Flat Clustering – K-Means Algorithm

1. Purpose. Clustering algorithms group a set of documents into subsets or clusters. The cluster algorithms' goal is to create clusters that are coherent internally, but clearly different from each other. In other words, documents within a cluster should be as similar as possible; and documents in one cluster should be as dissimilar as possible from documents in other clusters.

Clustering is the most common form of unsupervised learning. No supervision means that there is no human expert who has assigned documents to classes. In clustering, it is the distribution and makeup of the data that will determine cluster membership. An example is in figure 1. It is visually clear that there are three distinct clusters of points. The difference between clustering and classification may not seem great at first. After all, in both cases we have a partition of a set of documents into groups. But as we will see that problems are fundamentally different. Classification is a form of supervised learning. Our goal is to replicate a categorical distinction that a human supervisor imposes on the data. In unsupervised learning, of which clustering is the most important example, we have no such teacher to guide us.

The key input to a clustering algorithm is the distance measure. In Figure 1, the distance measure is distance in the two-dimensional(2D) plane. This measure suggests three different clusters in the figure. In document clustering, the distance measure is often Euclidean distance. Different distance
measures give rise to different clusterings. Thus, the distance measure is an important means by which we can influence the outcome of clustering. Flat clustering creates a flat set of clusters without any explicit structure that flat clustering would relate clusters to each other.

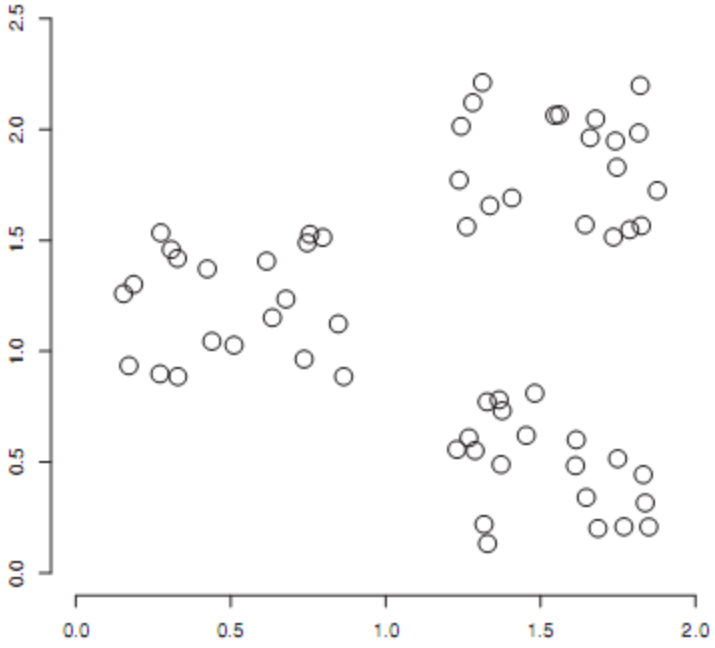

Figure 1: An example of a data set with a clear cluster structure.

A second important distinction can be made between hard and soft clustering

algorithms. Hard clustering computes a hard assignment –each document is a member of exactly one cluster. The assignment of soft clustering algorithms is soft – a document's assignment is a distribution over all clusters.
In a soft assignment, a document has fractional membership in several clusters.

Latent semantic indexing, a form of dimensionality reduction, is a soft
clustering algorithm. This laboratory motivates the use of clustering in information retrieval by introducing a number of applications, defines the problem we are trying to solve in clustering, and discusses measures for
evaluating cluster quality. It then describes the K means flat clustering algorithm, ,and the expectation maximization (or EM) algorithm, a soft clustering algorithm. K-means is perhaps the most widely used flat clustering algorithm because of its simplicity and efficiency. The EM algorithm is a generalization
of K-means and can be applied to a large variety of document representations and distributions.

2. Problem statement

We can define the goal in hard flat clustering as follows. Given

(i) a set of documents D ={d1, … ,dN},
(ii) a desired number of clusters K
(iii) an objective function that evaluates the quality of a clustering we want to compute an assignment γ : D□{1, … ,K} that minimizes (or, in other cases,

maximizes) the objective function. In most cases, we also demand that γ is
surjective, that is, that none of the K clusters is empty. The objective function is often defined in terms of similarity or distance between documents. Below, we will see that the objective in K-means clustering is to minimize the average distance between documents and their centroids or, equivalently, to maximize the similarity between documents and their centroids. We use both similarity and distance to talk about relatedness between documents.

For documents, the type of similarity we want is usually topic similarity or high values on the same dimensions in the vector space model. For example, documents about China have high values on dimensions like Chinese, Beijing, and Mao whereas documents about the UK tend to have high values for London, Britain,and Queen. We approximate topic similarity with cosine similarity or Euclidean distance in vector space. If we intend to capture similarity of a type other than topic, for example, similarity of language, then a different representation may be appropriate. When computing topic similarity, stop word scan be safely ignored, but they are important cues for separating clusters of English (in which 'the' occurs frequently and 'la' infrequently) and French documents (in which 'the' occurs infrequently and 'la' frequently).

A difficult issue in clustering is determining the number of clusters or cardinality of a clustering, which we denote by K. Often K is not hing more than a good guess based on experience or domain knowledge. But for K-means, we will also introduce a heuristic method for choosing K and an attempt to incorporate the selection of K into the objective function. Sometimes the application puts constraints on the range of K. For example, the scatter-gather interface in Figure16.3 could not display more than about K = 10 clusters per layer because of the size and resolution of computer monitors in the early 1990s.

Because our goal is to optimize an objective function, clustering is essentially a search

problem. The brute force solution would be to enumerate
all possible clusterings and pick the best. However, there are exponentially many partitions, so this approach is not feasible.

For this reason, most flat clustering algorithms refine an initial partition ingiteratively. If the search starts at an unfavorable initial point, we may miss the global optimum. Finding a good starting point is therefore another important problem we have to solve in flat clustering.

3. K-means

K-means is the most important flat clustering algorithm. Its objective is to minimize the average squared Euclidean distance of documents from their cluster centers where a cluster center is defined as the mean or centroid # μ of the documents in a cluster ω: centroid

$$\vec{\mu}(\omega) = \frac{1}{|\omega|} \sum_{\vec{x} \in \omega} \vec{x}.$$

The definition assumes that documents are represented as length normalized vectors in a real valued space in the familiar way. The ideal clustering K-means is a sphere with the centroid as its center of gravity. Ideally, the clusters should not overlap. Our desiderata for training set in clustering for which we know which documents should be in the same cluster classes in Rocchio classification were the same. The difference is that we have no labeled

K-MEANS($\{\vec{x}_1, \ldots, \vec{x}_N\}$, K)
1 $(\vec{s}_1, \vec{s}_2, \ldots, \vec{s}_K) \leftarrow$ SELECTRANDOMSEEDS($\{\vec{x}_1, \ldots, \vec{x}_N\}$, K)
2 **for** $k \leftarrow 1$ **to** K
3 **do** $\vec{\mu}_k \leftarrow \vec{s}_k$
4 **while** stopping criterion has not been met
5 **do for** $k \leftarrow 1$ **to** K
6 **do** $\omega_k \leftarrow \{\}$
7 **for** $n \leftarrow 1$ **to** N
8 **do** $j \leftarrow \arg\min_{j'} |\vec{\mu}_{j'} - \vec{x}_n|$
9 $\omega_j \leftarrow \omega_j \cup \{\vec{x}_n\}$ *(reassignment of vectors)*
10 **for** $k \leftarrow 1$ **to** K
11 **do** $\vec{\mu}_k \leftarrow \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} \vec{x}$ *(recomputation of centroids)*
12 **return** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

A measure of how well the centroids represent the members of their clusters is the residual sum of squares or RSS, the squared distance of each vector residual sum of squares from its centroids summed over all vectors:

$$\mathrm{RSS}_k = \sum_{\vec{x} \in \omega_k} |\vec{x} - \vec{\mu}(\omega_k)|^2$$

$$\mathrm{RSS} = \sum_{k=1}^{K} \mathrm{RSS}_k$$

RSS is the objective function in K-means and our goal is to minimize it. Because N is fixed, minimizing RSS is equivalent to minimizing the average squared distance, a measure of how well centroids represent their documents.

The first step of K-means is to select as initial cluster centers K randomly selected documents, the seeds. The algorithm then moves the cluster centers seed around in space to minimize RSS. As shown in Figure16.5, this is done iteratively by repeating two steps until a stopping criterion is met: Reassigning documents to the cluster with the closest centroid and recomputing each centroid based on the current members of its cluster. Figure16.6 shows snapshots from nine iterations of the K-means algorithm for a set of points. The “centroid” column of Table17.2 (page364) shows examples of centroids. We can apply one of the following termination conditions.

A fixed number of iterations I has been completed. This condition limits
The runtime of the clustering algorithm, but in some cases the quality of the clustering will be poor because of an insufficient number of iterations.

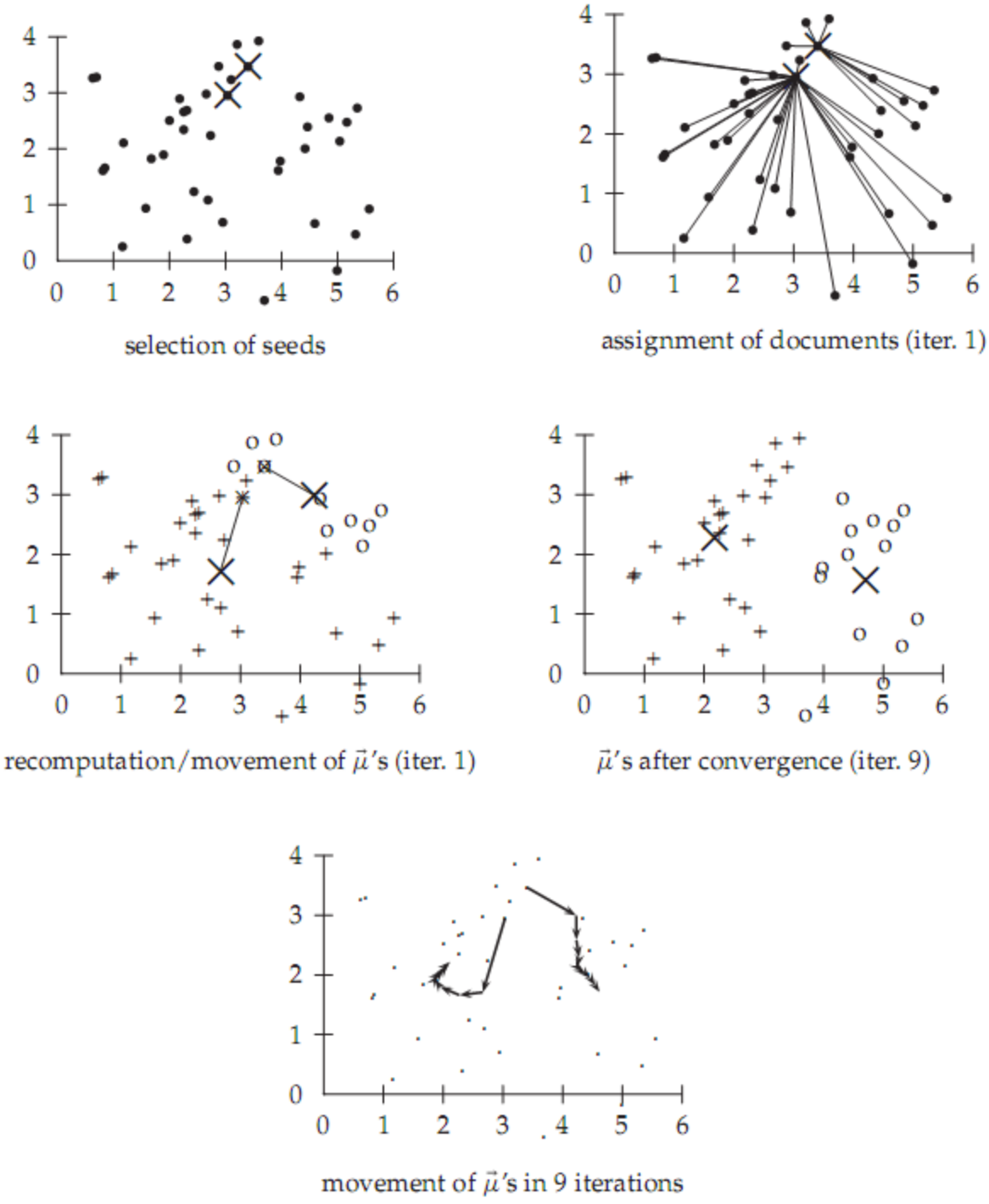

Figure 16.6 A K-means example for $K = 2$ in $\mathbb{R}^2$. The position of the two centroids ($\vec{\mu}$'s shown as X's in the top four panels) converges after nine iterations.

Assignment of documents to clusters (the partitioning function γ) does not change between iterations. Except for cases with a bad local minimum, this produces a good clustering, but run-time may be unacceptably long.

Terminate when the decrease in RSS falls below a threshold θ. For small θ, this indicates that we are close to convergence. Again, we need to combine it with a bound on the

number of iterations to prevent very long run-times. We now show that K-means converges by proving that RSS monotonically decreases in each iteration. We will use decrease in the meaning decrease or does not change in this section. First, RSS decreases in the reassignment step; each vector is assigned to the closest centroid, so the distance it contributes to RSS decreases. Second, it decreases in the re-computation step because the new centroid is the vector v for which RSSk reaches its minimum.

$$\mathrm{RSS}_k(\vec{v}) = \sum_{\vec{x} \in \omega_k} |\vec{v} - \vec{x}|^2 = \sum_{\vec{x} \in \omega_k} \sum_{m=1}^{M} (v_m - x_m)^2$$

$$\frac{\partial \mathrm{RSS}_k(\vec{v})}{\partial v_m} = \sum_{\vec{x} \in \omega_k} 2(v_m - x_m)$$

where xm and vm are the mth components of their respective vectors. Setting the partial derivative to zero, we get:

$$v_m = \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} x_m$$

which is the component wise definition of the centroid. Thus, we minimize RSSk when the old centroid is replaced with the new centroid. RSS, the sum of the RSSk, must then also decrease during recomputation. Because there is only a finite set of possible clusterings, a monotonically decreasing algorithm will eventually arrive at a (local) minimum. Take care, however, to break ties consistently, for example, by assigning a document to the cluster with the lowest index if there are several equidistant centroids. Otherwise, the algorithm can cycle forever in a loop of clusterings that have the same cost. Although this proves the convergence of K-means, there is unfortunately no guarantee that a global minimum in the objective function will be reached.

This is a particular problem if a document set contains many outliers, documents that are far from any other documents and therefore do not fit well
into any cluster. Frequently, if an outlier is chosen as an initial seed, then no
other vector is assigned to it during subsequent iterations. Thus, we end up
with a singleton cluster (a cluster with only one document) even though there singleton
cluster is probably a clustering with lower RSS.

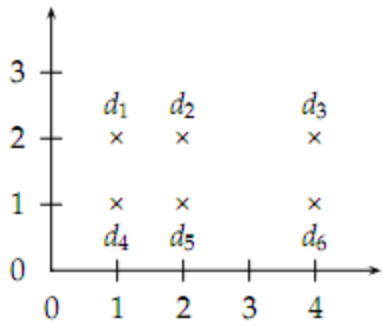

Figure 3: The outcome of clustering in K-means depends on the initial seeds. For seeds d2 and d5, K-means converges to {{d1,d2,d3}, {d4,d5,d6}},a suboptimal clustering. For seeds d2 and d3, it converges to {{d1,d2,d4,d5}, {d3,d6}},the global optimum for K = 2.

Effective heuristics for seed selection include (i) excluding outliers from the seed set;(ii)trying out multiple starting points and choosing the clustering with lowest cost; and (iii) obtaining seeds from another method such as hierarchical clustering. Because deterministic hierarchical clustering methods are more predictable than K-means, a hierarchical clustering of a small random sample of size iK (e.g., for i = 5 or i = 10)often provides good seeds. Other initialization methods compute seeds that are not selected from the vectors to be clustered. A robust method that works well for a large variety of document distributions is to select i (e.g., i = 10) random vectors for each cluster and use their centroid as the seed for this cluster.

What is the time complexity of K-means? Most of the time is spent on computing vector distances. One such operation costs #(M). The reassignment step computes KN distances, so its overall complexity is #(KNM). In the re-computation step, each vector gets added to a centroid once, so the complexity of this step is #(NM). For a fixed number of iterations I, the overall complexity is therefore #(IKNM).Thus, K-means is linear in all relevant factors: iterations, number of clusters, number of vectors, and dimensionality of the space. This means that K-means is more efficient than the hierarchical algorithms. We had to fix the number of iterations I, which can be tricky in practice. But in most cases, K-means quickly reaches either complete convergence or a clustering that is close to convergence. In the latter case, a few documents would switch membership if further iterations were computed, but this has a small effect on the overall quality of the clustering.

There is one subtlety in the preceding argument. Even a linear algorithm can be quite slow if one of the arguments of #(...) is large, and M usually is large. High dimensionality is not a problem for computing the distance of two documents. Their vectors are sparse, so that only a small fraction
of the theoretically possible M component wise differences need to be computed.

Centroids, however, are dense; they pool all terms that occur in any of the documents of their clusters. As a result, distance computations are time consuming in a naive implementation of K-means. But there are simple and effective heuristics for making centroid–document similarities as fast to compute as document document similarities. Truncating centroids to the most significant k terms (e.g., k = 1,000) hardly decreases cluster quality while achieving a significant speedup of the reassignment step.

The same efficiency problem is addressed by K-medoids, a variant of K-means that computes medoids instead of centroids as cluster centers. We define the medoid of a cluster as the document vector that is closest to the medoid centroid. Since medoids are sparse document vectors, distance computations are fast.

4. Working example using Weka

1. Under the Process tab in Experimenter window, press Open File;
2. Select <path_to_weka>/data/weather.arff or any other input data
3. The attributes and their possible values for the relation weather are as follows:

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

4. Under the Cluster tab, tick off the "Use training set combo box" and press start.
5. The output of the clustering is as follows:

```
Number of iterations: 3
Within cluster sum of squared errors: 16.237456311387238
Missing values globally replaced with mean/mode

Cluster centroids:
                           Cluster#
Attribute      Full Data          0          1
                    (14)        (9)        (5)
=============================================
outlook            sunny      sunny   overcast
temperature      73.5714    75.8889       69.4
humidity         81.6429    84.1111       77.2
windy              FALSE      FALSE       TRUE
play                 yes        yes        yes
```

Hereafter, we have 2 clusters. The fist cluster's centroid is represented by the vector on the (0) column, the second cluster is represented by the vector on the last column, and the mean of all the data is presented on the first column.

5. Real world clustering examples

5.1 Clustering of wines

Being given a database containing categories of wine, described by their properties, we should be able to create some clusters of wine categories, splitting them by major characteristics.

Type	Alcohol	Malic_Acid	Ash	Ash_Alcalinity	Magnesium	Total_Phenols
A	14.23	1.71	2.43	15.6	127	2.8
A	13.2	1.78	2.14	11.2	100	2.65
A	13.16	2.36	2.67	18.6	101	2.8
A	14.37	1.95	2.5	16.8	113	3.85
A	13.24	2.59	2.87	21	118	2.8
A	14.2	1.76	2.45	15.2	112	3.27
A	14.39	1.87	2.45	14.6	96	2.5
A	14.06	2.15	2.61	17.6	121	2.6
A	14.83	1.64	2.17	14	97	2.8
A	13.86	1.35	2.27	16	98	2.98
A	14.1	2.16	2.3	18	105	2.95
A	14.12	1.48	2.32	16.8	95	2.2
A	13.75	1.73	2.41	16	89	2.6

http://www.resample.com/xlminer/help/kMClst/KMClust_ex.htm

5.2 Clustering of students in a group

Being given a database of students from a group, we should be able to create 4 clusters named : Weak, Normal, Smart, Outstanding based on their grades on different disciplines. After the clustering is done, we can calculate the distance between a student and a cluster centroid

6. Assignments

6.1 Being given the following relation: Student(Name, gradeMath, gradeProgramming, gradePhysics, gradeEnglish, gradeOverall), create an arff file containing at least 15 instances, load it into Weka, and apply k-Means clustering to it. Also cluster the instances without Weka, and compare the results. Pick different initial cluster centroids and compare the results.

6.2 Also create an arff file according to the table with wine instances, load it into Weka and see the results after applying k-Means clustering.

6.3 *Develop a C program that clusters a planar set P of m=3k points into k triangles such that the sum of all triangle circumferences is minimized.

6.4 **Develop a C program that clusters m = nk points on a line into k clusters of equal size, i.e. a balanced clustering, with a minimum sum of all distances between points of the same subset consists of k disjoint segments of the line each containing n points.