if ~exist(savePath,'dir')  mkdir(savePath);  end    k\_ocurrence=nb;  save(strcat(savePath,'k\_ocurrence.mat'),'k\_ocurrence');  save(strcat(savePath,'K.mat'),'K');  [k\_ocurrenceSort,k\_ocurrenceIndexSort]=sort(k\_ocurrence,'descend');  save(strcat(savePath,'k\_ocurrenceSort.mat'),'k\_ocurrenceSort');  save(strcat(savePath,'k\_ocurrenceIndexSort.mat'),'k\_ocurrenceIndexSort');  save(strcat(savePath,'NNN.mat'),'NNN');        % AdjointMatrixGraph(NN,0);  fprintf('Nearest Neighbor End...\n');  disp('----------------');  Error( dataSet,dataSetName,K,k\_ocurrenceSort, k\_ocurrenceIndexSort,NNN);    end |

   Error

   |  |
   | --- |
   | function Error( dataSet,dataSetName )  fprintf('Error Begin...\n');  struct=load(strcat('Result/',dataSetName,'/NNN.mat'));  names=fieldnames(struct);  NNN=struct.(names{1});  [N, dim]=size(NNN);  err=zeros(1,N);  err\_avg=zeros(1,N);  %Load Labeled DataSet  struct=load(strcat('DataSet/',dataSetName,'/',dataSetName,'\_labeled.mat'));  names=fieldnames(struct);  dataLabel=struct.(names{1});    % Compute Avg Error Rate  for i=1:N  label=dataLabel(i,1);  for j=1:N  if(NNN(i,j)==0)  break;  end  label2=dataLabel(NNN(i,j),1);  if(label~=label2)  err(i)=err(i)+1;  end  end  if(j~=1)  err\_avg(i)=err(i)/(j-1);  end  end    savePath=strcat('result/',dataSetName,'/');  if ~exist(savePath,'dir')  mkdir(savePath);  end  save(strcat(savePath,'err.mat'),'err');  save(strcat(savePath,'err\_avg.mat'),'err\_avg');  [errSort,errorIndexSort]=sort(err,2,'descend');  save(strcat(savePath,'errorIndexSort.mat'),'errorIndexSort');  save(strcat(savePath,'errSort.mat'),'errSort');  [err\_avgSort,errorIndex\_avgSort]=sort(err\_avg,2,'descend');  save(strcat(savePath,'errorIndex\_avgSort.mat'),'errorIndex\_avgSort');  save(strcat(savePath,'err\_avgSort.mat'),'err\_avgSort');      fprintf('Error End...\n');  end |

   [↑](#endnote-ref-1)
2. Data

   [AbaloneDemo](D:\\迅雷下载\\AbaloneDemo.xlsx)

   [AbaloneDemoNL](file:///D:\迅雷下载\AbaloneDemoNL.xlsx) [↑](#endnote-ref-2)
3. 33 64 26 29 85 75 53 43 26 49 29 37 21 19 24 7 83 28 58 42 56 82 38 25 47 35 40 13 25 10 18 73 75 90 83 30 56 24 34 54 31 36 91 92 92 81 26 28 88 28 67 51 25 22 48 28 38 25 90 22 38 13 16 47 17 31 37 57 21 88 9 38 63 33 46 40 15 38 26 43 41 84 44 78 21 46 33 10 22 39 15 5 28 57 78 78 2 27 29 22 85 15 18 2 29 22 28 27 20 [↑](#endnote-ref-3)