DATA MINING
LECTURE 6

Mixture of Gaussians and the EM algorithm

DBSCAN: A Density-Based Clustering Algorithm

Clustering Validation

Ανακοινώσεις

- Η προθεσμία για τη παράδοση των θεωρητικών ασκήσεων είναι Τετάρτη 4/4
  - Καθυστερημένες ασκήσεις σε μετέπειτα ημέρες θα πρέπει να παραδοθούν στην κα. Σουλίου, η οποία θα σημειώνει την ημέρα παράδοσης (Πέμπτη-Παρασκευή μέχρι τις 2:30, ή κάτω από την πόρτα μετά).
  - Μπορείτε να τις παραδώσετε και ηλεκτρονικά αν θέλετε.
- WEKA: Αν σας βγάλει out-of-memory error, ακολουθείστε τις οδηγίες και αλλάξτε το .ini αρχείο. Θα σας πει ότι δεν επιτρέπεται, οπότε πρέπει να το αντιγράψετε αλλού να το αλλάξετε και μετά να το αντιγράψετε ξανά.
- LSH Implementation: Για την υλοποίηση θα χρειαστείτε μια υλοποίηση ενός Dictionary και List. Η C++ έχει μια βιβλιοθήκη (STL), και η Java τα έχει ως built-in data types.

ΣΥΝΤΟΜΟ ΜΑΘΗΜΑ ΓΙΑ HASH TABLE IMPLEMENTATIONS

Vector, map

Container	Χαρακτηριστικά
vector	Δυναμικός πίνακας
map	Κρατάει ζεύγη κλειδιών και τιμών (key-value pairs). Συσχετίζει αντικείμενα-κλειδιά με αντικείμενα-τιμές. Το κάθε κλειδί μπορεί να συσχετίζεται με μόνο μία τιμή

vector

Μέθοδος	Λειτουργία
`size()`	επιστρέφει τον αριθμό των στοιχείων μέσα στον πίνακα
`push_back()`	προσθέτει ένα στοιχείο στο τέλος του πίνακα
`pop_back()`	αφαιρεί το τελευταίο στοιχείο του πίνακα
`back()`	επιστρέφει το τελευταίο στοιχείο του πίνακα
`operator []`	τυχαία πρόσβαση στα στοιχεία του πίνακα
`empty()`	επιστρέφει true αν το vector είναι άδειο
`insert()`	προσθέτει ένα στοιχείο σε ενδιάμεση θέση (χρησιμοποιώντας iterator)
`erase()`	αφαιρεί ένα στοιχείο από ενδιάμεση θέση (χρησιμοποιώντας iterator)

Παράδειγμα

```
#include <iostream>
#include <vector>

using namespace std;

int main(){
  vector<int> v;
  int x;

  do{
    cin >> x;
    v.push_back(x);
  }while (x != -1);

  v[v.size() - 1] = 0;

  cout << "vector elements: ";
  for (int i = 0; i < v.size(); i ++){
    cout << v[i] << " ";
  }
  cout << endl;
}
```

Παραδειγμα map

```
#include <iostream>
#include <map>
using namespace std;

int main(){
  map<string,Person*> M;
  string fname,lname;

  while (!cin.eof()){
    cin >> fname >> lname;
    Person *p =new Person(fname,lname);
    M[lname] = p;
  }

  map<string,Person*>::iterator iter = M.find("marley");
  if (iter == M.end()){
    cout << "marley is not in\n";
  }else{
    M["marley"]->PrintPersonalDetails();
  }
}
```

Iterators map

```
#include <iostream>
#include <map>
using namespace std;

int main(){
  map<string,Person*> M;
  string fname,lname;

  while (!cin.eof()){
    cin >> fname >> lname;
    Person *p =new Person(fname,lname);
    M[lname] = p;
  }

  map<string,Person*>::iterator iter;
  for (iter = M.begin(); iter != M.end(); i++){
    cout << (*iter).first << ":";
    (*iter).second->PrintPersonalDetails();
  }
}
```

`(*iter).first:` Το κλειδί του map στη θέση στην οποία δείχνει ο iterator

`(*iter).second:` Η τιμή του map στη θέση στην οποία δείχνει ο iterator

ArrayList

- Ο Container ArrayList κληρονομει από το List και αυτό από το Collection.
- Προσφέρει σειριακή αποθήκευση δεδομένων και έχει όλα τα πλεονεκτήματα και μειονεκτήματα του vector στην C++.
- Στην Java δεν επιτρέπεται υπερφόρτωση τελεστών οπότε χρησιμοποιούμε την μέθοδο get(index) για να διαβάσουμε ένα στοιχείο.
- Διάσχιση του ArrayList με την foreach εντολή είναι πιο απλή απ ότι με τον iterator.

```
import java.io.*;
import java.util.*;

public class arraylist {
    public static void main(String[] args) {
        ArrayList<Integer> A = new ArrayList<Integer>();
        for (int i = 0; i < 10; i ++){
              Random r = new Random();
              A.add(r.nextInt(100));
              System.out.println(A.get(i));
        }
        Collections.sort(A);
        System.out.println("");
        for (int x: A){
              System.out.println(x);
        }
        System.out.println("");
        System.out.println(A.toString());
    }
}
```

HashMap

- Το HashMap ορίζει ένα συνειρμικό αποθηκευτή (associative container) ο οποίος συσχετίζει κλειδιά με τιμές, κληρονομεί από την πιο γενική κλάση Map.
  - Π.χ., ο βαθμός ενός φοιτητή, η συχνότητα με την οποία εμφανίζεται μια λέξη σε ένα κείμενο.
- Η βιβλιοθήκη της Java μας δίνει πιο εύκολη πρόσβαση στα κλειδιά και τις τιμές του map.
- Χρήσιμες μέθοδοι:
  - put(key,value): προσθέτει ένα νέο key-value ζεύγος
  - containsKey(key): επιστρέφει αληθές αν υπάρχει το κλειδί
  - containsValue(value): επιστρέφει αληθές αν υπάρχει η τιμή
  - values(): επιστρέφει ένα Collection με τις τιμές
  - keySet(): επιστρέφει ένα Set με τις τιμές.

```
import java.io.*;
import java.util.*;

public class mapexample {
    public static void main(String[] args) {
        String line;
        Map<String,Integer> namesGrades = new HashMap<String,Integer>();
        try{
            FileReader fr = new FileReader("Files/in.txt");
            BufferedReader inReader = new BufferedReader(fr);

            while((line = inReader.readLine())!=null){
                System.out.println(line);
                String [] words = line.split("\t");
                Integer grade = Integer.parseInt(words[1]);
                namesGrades.put(words[0],grade);
            }
        } catch(IOException ex){
            System.out.println("IO Error" + ex);
        }
        for(String x: namesGrades.keySet()){
                System.out.println(x + " -- " + namesGrades.get(x));
        }
    }
}
```

MIXTURE MODELS AND THE EM ALGORITHM

Model-based clustering

- In order to understand our data, we will assume that there is a generative process (a model) that creates/describes the data, and we will try to find the model that best fits the data.
  - Models of different complexity can be defined, but we will assume that our model is a distribution from which data points are sampled
  - Example: the data is the height of all people in Greece

- In most cases, a single distribution is not good enough to describe all data points: different parts of the data follow a different distribution
  - Example: the data is the height of all people in Greece and China
  - We need a mixture model
  - Different distributions correspond to different clusters in the data.

Gaussian Distribution

- Example: the data is the height of all people in Greece
  - Experience has shown that this data follows a Gaussian (Normal) distribution
  - Reminder: Normal distribution:

$$P(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

  - μ = mean, σ = standard deviation

Gaussian Model

- What is a model?
  - A Gaussian distribution is fully defined by the mean μ and the standard deviation σ
  - We define our model as the pair of parameters $\theta = (\mu, \sigma)$

- This is a general principle: a model is defined as a vector of parameters θ

Fitting the model

- We want to find the normal distribution that best fits our data
  - Find the best values for μ and σ
  - But what does best fit mean?

Maximum Likelihood Estimation (MLE)

- Suppose that we have a vector $X = (x_1, \ldots, x_n)$ of values
- And we want to fit a Gaussian $N(\mu, \sigma)$ model to the data
- Probability of observing point x_i:

$$P(x_i) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- Probability of observing all points (assume independence)

$$P(X) = \prod_{i=1}^{n} P(x_i) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- We want to find the parameters $\theta = (\mu, \sigma)$ that maximize the probability $P(X|\theta)$

Maximum Likelihood Estimation (MLE)

- The probability $P(X|\theta)$ as a function of θ is called the Likelihood function

$$L(\theta) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x_i-\mu)^2}{2\sigma^2}}$$

- It is usually easier to work with the Log-Likelihood function

$$LL(\theta) = -\sum_{i=1}^{n} \frac{(x_i-\mu)^2}{2\sigma^2} - \frac{1}{2} n \log 2\pi - n \log \sigma$$

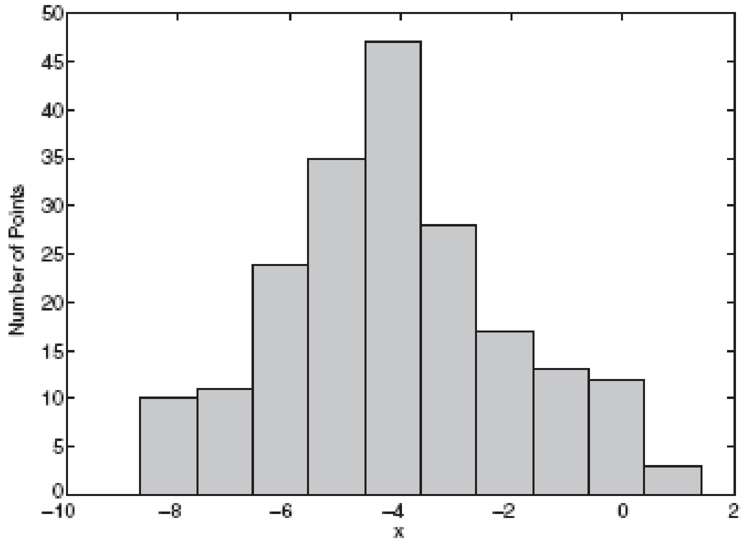
- Maximum Likelihood Estimation
  - Find parameters μ, σ that maximize $LL(\theta)$

$$\mu = \frac{1}{n}\sum_{i=1}^{n} x_i = \mu_X$$

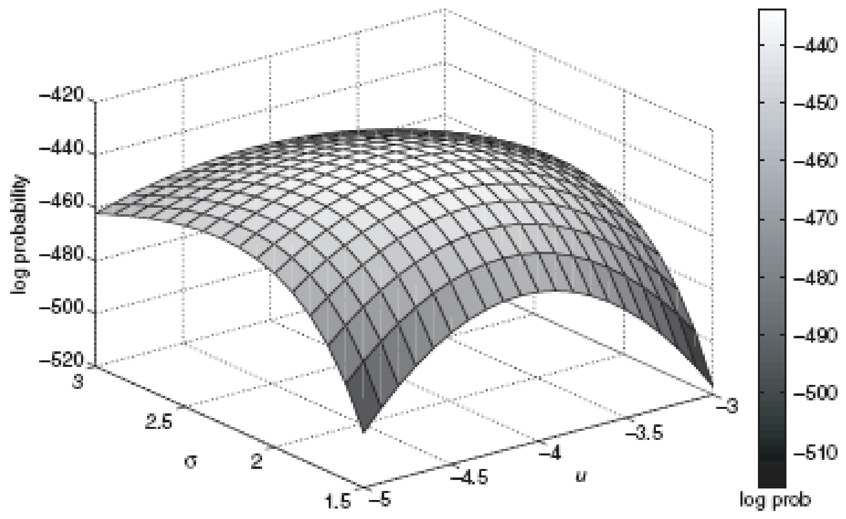
Sample Mean

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} (x_i-\mu)^2 = \sigma_X^2$$

Sample Variance



(a) Histogram of 200 points from a Gaussian distribution.

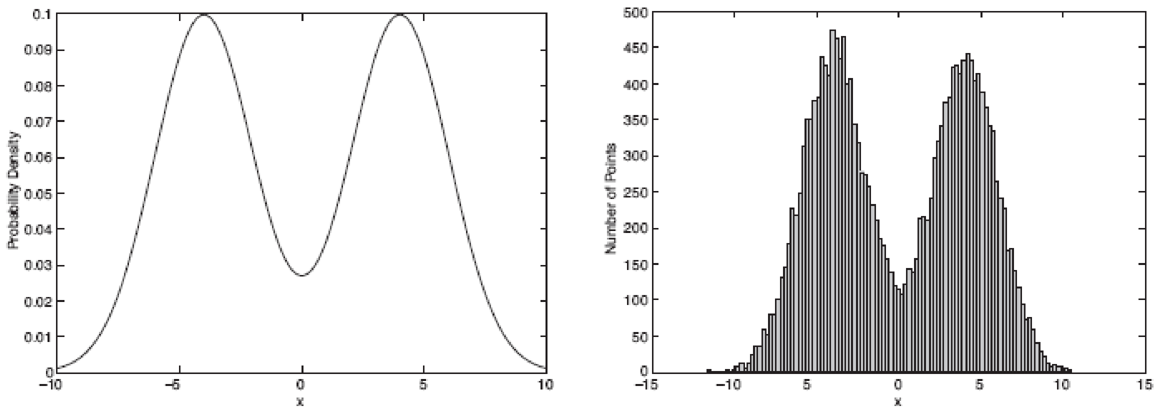


(b) Log likelihood plot of the 200 points for different values of the mean and standard deviation.

Figure 9.3. 200 points from a Gaussian distribution and their log probability for different parameter values.

Mixture of Gaussians

- Suppose that you have the heights of people from Greece and China and the distribution looks like the figure below (dramatization)



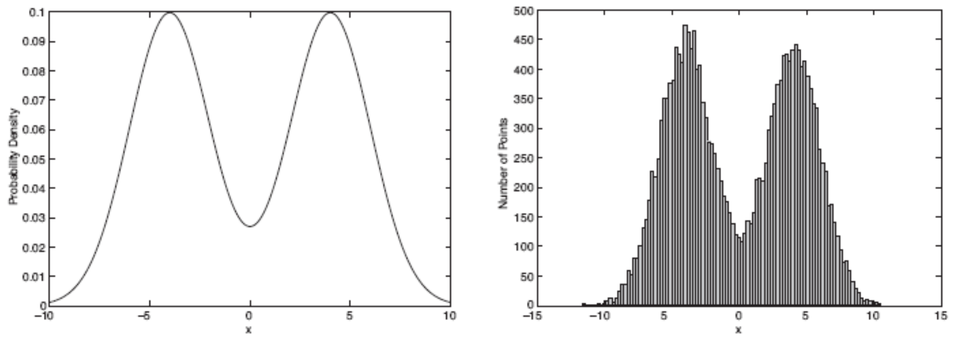
(a) Probability density function for the mixture model.

(b) 20,000 points generated from the mixture model.

Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

Mixture of Gaussians

- In this case the data is the result of the mixture of two Gaussians
  - One for Greek people, and one for Chinese people
  - Identifying for each value which Gaussian is most likely to have generated it will give us a clustering.



(a) Probability density function for the mixture model.

(b) 20,000 points generated from the mixture model.

Figure 9.2. Mixture model consisting of two normal distributions with means of -4 and 4, respectively. Both distributions have a standard deviation of 2.

Mixture model

- A value x_i is generated according to the following process:
  - First I select the nationality
    - With probability π_G I select Greek, with probability π_C I select China $(\pi_G + \pi_C = 1)$
  - Given the nationality, I generate the point from the corresponding Gaussian
    - $P(x_i|\theta_G) \sim N(\mu_G, \sigma_G)$ if Greece
    - $P(x_i|\theta_G) \sim N(\mu_G, \sigma_G)$ if China

Mixture Model

- For value x_i, we have:

$$P(x_i) = \pi_G P(x_i|\theta_G) + \pi_C P(x_i|\theta_C)$$

- For all values $X = (x_1, \dots, x_n)$

$$P(X) = \prod_{i=1}^{n} P(x_i)$$

- Our model has the following parameters

$$\Theta = (\underbrace{\pi_G, \pi_C,}_{\text{Mixture probabilities}} \underbrace{\mu_G, \mu_C, \sigma_G, \sigma_C}_{\text{Distribution Parameters}})$$

Mixture probabilities

Distribution Parameters

- We want to estimate the parameters that maximize the Likelihood of the data

Mixture Models

- Once we have the parameters $\Theta = (\pi_G, \pi_C, \mu_G, \mu_C, \sigma_G, \sigma_C)$ we can estimate the membership probabilities $P(G|x_i)$ and $P(C|x_i)$ for each point x_i:
  - This is the probability that point x_i belongs to the Greek or the Chinese population (cluster)

$$P(G|x_i) = \frac{P(x_i|G)P(G)}{P(x_i|G)P(G) + P(x_i|C)P(C)}$$

$$= \frac{P(x_i|G)\pi_G}{P(x_i|G)\pi_G + P(x_i|C)\pi_C}$$

EM (Expectation Maximization) Algorithm

- Initialize the values of the parameters in Θ to some random values
- Repeat until convergence
  - E-Step: Given the parameters Θ estimate the membership probabilities $P(G|x_i)$ and $P(C|x_i)$
  - M-Step: Compute the parameter values that (in expectation) maximize the data likelihood

$$\pi_G = \frac{1}{n}\sum_{i=1}^{n} P(G|x_i) \qquad \pi_C = \frac{1}{n}\sum_{i=1}^{n} P(C|x_i)$$

$$\mu_C = \sum_{i=1}^{n} \frac{P(C|x_i)}{n * \pi_C} x_i \qquad \mu_G = \sum_{i=1}^{n} \frac{P(G|x_i)}{n * \pi_G} x_i$$

MLE Estimates if π's were fixed

$$\sigma_C^2 = \sum_{i=1}^{n} \frac{P(C|x_i)}{n * \pi_C} (x_i - \mu_C)^2 \qquad \sigma_G^2 = \sum_{i=1}^{n} \frac{P(G|x_i)}{n * \pi_G} (x_i - \mu_G)^2$$

Relationship to K-means

- E-Step: Assignment of points to clusters
  - K-means: hard assignment, EM: soft assignment
- M-Step: Computation of centroids
  - K-means assumes common fixed variance (spherical clusters)
  - EM: can change the variance for different clusters or different dimensions (elipsoid clusters)
- If the variance is fixed then both minimize the same error function

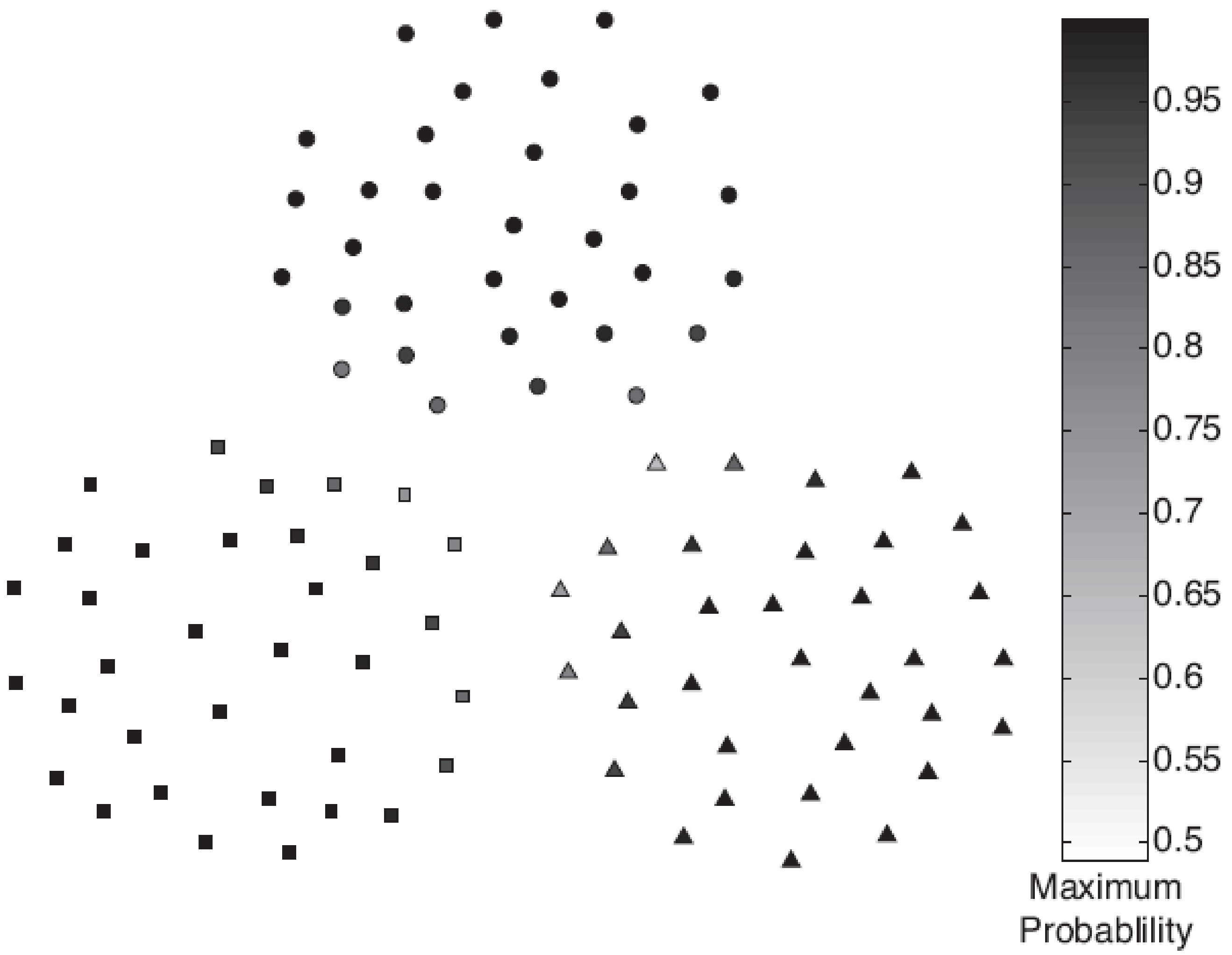


Figure 9.4. EM clustering of a two-dimensional point set with three clusters.

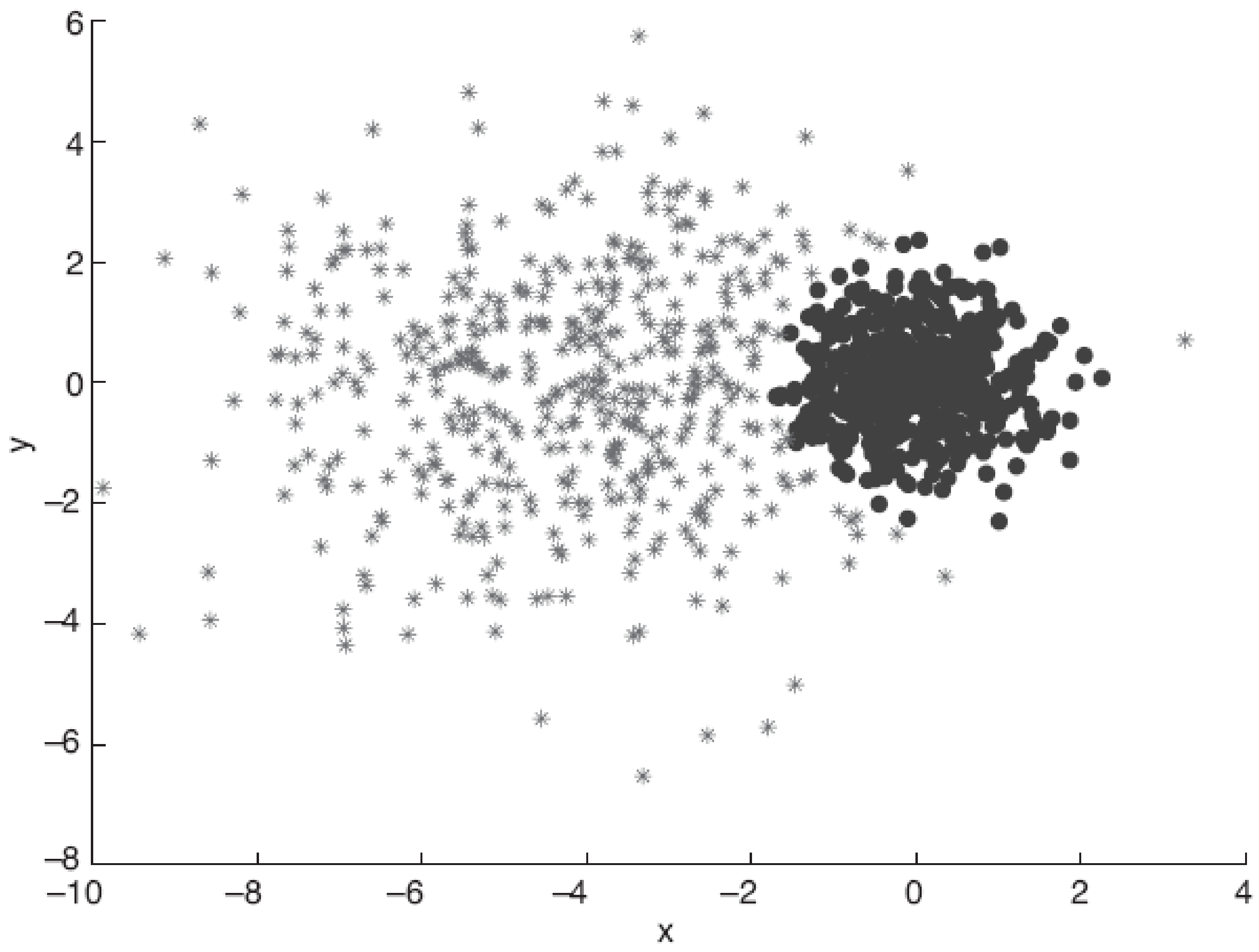
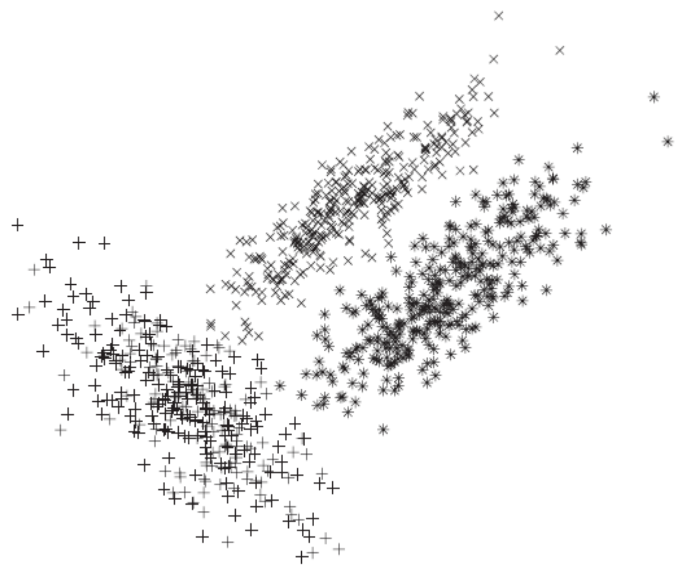
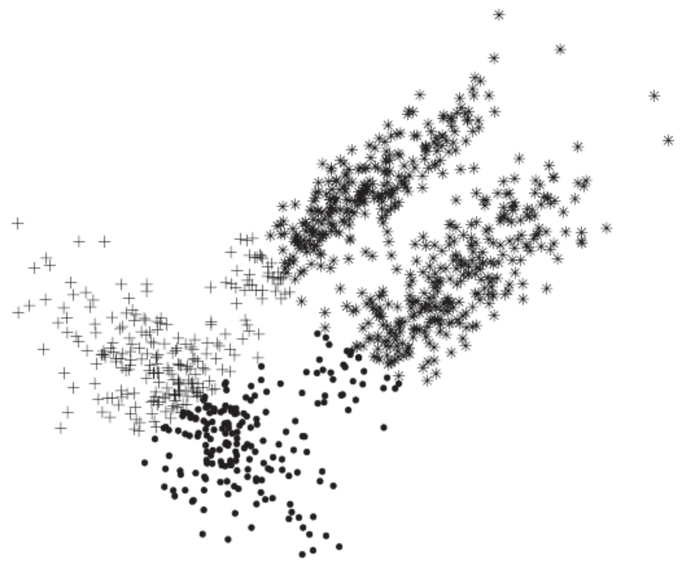


Figure 9.5. EM clustering of a two-dimensional point set with two clusters of differing density.



(a) Clusters produced by mixture model clustering.



(b) Clusters produced by K-means clustering.

Figure 9.6. Mixture model and K-means clustering of a set of two-dimensional points.

DBSCAN: A DENSITY-BASED CLUSTERING ALGORITHM

Thanks to:

“Introduction to Data Mining” by Tan, Steinbach, Kumar.

DBSCAN: Density-Based Clustering

- DBSCAN is a Density-Based Clustering algorithm

- Reminder: In density based clustering we partition points into dense regions separated by not-so-dense regions.

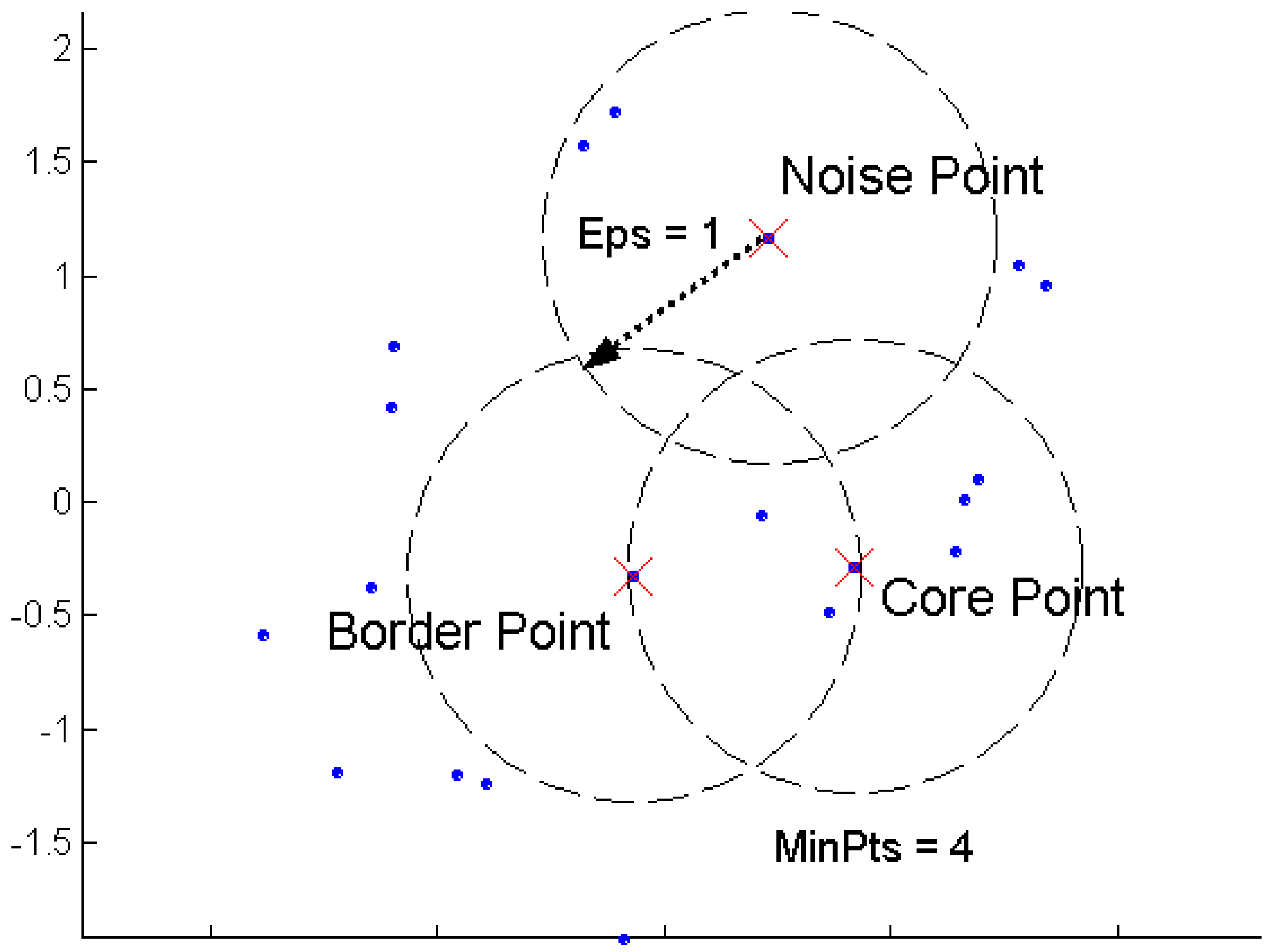
- Important Questions:
  - How do we measure density?
  - What is a dense region?

- DBSCAN:
  - Density at point p: number of points within a circle of radius Eps
  - Dense Region: A circle of radius Eps that contains at least MinPts points

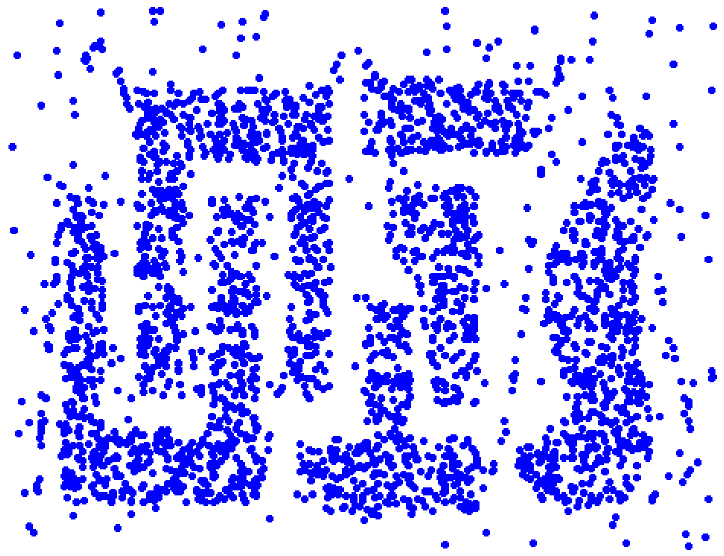
DBSCAN

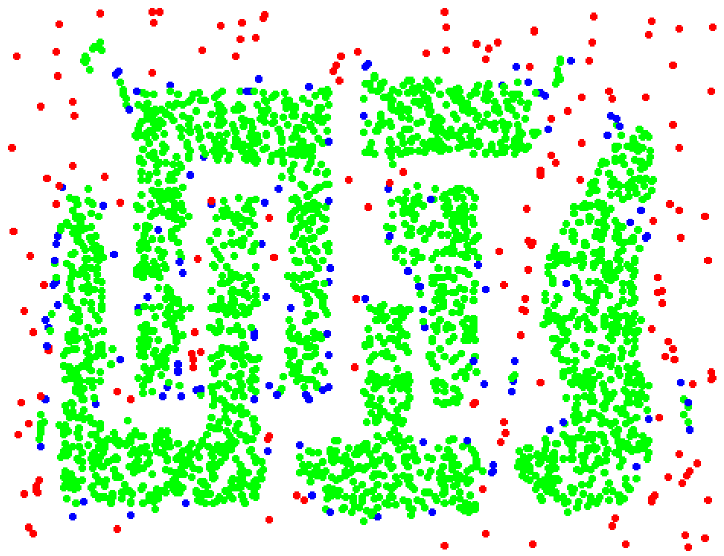
- Characterization of points
  - A point is a core point if it has more than a specified number of points (MinPts) within Eps
    - These points belong in a dense region and are at the interior of a cluster
  - A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.
  - A noise point is any point that is not a core point or a border point.

DBSCAN: Core, Border, and Noise Points



DBSCAN: Core, Border and Noise Points

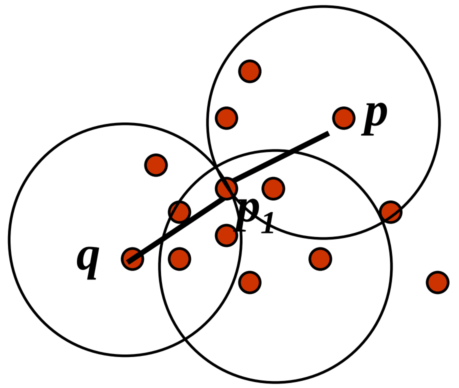

Original Points


Point types: core, border and noise

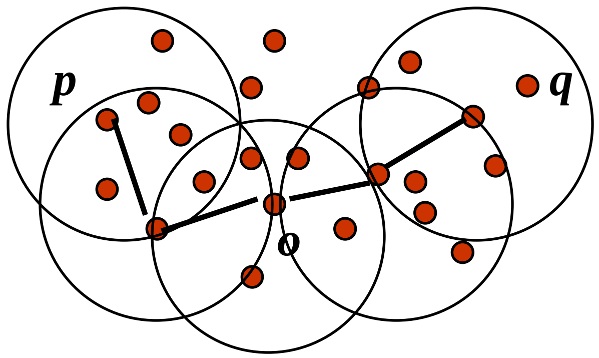
Eps = 10, MinPts = 4

Density-Connected points

- Density edge
  - We place an edge between two core points q and p if they are within distance Eps.



- Density-connected
  - A point p is density-connected to a point q if there is a path of edges from p to q

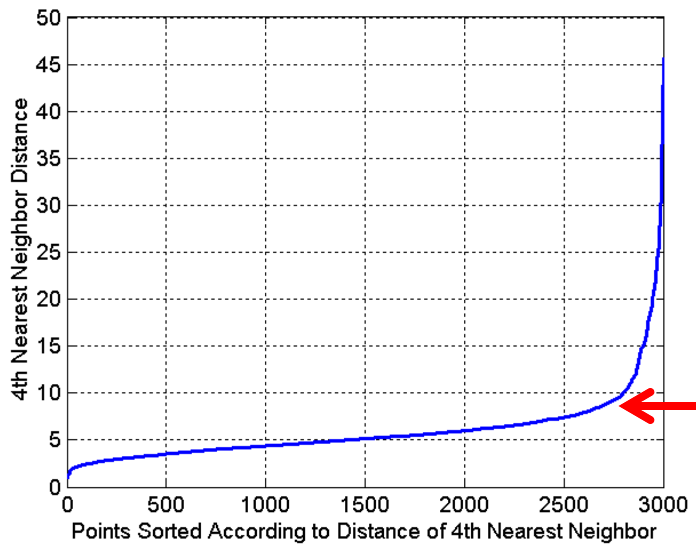


DBSCAN Algorithm

- Label points as core, border and noise
- Eliminate noise points
- For every core point p that has not been assigned to a cluster
  - Create a new cluster with the point p and all the points that are density-connected to p.
- Assign border points to the cluster of the closest core point.

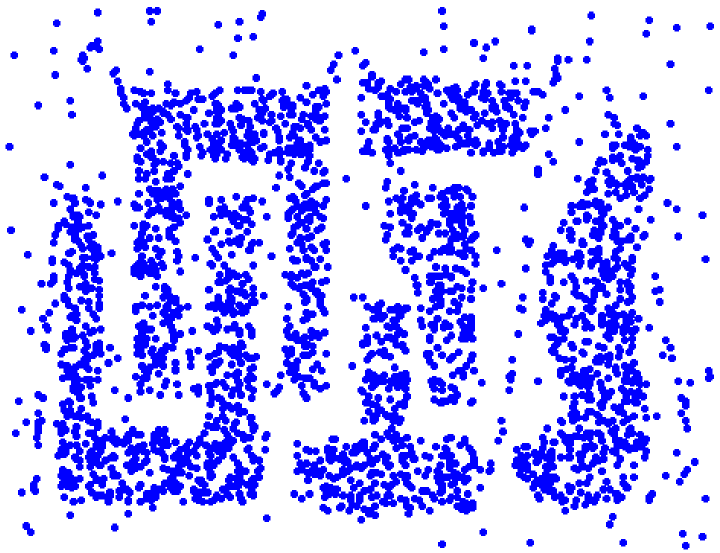
DBSCAN: Determining Eps and MinPts

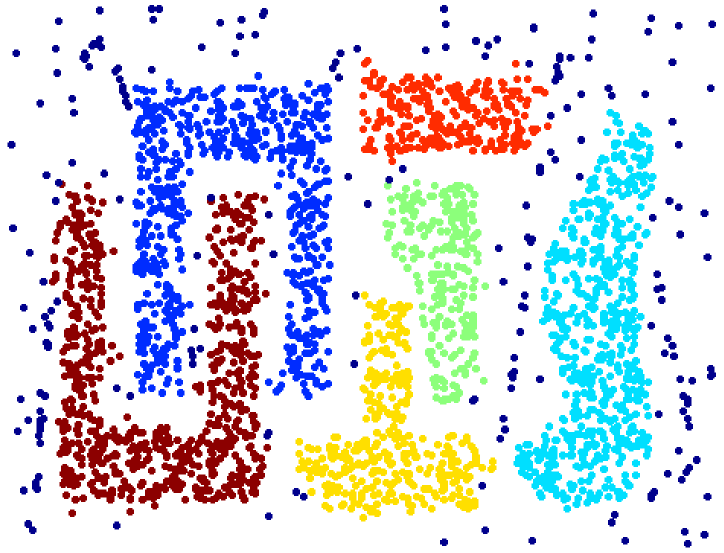
- Idea is that for points in a cluster, their k^th nearest neighbors are at roughly the same distance
- Noise points have the k^th nearest neighbor at farther distance
- So, plot sorted distance of every point to its k^th nearest neighbor
- Find the distance d where there is a "knee" in the curve
  - Eps = d, MinPts = k



Eps ~ 7-10
MinPts = 4

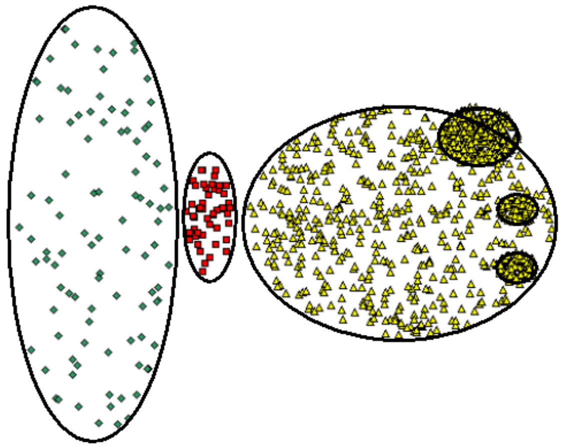
When DBSCAN Works Well


Original Points

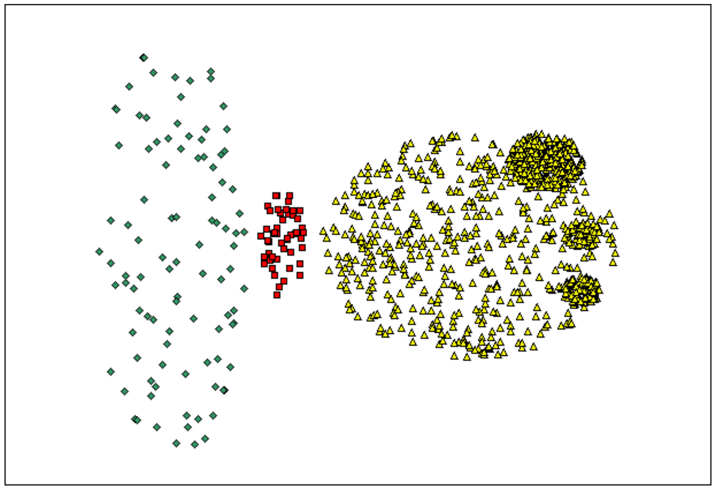

Clusters

- Resistant to Noise
- Can handle clusters of different shapes and sizes

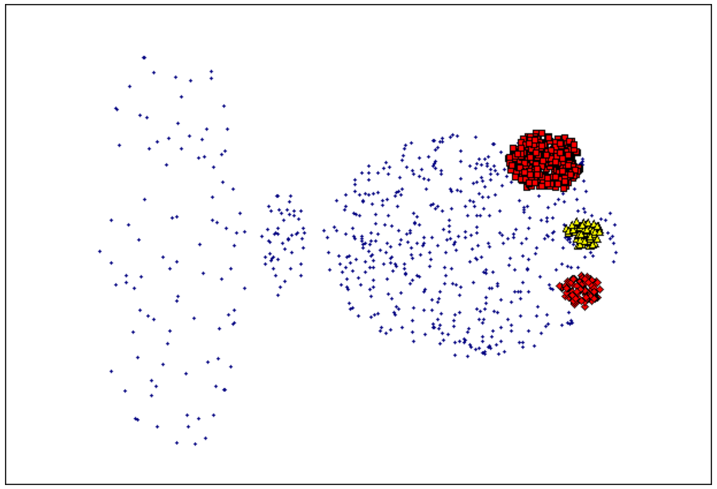
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.75).



(MinPts=4, Eps=9.92)

- Varying densities
- High-dimensional data

DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.

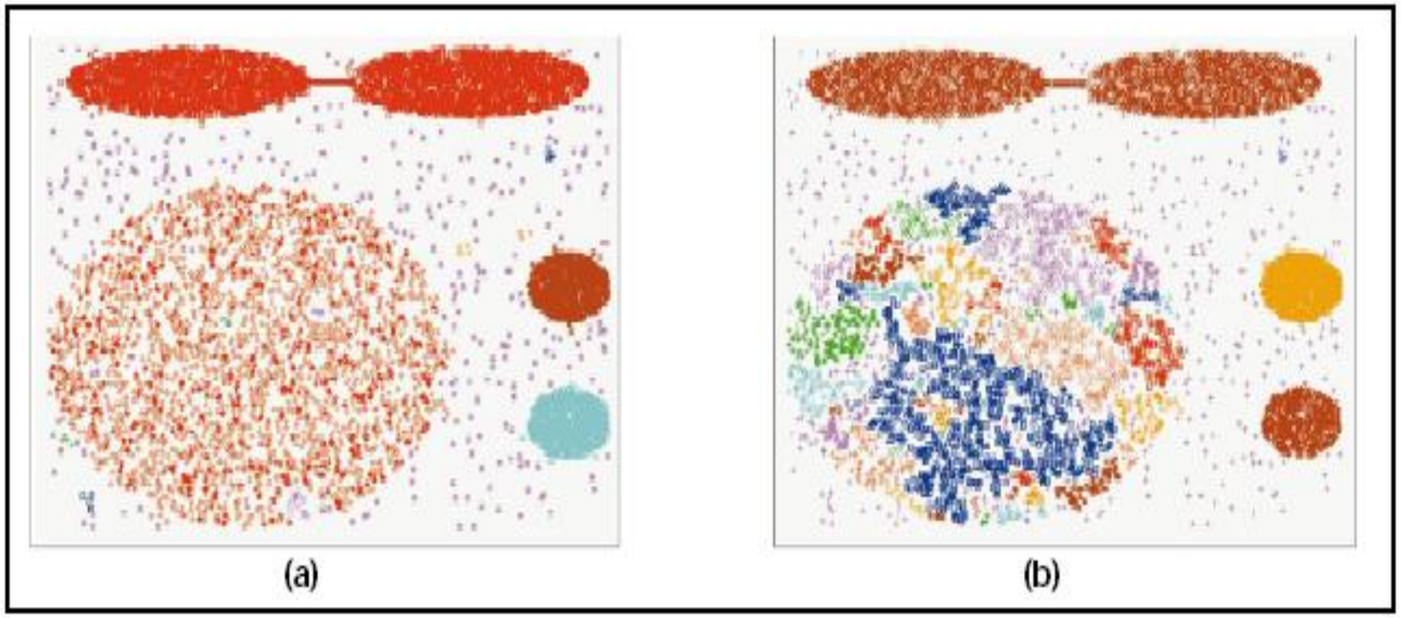
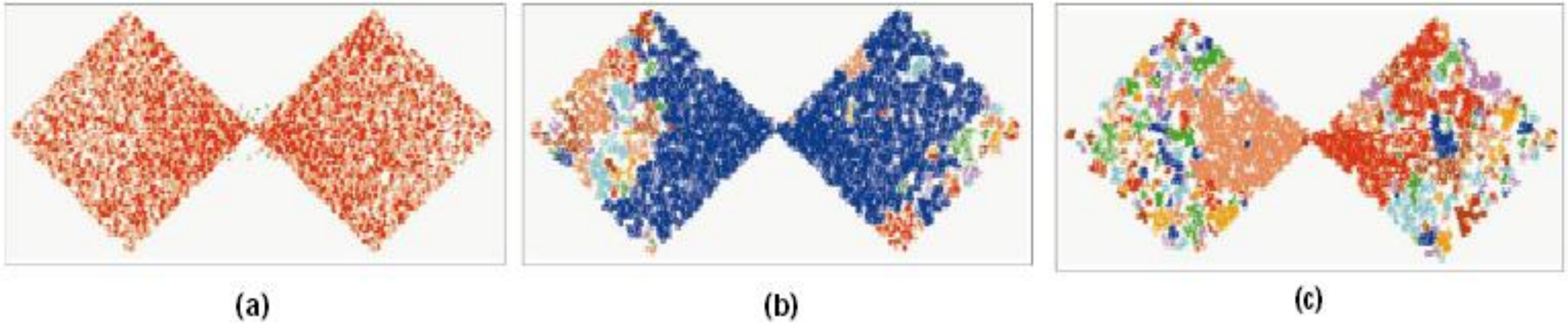


Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



Other algorithms

- PAM, CLARANS: Solutions for the k-medoids problem
- BIRCH: Constructs a hierarchical tree that acts a summary of the data, and then clusters the leaves.
- MST: Clustering using the Minimum Spanning Tree.
- ROCK: clustering categorical data by neighbor and link analysis
- LIMBO, COOLCAT: Clustering categorical data using information theoretic tools.
- CURE: Hierarchical algorithm uses different representation of the cluster
- CHAMELEON: Hierarchical algorithm uses closeness and interconnectivity for merging

CLUSTERING VALIDITY

Cluster Validity

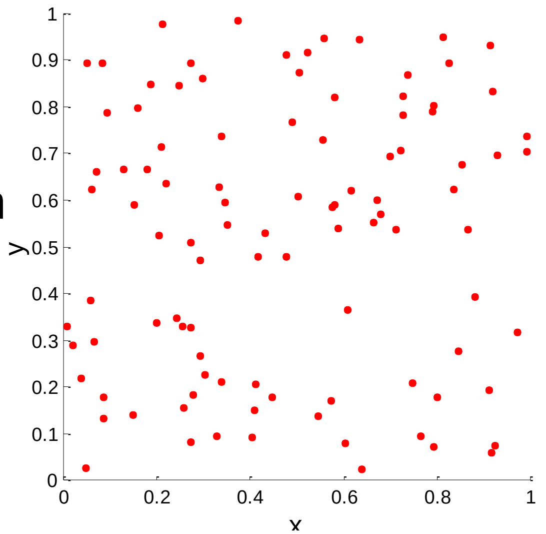
- How do we evaluate the “goodness” of the resulting clusters?

- But “clusters are in the eye of the beholder”!

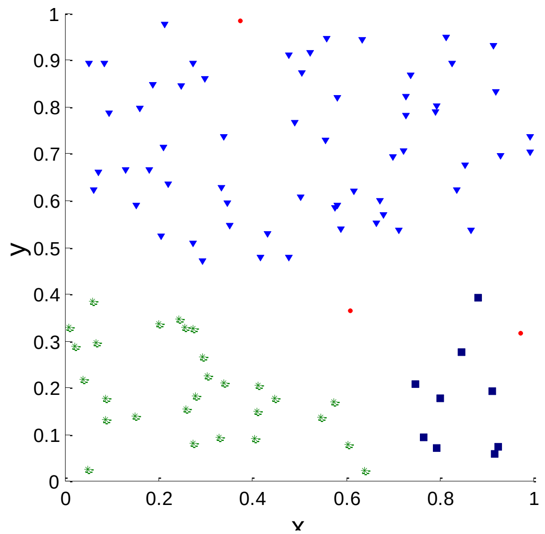
- Then why do we want to evaluate them?
  - To avoid finding patterns in noise
  - To compare clustering algorithms
  - To compare two sets of clusters
  - To compare two clusters

Clusters found in Random Data

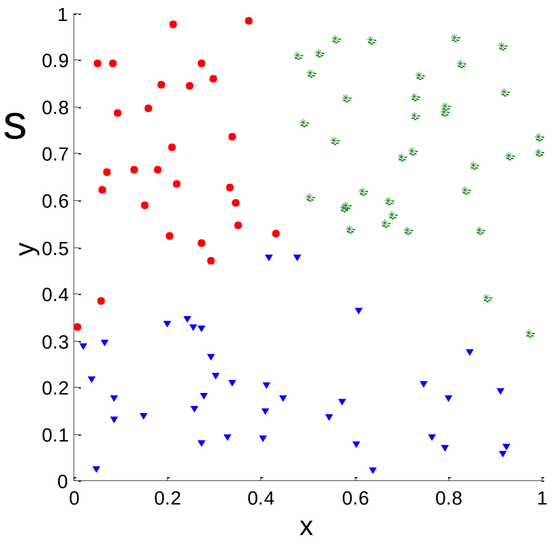
Random Points



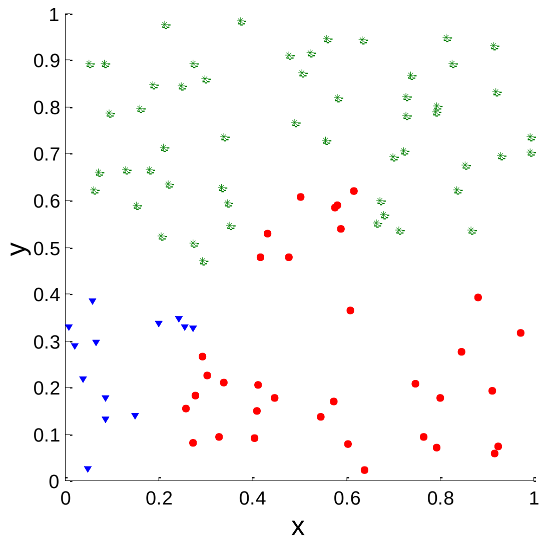
DBSCAN



K-means



Complete Link



Different Aspects of Cluster Validation

1. Determining the clustering tendency of a set of data, i.e., distinguishing whether non-random structure actually exists in the data.
2. Comparing the results of a cluster analysis to externally known results, e.g., to externally given class labels.
3. Evaluating how well the results of a cluster analysis fit the data *without* reference to external information.
   - Use only the data
4. Comparing the results of two different sets of cluster analyses to determine which is better.
5. Determining the ‘correct’ number of clusters.

For 2, 3, and 4, we can further distinguish whether we want to evaluate the entire clustering or just individual clusters.

Measures of Cluster Validity

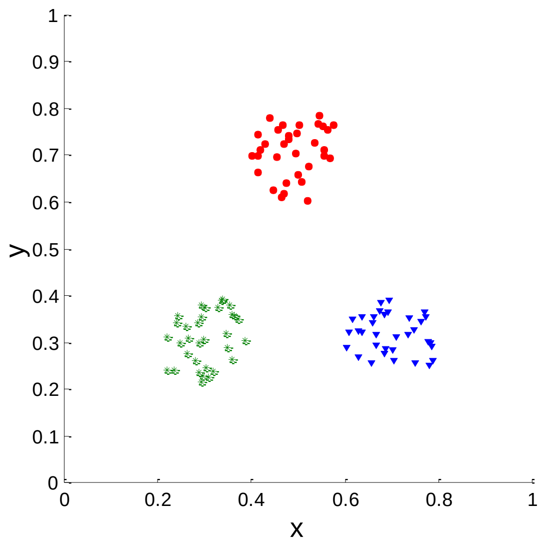
- Numerical measures that are applied to judge various aspects of cluster validity, are classified into the following three types.
  - External Index: Used to measure the extent to which cluster labels match externally supplied class labels.
    - Entropy
  - Internal Index: Used to measure the goodness of a clustering structure *without* respect to external information.
    - Sum of Squared Error (SSE)
  - Relative Index: Used to compare two different clusterings or clusters.
    - Often an external or internal index is used for this function, e.g., SSE or entropy
- Sometimes these are referred to as criteria instead of indices
  - However, sometimes criterion is the general strategy and index is the numerical measure that implements the criterion.

Measuring Cluster Validity Via Correlation

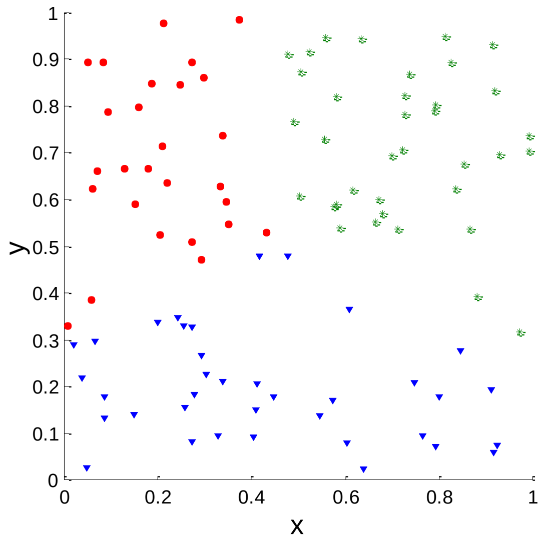
- Two matrices
  - Proximity Matrix
  - “Incidence” Matrix
    - One row and one column for each data point
    - An entry is 1 if the associated pair of points belong to the same cluster
    - An entry is 0 if the associated pair of points belongs to different clusters
- Compute the correlation between the two matrices
  - Since the matrices are symmetric, only the correlation between n(n-1) / 2 entries needs to be calculated.
- High correlation indicates that points that belong to the same cluster are close to each other.
- Not a good measure for some density or contiguity based clusters.

Measuring Cluster Validity Via Correlation

- Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.



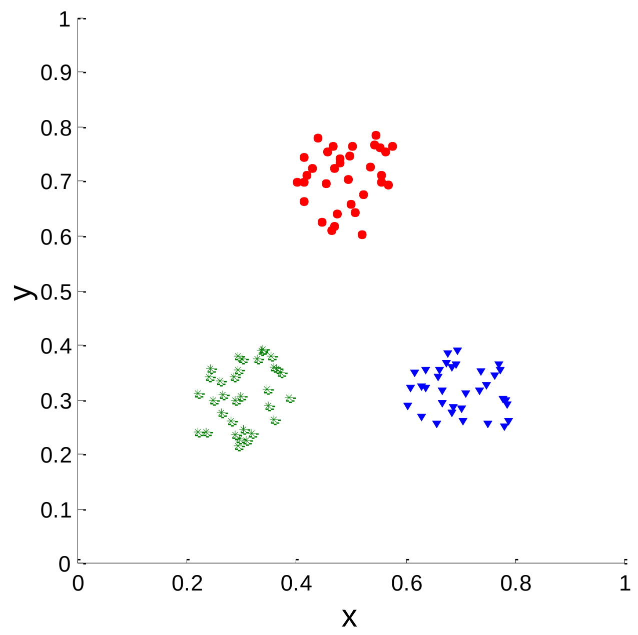
Corr = -0.9235

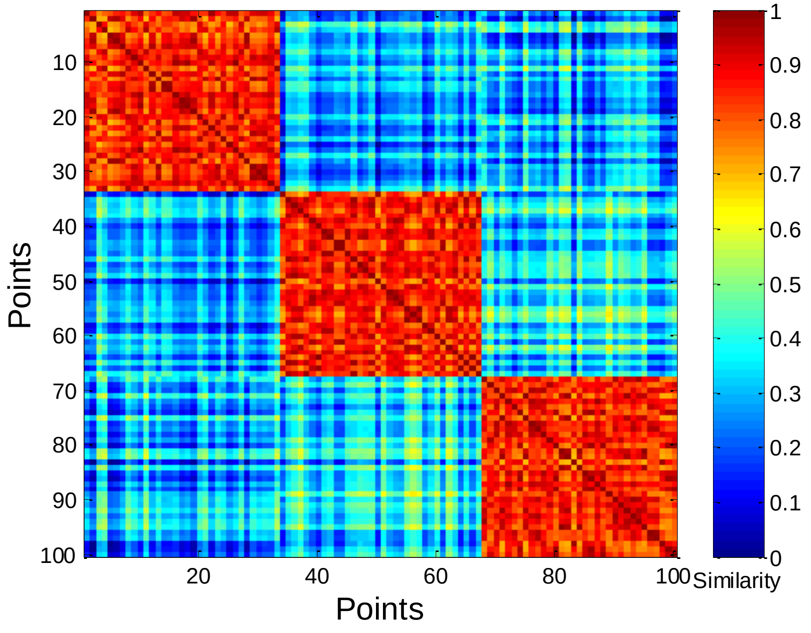


Corr = -0.5810

Using Similarity Matrix for Cluster Validation

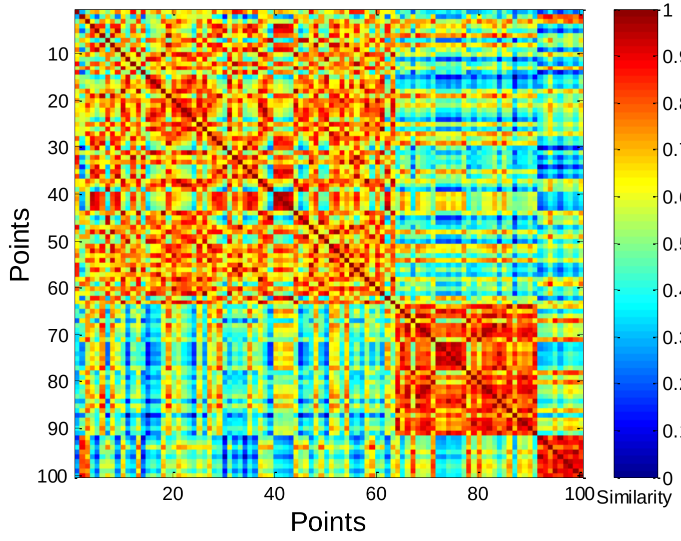
- Order the similarity matrix with respect to cluster labels and inspect visually.

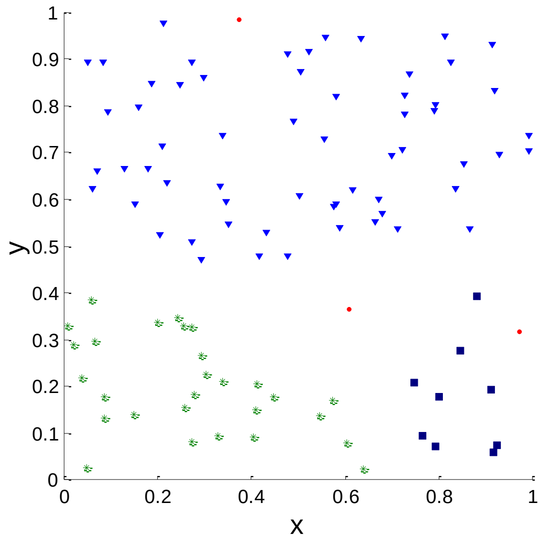




Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

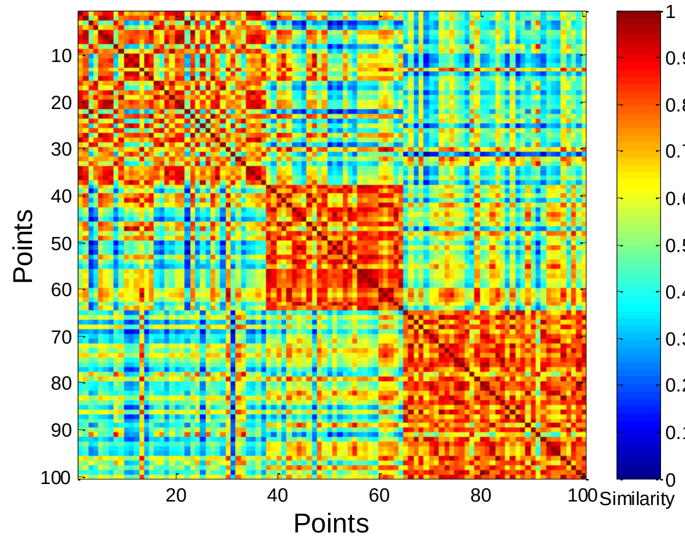


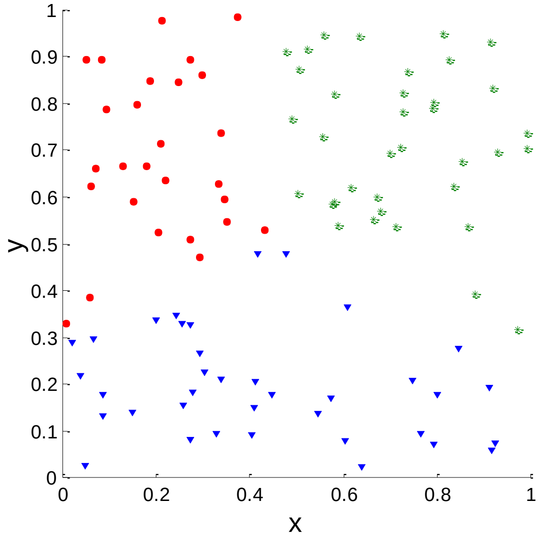


DBSCAN

Using Similarity Matrix for Cluster Validation

- Clusters in random data are not so crisp

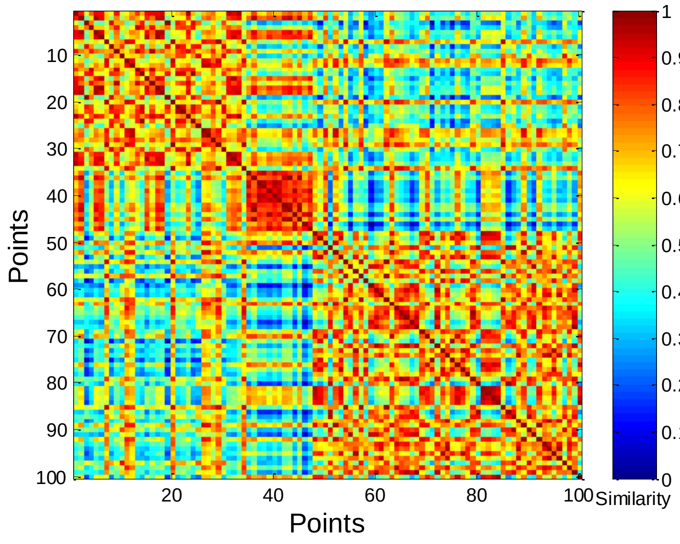


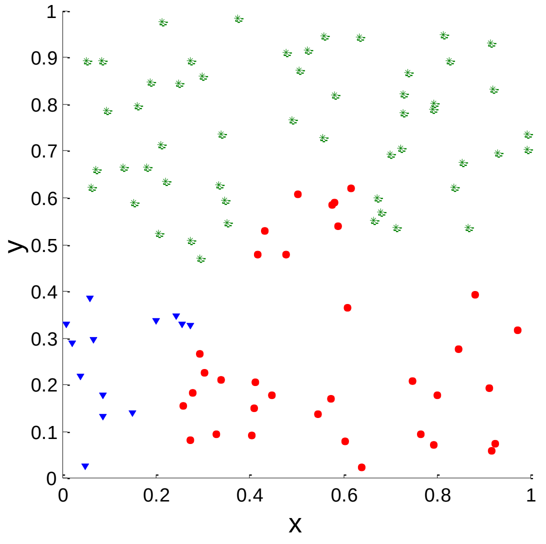


K-means

Using Similarity Matrix for Cluster Validation

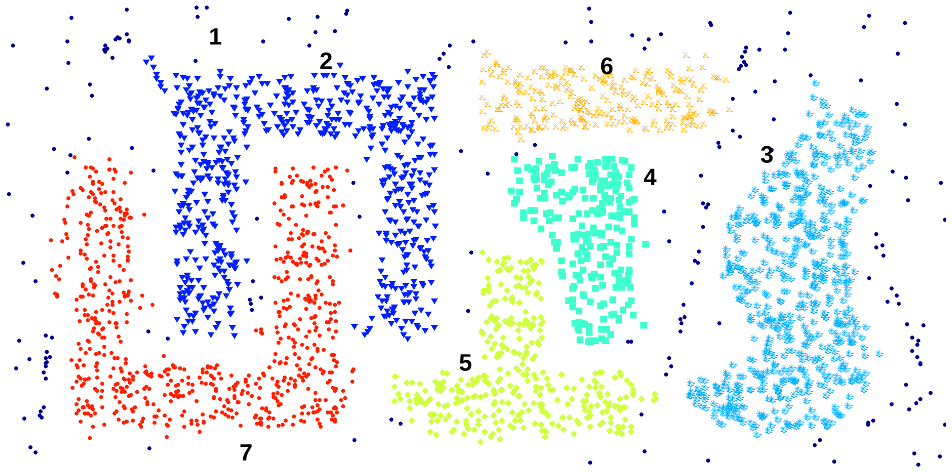
- Clusters in random data are not so crisp

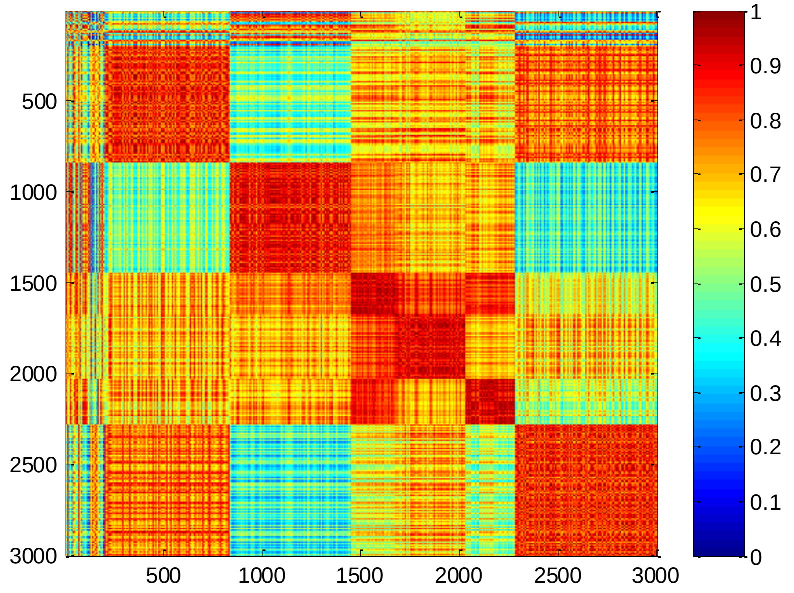




Complete Link

Using Similarity Matrix for Cluster Validation

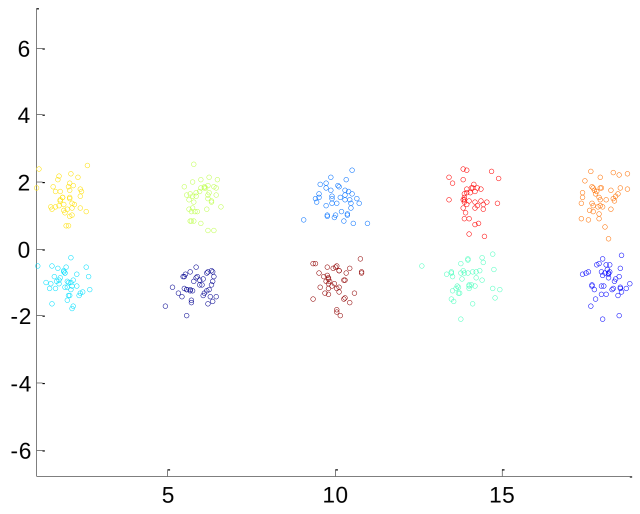


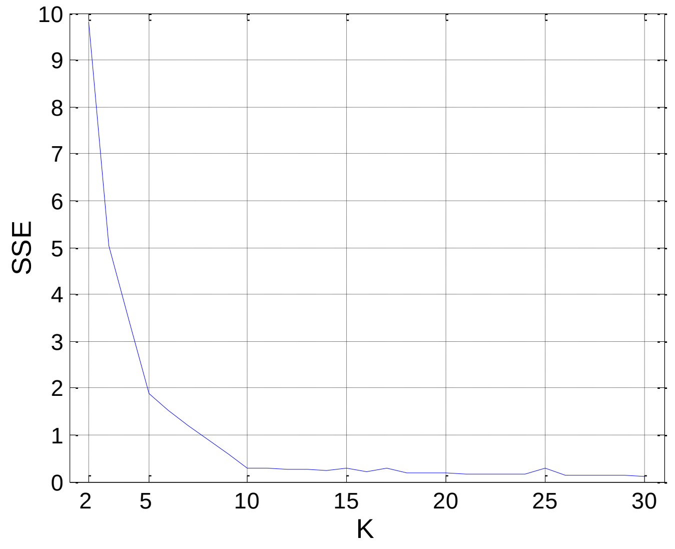


DBSCAN

Internal Measures: SSE

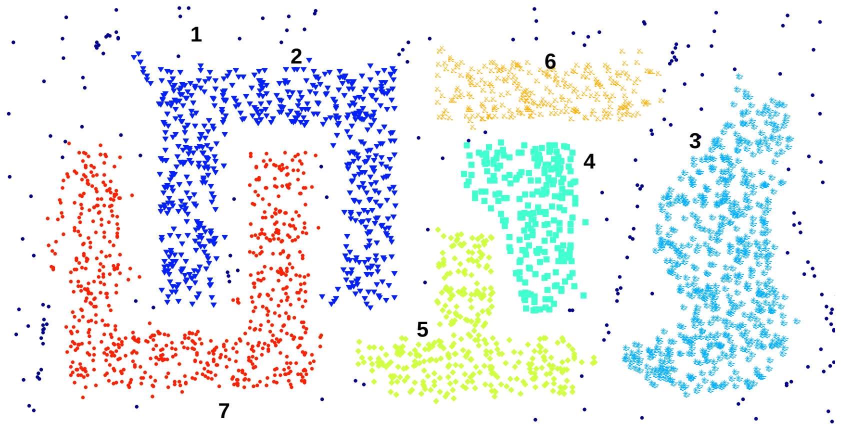
- Clusters in more complicated figures aren't well separated
- Internal Index: Used to measure the goodness of a clustering structure without respect to external information
  - SSE
- SSE is good for comparing two clusterings or two clusters (average SSE).
- Can also be used to estimate the number of clusters

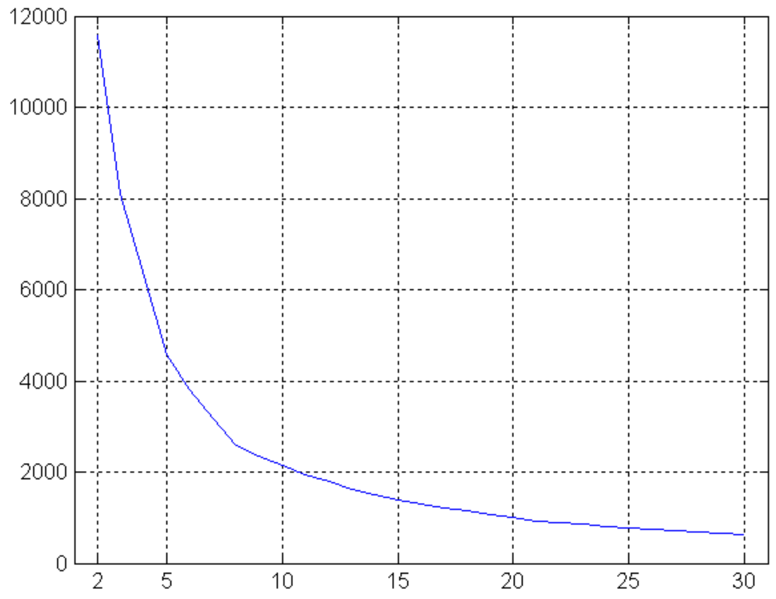




Internal Measures: SSE

- SSE curve for a more complicated data set





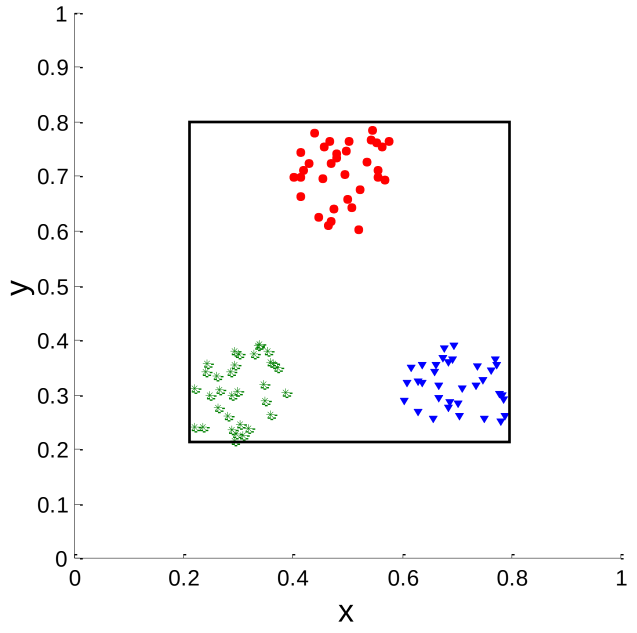
SSE of clusters found using K-means

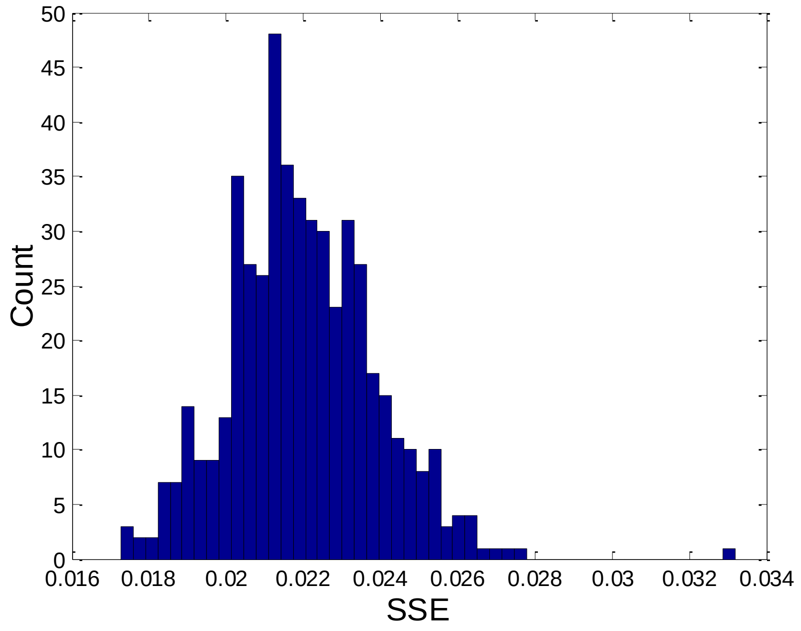
Framework for Cluster Validity

- Need a framework to interpret any measure.
  - For example, if our measure of evaluation has the value, 10, is that good, fair, or poor?
- Statistics provide a framework for cluster validity
  - The more “atypical” a clustering result is, the more likely it represents valid structure in the data
  - Can compare the values of an index that result from random data or clusterings to those of a clustering result.
    - If the value of the index is unlikely, then the cluster results are valid
  - These approaches are more complicated and harder to understand.
- For comparing the results of two different sets of cluster analyses, a framework is less necessary.
  - However, there is the question of whether the difference between two index values is significant

Statistical Framework for SSE

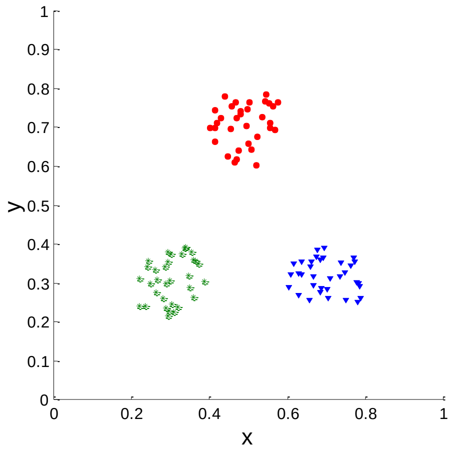
- Example
  - Compare SSE of 0.005 against three clusters in random data
  - Histogram shows SSE of three clusters in 500 sets of random data points of size 100 distributed over the range 0.2 – 0.8 for x and y values

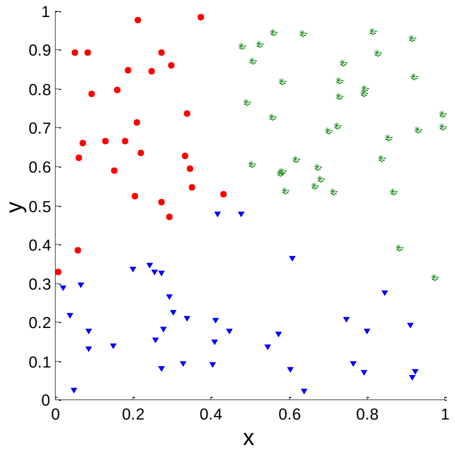


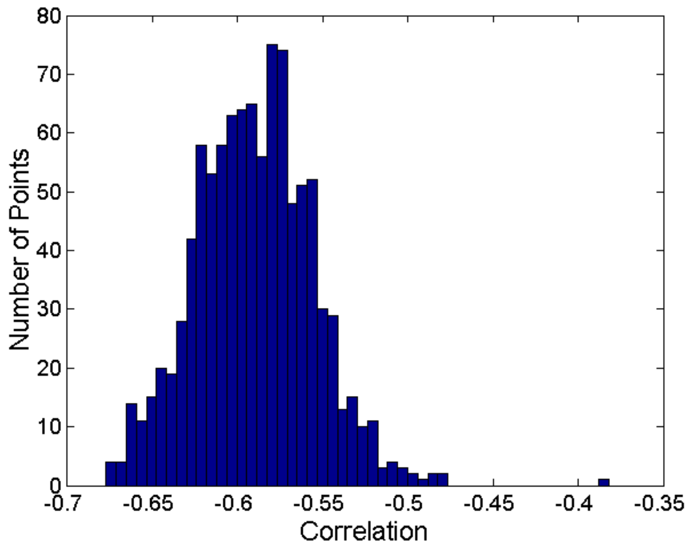


Statistical Framework for Correlation

- Correlation of incidence and proximity matrices for the K-means clusterings of the following two data sets.







Corr = -0.9235

Corr = -0.5810

Internal Measures: Cohesion and Separation

- Cluster Cohesion: Measures how closely related are objects in a cluster
  - Example: SSE
- Cluster Separation: Measure how distinct or well-separated a cluster is from other clusters
- Example: Squared Error
  - Cohesion is measured by the within cluster sum of squares (SSE)

$$WSS = \sum_{i} \sum_{x \in C_i} (x - m_i)^2$$

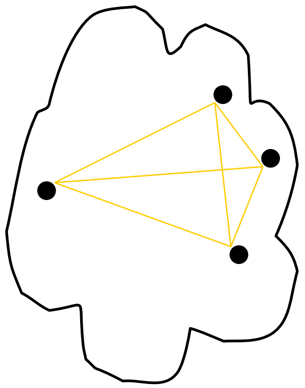
  - Separation is measured by the between cluster sum of squares

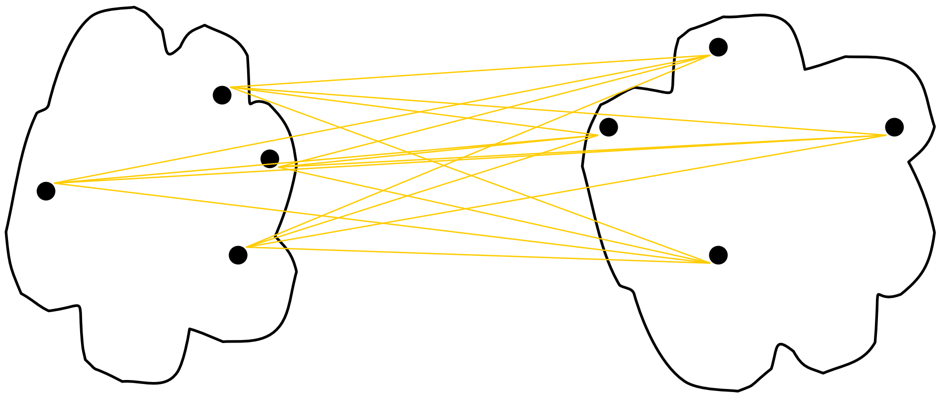
$$BSS = \sum_{i} |C_i| (m - m_i)^2$$

    - Where $|C_i|$ is the size of cluster i

Internal Measures: Cohesion and Separation

- A proximity graph based approach can also be used for cohesion and separation.
  - Cluster cohesion is the sum of the weight of all links within a cluster.
  - Cluster separation is the sum of the weights between nodes in the cluster and nodes outside the cluster.


cohesion


separation

External Measures for Clustering Validity

- Assume that the data is labeled with some class labels
    - E.g., documents are classified into topics, people classified according to their income, senators classified as republican or democrat.
- In this case we want the clusters to be homogeneous with respect to classes
    - Each cluster should contain elements of mostly one class
    - Also each class should ideally be assigned to a single cluster
- This does not always make sense
    - Clustering is not the same as classification
- But this is what people use most of the time

Measures

- m = number of points
- m_i = points in cluster i
- m_j = points in class j
- m_{ij}= points in cluster i coming from class j
- $p_{ij} = m_{ij}/m_i$= prob of element from class j in cluster i

- Entropy:
  - Of a cluster i: $e_i = -\sum_{j=1}^{L} p_{ij} \log p_{ij}$
    - Highest when uniform, zero when single class
  - Of a clustering: $e = \sum_{i=1}^{K} \frac{m_i}{m} e_i$
- Purity:
  - Of a cluster i: $p_i = \max_j p_{ij}$
  - Of a clustering: $purity = \sum_{i=1}^{K} \frac{m_i}{m} p_i$

Measures

- Precision:
  - Of cluster i with respect to class j: $Prec(i,j) = p_{ij}$
- Recall:
  - Of cluster I with respect to class j: $Rec(i,j) = \frac{m_{ij}}{m_j}$
- F-measure:
  - Harmonic Mean of Precision and Recall:

$$F(i,j) = \frac{2 * Prec(i,j) * Rec(i,j)}{Prec(i,j) + Rec(i,j)}$$

External Measures of Cluster Validity: Entropy and Purity

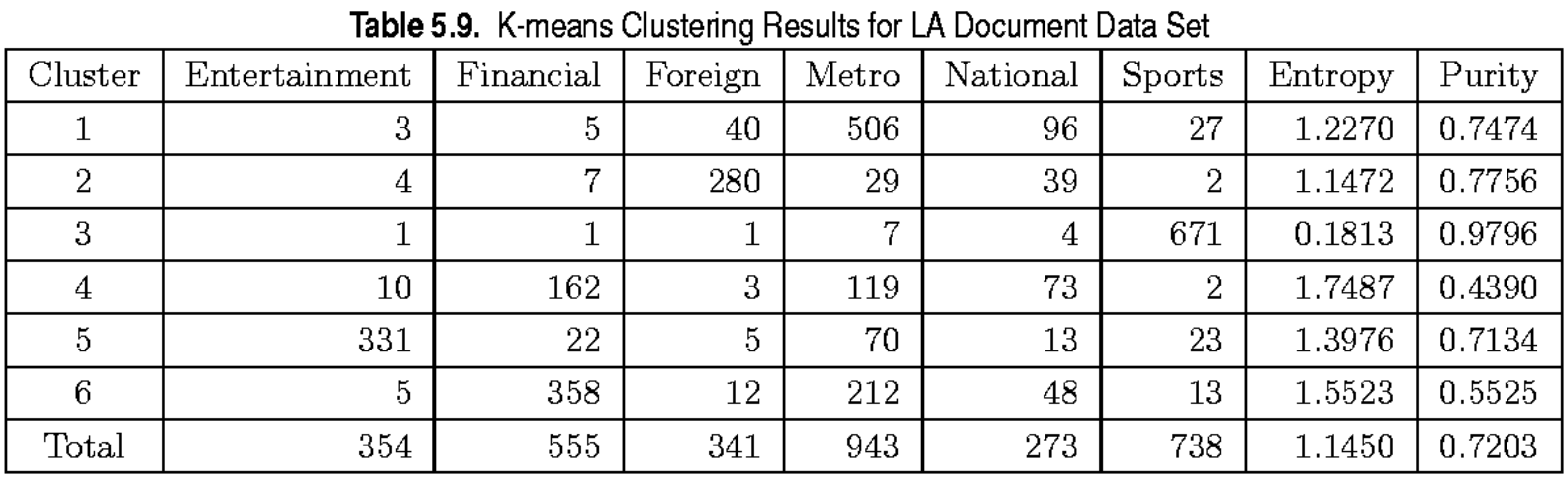
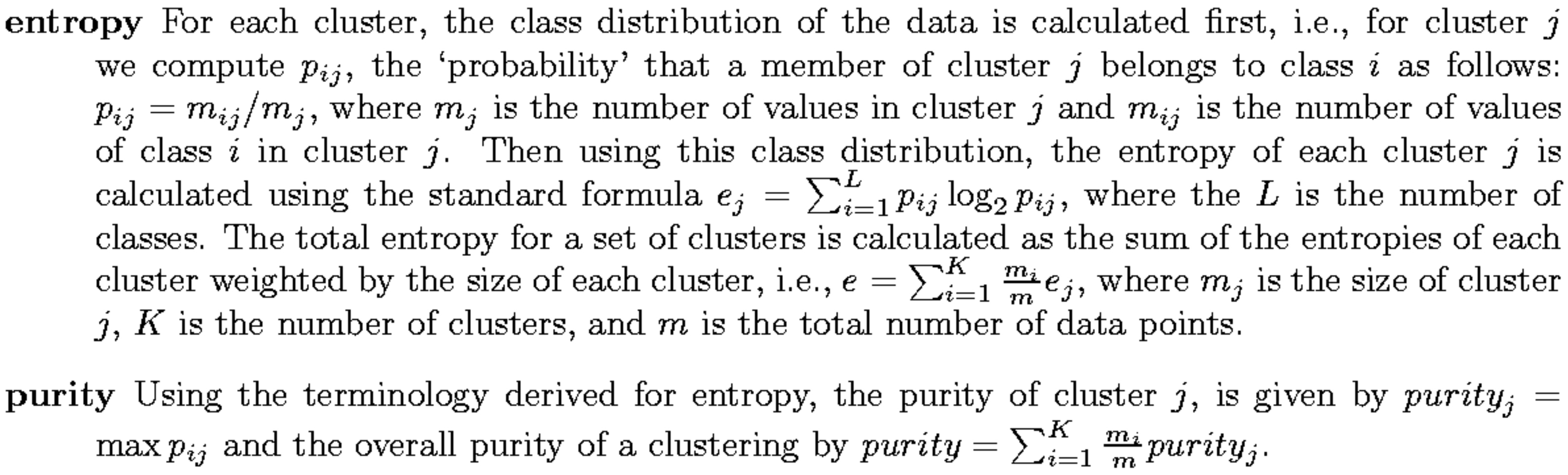


Table 5.9. K-means Clustering Results for LA Document Data Set

Cluster	Entertainment	Financial	Foreign	Metro	National	Sports	Entropy	Purity
1	3	5	40	506	96	27	1.2270	0.7474
2	4	7	280	29	39	2	1.1472	0.7756
3	1	1	1	7	4	671	0.1813	0.9796
4	10	162	3	119	73	2	1.7487	0.4390
5	331	22	5	70	13	23	1.3976	0.7134
6	5	358	12	212	48	13	1.5523	0.5525
Total	354	555	341	943	273	738	1.1450	0.7203



entropy For each cluster, the class distribution of the data is calculated first, i.e., for cluster j we compute p_{ij}, the 'probability' that a member of cluster j belongs to class i as follows: $p_{ij} = m_{ij}/m_j$, where m_j is the number of values in cluster j and m_{ij} is the number of values of class i in cluster j. Then using this class distribution, the entropy of each cluster j is calculated using the standard formula $e_j = \sum_{i=1}^{L} p_{ij} \log_2 p_{ij}$, where the L is the number of classes. The total entropy for a set of clusters is calculated as the sum of the entropies of each cluster weighted by the size of each cluster, i.e., $e = \sum_{i=1}^{K} \frac{m_i}{m} e_j$, where m_j is the size of cluster j, K is the number of clusters, and m is the total number of data points.

purity Using the terminology derived for entropy, the purity of cluster j, is given by $purity_j = \max p_{ij}$ and the overall purity of a clustering by $purity = \sum_{i=1}^{K} \frac{m_i}{m} purity_j$.

Final Comment on Cluster Validity

"The validation of clustering structures is the most difficult and frustrating part of cluster analysis.

Without a strong effort in this direction, cluster analysis will remain a black art accessible only to those true believers who have experience and great courage."

Algorithms for Clustering Data, Jain and Dubes