16 *Flat clustering*

CLUSTER Clustering algorithms group a set of documents into subsets or *clusters*. The 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.

UNSUPERVISED LEARNING 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. A simple example is Figure 16.1. It is visually clear that there are three distinct clusters of points. This chapter and Chapter 17 introduce algorithms that find such clusters in an unsupervised fashion.

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 the two problems are fundamentally different. Classification is a form of supervised learning (Chapter 13, page 237): 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 16.1, 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 *Flat clustering* creates a flat set of clusters without any explicit structure that would relate clusters to each other. *Hierarchical clustering* creates a hierarchy of clusters and will be covered in Chapter 17. Chapter 17 also addresses the difficult problem of labeling clusters automatically.

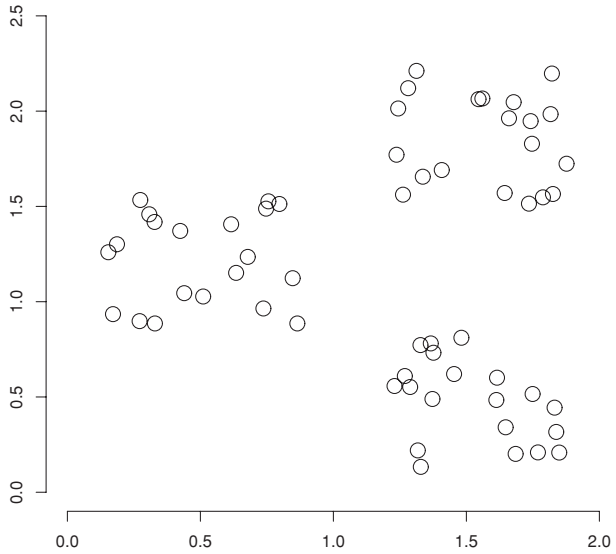

Figure 16.1 An example of a data set with a clear cluster structure.

HARD CLUSTERING

SOFT CLUSTERING

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 (Chapter 18, page 382).

This chapter motivates the use of clustering in information retrieval by introducing a number of applications (Section 16.1), defines the problem we are trying to solve in clustering (Section 16.2), and discusses measures for evaluating cluster quality (Section 16.3). It then describes two flat clustering algorithms, K-means (Section 16.4), a hard clustering algorithm, and the expectation maximization (or EM) algorithm (Section 16.5), 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.

16.1 Clustering in information retrieval

CLUSTER HYPOTHESIS

The *cluster hypothesis* states the fundamental assumption we make when using clustering in information retrieval (IR).

Cluster hypothesis. Documents in the same cluster behave similarly with respect to relevance to information needs.

Table 16.1 Some applications of clustering in IR.

application	What is clustered?	benefit	example
search result clustering	search results	more effective information presentation to user	Figure 16.2
scatter-gather	(subsets of) collection	alternative user interface: "search without typing"	Figure 16.3
collection clustering	collection	effective information presentation for exploratory browsing	McKeown et al. (2002), http://news.google.com
language modeling	collection	increased precision and/or recall	Liu and Croft (2004)
cluster-based retrieval	collection	higher efficiency: faster search	Salton (1971a)

The hypothesis states that if there is a document from a cluster that is relevant to a search request, then it is likely that other documents from the same cluster are also relevant. This is because clustering puts together documents that share many terms. The cluster hypothesis essentially is the contiguity hypothesis in Chapter 14 (page 266). In both cases, we posit that similar documents behave similarly with respect to relevance.

Table 16.1 shows some of the main applications of clustering in IR. They differ in the set of documents that they cluster – search results, collection, or subsets of the collection – and the aspect of an IR system they try to improve – user experience, user interface, effectiveness or efficiency of the search system. But they are all based on the basic assumption stated by the cluster hypothesis.

SEARCH RESULT CLUSTERING

The first application mentioned in Table 16.1 is *search result clustering* where by search results we mean the documents that were returned in response to a query. The default presentation of search results in IR is a simple list. Users scan the list from top to bottom until they have found the information they are looking for. Instead, search result clustering clusters the search results, so that similar documents appear together. It is often easier to scan a few coherent groups than many individual documents. This is particularly useful if a search term has different word senses. The example in Figure 16.2 is jaguar. Three frequent senses on the web refer to the car, the animal, and an Apple operating system. The *Clustered Results* panel returned by the Vivísimo search engine (http://vivisimo.com) can be a more effective user interface for understanding what is in the search results than a simple list of documents.

SCATTER-GATHER

A better user interface is also the goal of *scatter-gather*, the second application in Table 16.1. Scatter-gather clusters the whole collection to get groups of documents that the user can select or *gather*. The selected groups are merged and the resulting set is again clustered. This process is repeated until a cluster of interest is found. An example is shown in Figure 16.3.

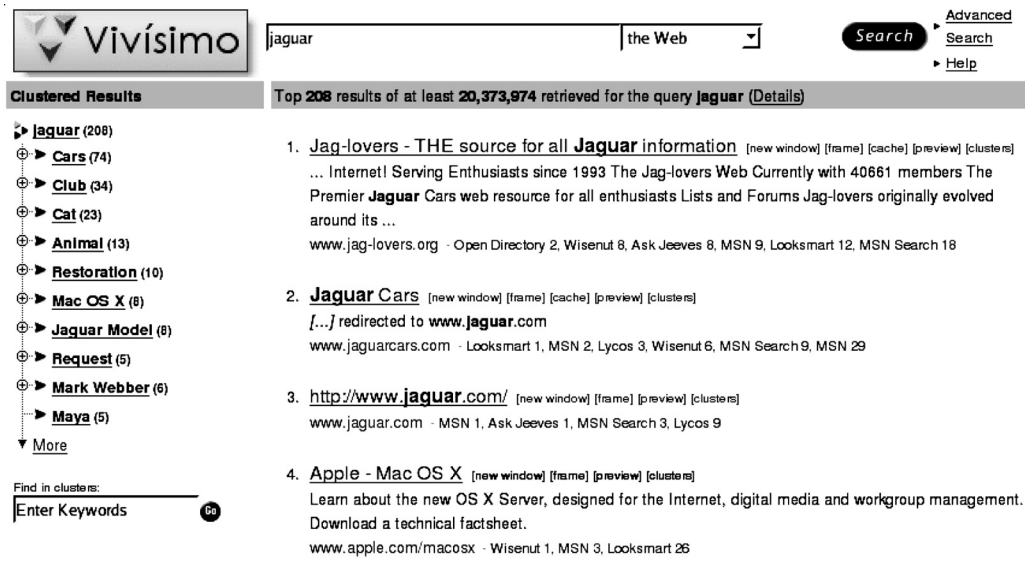

Figure 16.2 Clustering of search results to improve recall. None of the top hits cover the animal sense of jaguar, but users can easily access it by clicking on the *Cat Cluster* in the *Clustered Results* panel on the left (third arrow from the top).

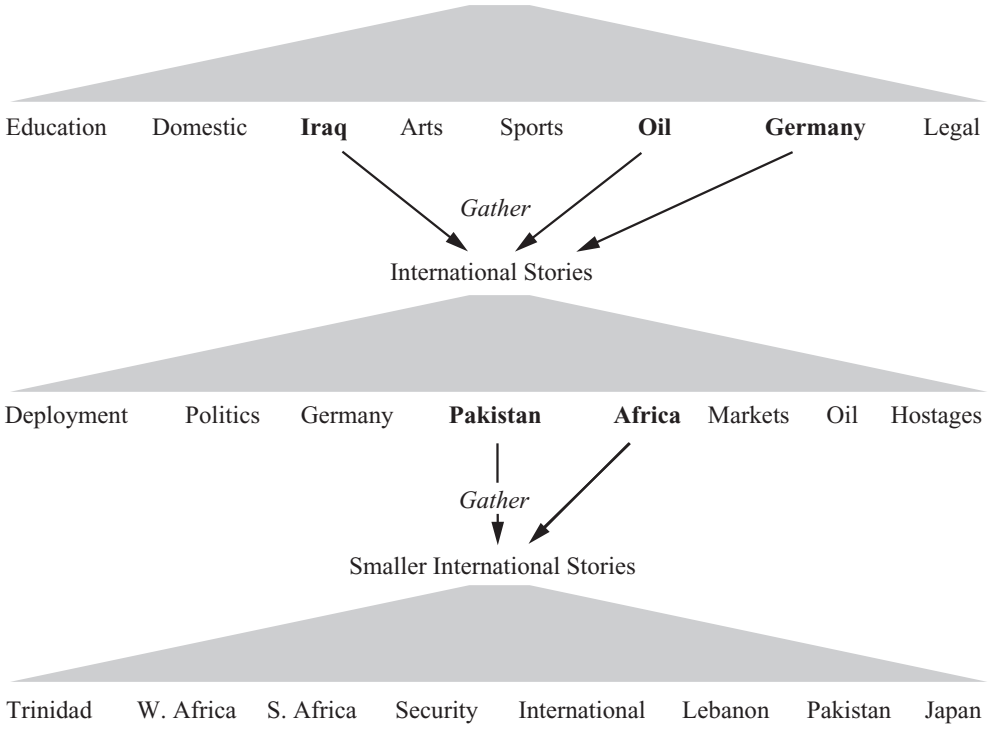

Figure 16.3 An example of a user session in scatter-gather. A collection of *New York Times* news stories is clustered ("scattered") into eight clusters (*top row*). The user manually *gathers* three of these into a smaller collection *International Stories* and performs another scattering operation. This process repeats until a small cluster with *relevant* documents is found (e.g., *Trinidad*).

Automatically generated clusters like those in Figure 16.3 are not as neatly organized as a manually constructed hierarchical tree like the Open Directory at http://dmoz.org. Also, finding descriptive labels for clusters automatically is a difficult problem (Section 17.7, page 363). But cluster-based navigation is an interesting alternative to keyword searching, the standard IR paradigm. This is especially true in scenarios where users prefer browsing over searching because they are unsure about which search terms to use.

As an alternative to the user-mediated iterative clustering in scatter-gather, we can also compute a static hierarchical clustering of a collection that is not influenced by user interactions ("Collection clustering" in Table 16.1). Google News and its precursor, the Columbia NewsBlaster system, are examples of this approach. In the case of news, we need to frequently recompute the clustering to make sure that users can access the latest breaking stories. Clustering is well suited for access to a collection of news stories because news reading is not really search, but rather a process of selecting a subset of stories about recent events.

The fourth application of clustering exploits the cluster hypothesis directly for improving search results, based on a clustering of the entire collection. We use a standard inverted index to identify an initial set of documents that match the query, but we then add other documents from the same clusters even if they have low similarity to the query. For example, if the query is car and several car documents are taken from a cluster of automobile documents, then we can add documents from this cluster that use terms other than car (automobile, vehicle, etc.). This can increase recall; a group of documents with high mutual similarity is often relevant as a whole.

More recently, this idea has been used for language modeling. Equation (12.10), page 226, showed that to avoid sparse data problems in the language modeling approach to IR, the model of document d can be interpolated with a collection model. But the collection contains many documents with terms untypical of d. By replacing the collection model with a model derived from d's cluster, we get more accurate estimates of the occurrence probabilities of terms in d.

Clustering can also speed up search. As we saw in Section 6.3.2 (page 113), search in the vector space model amounts to finding the nearest neighbors to the query. The inverted index supports fast nearest-neighbor search for the standard IR setting. However, sometimes we may not be able to use an inverted index efficiently, for example, in latent semantic indexing (Chapter 18). In such cases, we could compute the similarity of the query to every document, but this is slow. The cluster hypothesis offers an alternative: Find the clusters that are closest to the query and only consider documents from these clusters. Within this much smaller set, we can compute similarities exhaustively and rank documents in the usual way. Because there are many fewer clusters than documents, finding the closest cluster is fast, and because the documents matching a query are all similar to each other, they

tend to be in the same clusters. Although this algorithm is inexact, the expected decrease in search quality is small. This is essentially the application of clustering that was covered in Section 7.1.6 (page 130).

? **Exercise 16.1** Define two documents as similar if they have at least two proper names like Clinton or Sarkozy in common. Give an example of an information need and two documents, for which the cluster hypothesis does *not* hold for this notion of similarity.

Exercise 16.2 Make up a simple, one-dimensional example (i.e., points on a line) with two clusters where the inexactness of cluster-based retrieval shows up. In your example, retrieving clusters close to the query should do worse than direct nearest neighbor search.

16.2 Problem statement

We can define the goal in hard flat clustering as follows. Given (i) a set of documents $D = \{d_1, \ldots, d_N\}$, (ii) a desired number of clusters K, and (iii) an objective function that evaluates the quality of a clustering, we want to compute an assignment $\gamma : D \rightarrow \{1, \ldots, K\}$ 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.

OBJECTIVE FUNCTION

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. The discussion of similarity measures and distance metrics in Chapter 14 (page 267) also applies to this chapter. As in Chapter 14, 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 (Chapter 6). 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 words can 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).

PARTITIONAL CLUSTERING

A note on terminology. An alternative definition of hard clustering is that a document can be a full member of more than one cluster. *Partitional*

clustering always refers to a clustering where each document belongs to exactly one cluster. (But in a partitional hierarchical clustering (Chapter 17), all members of a cluster are of course also members of its parent.) On the definition of hard clustering that permits multiple membership, the difference between soft clustering and hard clustering is that membership values in hard clustering are either 0 or 1, whereas they can take on any non-negative value in soft clustering.

EXHAUSTIVE Some researchers distinguish between *exhaustive* clusterings that assign each document to a cluster and nonexhaustive clusterings, in which some documents will be assigned to no cluster. Nonexhaustive clusterings, in which each document is a member of either no cluster or one cluster, are called *exclusive*. We define clustering to be exhaustive in this book.

16.2.1 Cardinality – The number of clusters

CARDINALITY A difficult issue in clustering is determining the number of clusters or *cardinality* of a clustering, which we denote by K. Often K is nothing 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 Figure 16.3 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.[1] For this reason, most flat clustering algorithms refine an initial partitioning iteratively. 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.

16.3 Evaluation of clustering

Typical objective functions in clustering formalize the goal of attaining high intracluster similarity (documents within a cluster are similar) and low intercluster similarity (documents from different clusters are dissimilar). This is an *internal criterion* for the quality of a clustering. But good scores on an internal criterion do not necessarily translate into good effectiveness in an

INTERNAL CRITERION OF QUALITY

[1] An upper bound on the number of clusterings is $K^N/K!$. The exact number of different partitions of N documents into K clusters is the Stirling number of the second kind. See http://mathworld.wolfram.com/StirlingNumberoftheSecondKind.html or Comtet (1974).

application. An alternative to internal criteria is direct evaluation in the application of interest. For search result clustering, we may want to measure the time it takes users to find an answer with different clustering algorithms. This is the most direct evaluation, but it is expensive, especially if large user studies are necessary.

As a surrogate for user judgments, we can use a set of classes in an evaluation benchmark or gold standard (see Section 8.5, page 151, and Section 13.6, page 258). The gold standard is ideally produced by human judges with a good level of inter-judge agreement (see Chapter 8, page 140). We can then compute an *external criterion* that evaluates how well the clustering matches the gold standard classes. For example, we may want to say that the optimal clustering of the search results for jaguar in Figure 16.2 consists of three classes corresponding to the three senses *car*, *animal*, and *operating system*. In this type of evaluation, we only use the partition provided by the gold standard, not the class labels.

EXTERNAL CRITERION OF QUALITY

This section introduces four external criteria of clustering quality. *Purity* is a simple and transparent evaluation measure. *Normalized mutual information* can be information-theoretically interpreted. The *Rand index* penalizes both false-positive and false-negative decisions during clustering. The *F measure* in addition supports differential weighting of these two types of errors.

PURITY

To compute *purity*, each cluster is assigned to the class which is most frequent in the cluster, and then the accuracy of this assignment is measured by counting the number of correctly assigned documents and dividing by N. Formally:

$$\text{purity}(\Omega, \mathbb{C}) = \frac{1}{N} \sum_k \max_j |\omega_k \cap c_j| \tag{16.1}$$

where $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ is the set of clusters and $\mathbb{C} = \{c_1, c_2, \ldots, c_J\}$ is the set of classes. We interpret ω_k as the set of documents in ω_k and c_j as the set of documents in c_j in Equation (16.1).

We present an example of how to compute purity in Figure 16.4.[2] Bad clusterings have purity values close to 0, a perfect clustering has a purity of 1. Purity is compared with the other three measures discussed in this chapter in Table 16.2.

High purity is easy to achieve when the number of clusters is large – in particular, purity is 1 if each document gets its own cluster. Thus, we cannot use purity to trade off the quality of the clustering against the number of clusters.

[2] Recall our note of caution from Figure 14.2 (page 268) when looking at this and other 2D figures in this and the following chapter: these illustrations can be misleading because 2D projections of length-normalized vectors distort similarities and distances between points.

Table 16.2 The four external evaluation measures applied to the clustering in Figure 16.4.

	purity	NMI	RI	F_5
lower bound	0.0	0.0	0.0	0.0
maximum	1.0	1.0	1.0	1.0
value for Figure 16.4	0.71	0.36	0.68	0.46

NORMALIZED MUTUAL INFORMATION

A measure that allows us to make this tradeoff is *normalized mutual information* or *NMI*:

$$\text{NMI}(\Omega, \mathbb{C}) = \frac{I(\Omega;\mathbb{C})}{[H(\Omega) + H(\mathbb{C})]/2}. \tag{16.2}$$

I is mutual information (cf. Chapter 13, page 252):

$$I(\Omega;\mathbb{C}) = \sum_k \sum_j P(\omega_k \cap c_j) \log \frac{P(\omega_k \cap c_j)}{P(\omega_k)P(c_j)} \tag{16.3}$$

$$= \sum_k \sum_j \frac{|\omega_k \cap c_j|}{N} \log \frac{N|\omega_k \cap c_j|}{|\omega_k||c_j|} \tag{16.4}$$

where $P(\omega_k)$, $P(c_j)$, and $P(\omega_k \cap c_j)$ are the probabilities of a document being in cluster ω_k, class c_j, and in the intersection of ω_k and c_j, respectively. Equation (16.4) is equivalent to Equation (16.3) for maximum likelihood estimates (MLE) of the probabilities (i.e., the estimate of each probability is the corresponding relative frequency).

H is entropy as defined in Chapter 5 (page 91):

$$H(\Omega) = -\sum_k P(\omega_k) \log P(\omega_k) \tag{16.5}$$

$$= -\sum_k \frac{|\omega_k|}{N} \log \frac{|\omega_k|}{N} \tag{16.6}$$

where, again, the second equation is based on MLEs of the probabilities.

$I(\Omega;\mathbb{C})$ in Equation (16.3) measures the amount of information by which our knowledge about the classes increases when we are told what the clusters

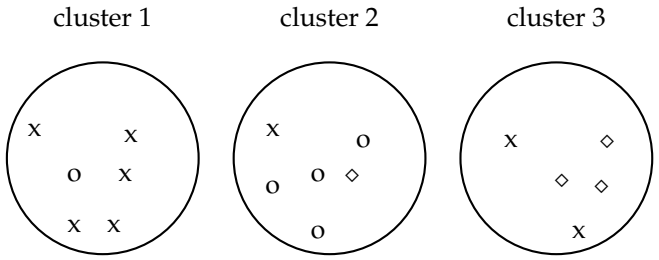

Figure 16.4 Purity as an external evaluation criterion for cluster quality. Majority class and number of members of the majority class for the three clusters are: x, 5 (cluster 1); o, 4 (cluster 2); and ◇, 3 (cluster 3). Purity is $(1/17) \times (5 + 4 + 3) \approx 0.71$.

are. The minimum of $I(\Omega;\mathbb{C})$ is 0 if the clustering is random with respect to class membership. In that case, knowing that a document is in a particular cluster does not give us any new information about what its class might be. Maximum mutual information is reached for a clustering Ω_{exact} that perfectly recreates the classes – but also if clusters in Ω_{exact} are further subdivided into smaller clusters (Exercise 16.7). In particular, a clustering with $K = N$ one-document clusters has maximum MI. So MI has the same problem as purity: It does not penalize large cardinalities and thus does not formalize our bias that, other things being equal, fewer clusters are better.

The normalization by the denominator $[H(\Omega) + H(\mathbb{C})]/2$ in Equation (16.2) fixes this problem; entropy tends to increase with the number of clusters. For example, $H(\Omega)$ reaches its maximum $\log N$ for $K = N$, which ensures that NMI is low for $K = N$. Because NMI is normalized, we can use it to compare clusterings with different numbers of clusters. The particular form of the denominator is chosen because $[H(\Omega) + H(\mathbb{C})]/2$ is a tight upper bound on $I(\Omega;\mathbb{C})$ (Exercise 16.8). Thus, NMI is always a number between 0 and 1.

An alternative to this information-theoretic interpretation of clustering is to view it as a series of decisions, one for each of the $N(N-1)/2$ pairs of documents in the collection. We want to assign two documents to the same cluster if and only if they are similar. A true-positive (TP) decision assigns two similar documents to the same cluster, a true-negative (TN) decision assigns two dissimilar documents to different clusters. There are two types of errors we can commit. A false-positive (FP) decision assigns two dissimilar documents to the same cluster. A false-negative (FN) decision assigns two similar documents to different clusters. The *Rand index (RI)* measures the RI percentage of decisions that are correct. That is, it is simply accuracy (Section 8.3, page 143).

RAND INDEX (RI)

$$\text{RI} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{FP} + \text{FN} + \text{TN}}$$

As an example, we compute RI for Figure 16.4. We first compute TP + FP. The three clusters contain six, six, and five points, respectively, so the total number of "positives" or pairs of documents that are in the same cluster is:

$$\text{TP} + \text{FP} = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40.$$

Of these, the x pairs in cluster 1, the o pairs in cluster 2, the $\diamond$ pairs in cluster 3, and the x pair in cluster 3 are true positives:

$$\text{TP} = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20.$$

Thus, $\text{FP} = 40 - 20 = 20$.

FN and TN are computed similarly, resulting in the following contingency table:

	same cluster	different clusters
same class	TP = 20	FN = 24
different classes	FP = 20	TN = 72

RI is then $(20 + 72)/(20 + 20 + 24 + 72) \approx 0.68$.

The RI gives equal weight to FPs and FNs. Separating similar documents is sometimes worse than putting pairs of dissimilar documents in the same cluster. We can use the *F measure* (Section 8.3, page 142) to penalize FNs more strongly than FPs by selecting a value $\beta > 1$, thus giving more weight to recall.

F MEASURE

$$P = \frac{\text{TP}}{\text{TP} + \text{FP}} \qquad R = \frac{\text{TP}}{\text{TP} + \text{FN}} \qquad F_\beta = \frac{(\beta^2 + 1)PR}{\beta^2 P + R}.$$

Based on the numbers in the contingency table, $P = 20/40 = 0.5$ and $R = 20/44 \approx 0.455$. This gives us $F_1 \approx 0.48$ for $\beta = 1$ and $F_5 \approx 0.456$ for $\beta = 5$. In IR, evaluating clustering with F has the advantage that the measure is already familiar to the research community.

? **Exercise 16.3** Replace every point d in Figure 16.4 with two identical copies of d in the same class. (i) Is it less difficult, equally difficult or more difficult to cluster this set of thirty-four points as opposed to the seventeen points in Figure 16.4? (ii) Compute purity, NMI, RI, and F_5 for the clustering with thirty-four points. Which measures increase and which stay the same after doubling the number of points? (iii) Given your assessment in (i) and the results in (ii), which measures are best suited to compare the quality of the two clusterings?

16.4 *K*-means

K-means is the most important flat clustering algorithm. Its objective is to minimize the average squared Euclidean distance (Chapter 6, page 121) of documents from their cluster centers where a cluster center is defined as the mean or *centroid* $\vec{\mu}$ 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. We used centroids for Rocchio classification in Chapter 14 (page 269). They play a similar role here. The ideal cluster in *K*-means is a sphere with the centroid as its center of gravity. Ideally, the clusters should not overlap. Our desiderata for classes in Rocchio classification were the same. The difference is that we have no labeled training set in clustering for which we know which documents should be in the same cluster.

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** $\{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$

Figure 16.5 The K-means algorithm. For most IR applications, the vectors $\vec{x}_n \in \mathbb{R}^M$ should be length normalized. Alternative methods of seed selection and initialization are discussed on page 334.

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 from its centroid summed over all vectors:

RESIDUAL SUM OF SQUARES

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

$$\text{RSS} = \sum_{k=1}^{K} \text{RSS}_k \tag{16.7}$$

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 around in space to minimize RSS. As shown in Figure 16.5, 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. Figure 16.6 shows snapshots from nine iterations of the K-means algorithm for a set of points. The "centroid" column of Table 17.2 (page 364) shows examples of centroids.

SEED

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.

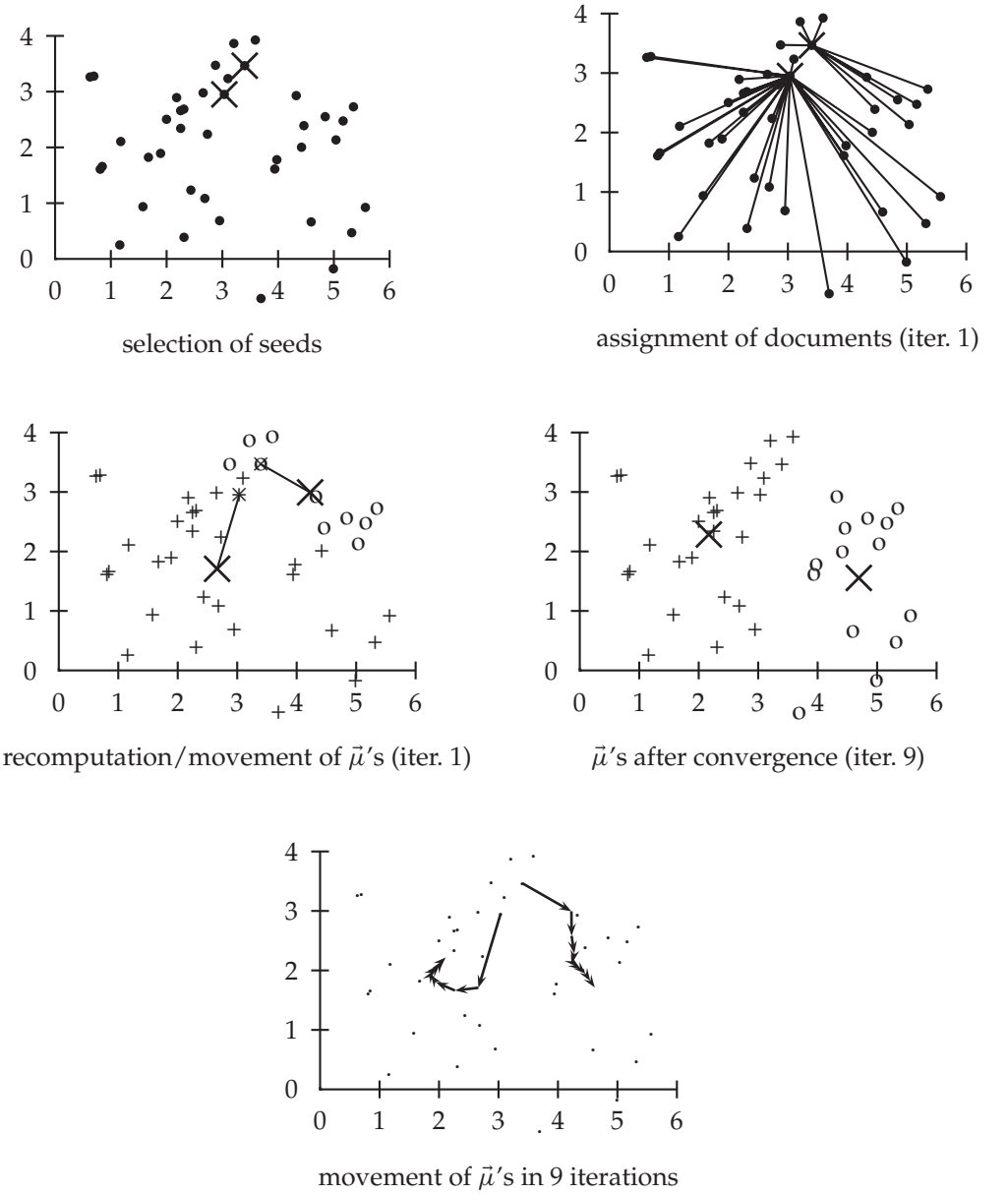

Figure 16.6 A K-means example for $K = 2$ in $\mathbb{R}^2$. The position of the two centroids ($\vec{\mu}$'s shown as X's in the top four panels) converges after nine iterations.

- 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 runtime may be unacceptably long.
- Centroids $\vec{\mu}_k$ do not change between iterations. This is equivalent to γ not changing (Exercise 16.5).

- Terminate when RSS falls below a threshold. This criterion ensures that the clustering is of a desired quality after termination. In practice, we need to combine it with a bound on the number of iterations to guarantee termination.
- 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 runtimes.

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 recomputation step because the new centroid is the vector $\vec{v}$ for which RSS_k reaches its minimum.

$$\text{RSS}_k(\vec{v}) = \sum_{\vec{x} \in \omega_k} |\vec{v} - \vec{x}|^2 = \sum_{\vec{x} \in \omega_k} \sum_{m=1}^{M} (v_m - x_m)^2 \tag{16.8}$$

$$\frac{\partial \text{RSS}_k(\vec{v})}{\partial v_m} = \sum_{\vec{x} \in \omega_k} 2(v_m - x_m) \tag{16.9}$$

where x_m and v_m are the m^{th} components of their respective vectors. Setting the partial derivative to zero, we get:

$$v_m = \frac{1}{|\omega_k|} \sum_{\vec{x} \in \omega_k} x_m \tag{16.10}$$

which is the componentwise definition of the centroid. Thus, we minimize RSS_k when the old centroid is replaced with the new centroid. RSS, the sum of the RSS_k, 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.

OUTLIER

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

SINGLETON CLUSTER

with a *singleton cluster* (a cluster with only one document) even though there is probably a clustering with lower RSS. Figure 16.7 shows an example of a suboptimal clustering resulting from a bad choice of initial seeds.

Another type of suboptimal clustering that frequently occurs is one with empty clusters (Exercise 16.11).

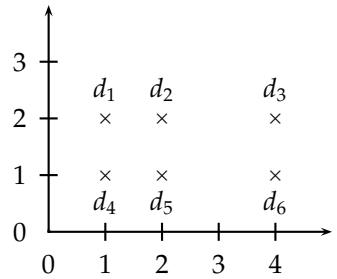

Figure 16.7 The outcome of clustering in K-means depends on the initial seeds. For seeds d_2 and d_5, K-means converges to $\{\{d_1, d_2, d_3\}, \{d_4, d_5, d_6\}\}$, a suboptimal clustering. For seeds d_2 and d_3, it converges to $\{\{d_1, d_2, d_4, d_5\}, \{d_3, d_6\}\}$, the global optimum for $K = 2$.

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 (see the description of the Buckshot algorithm, Chapter 17, page 366).

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. See Section 16.6 for more sophisticated initializations.

What is the time complexity of K-means? Most of the time is spent on computing vector distances. One such operation costs $\Theta(M)$. The reassignment step computes KN distances, so its overall complexity is $\Theta(KNM)$. In the recomputation step, each vector gets added to a centroid once, so the complexity of this step is $\Theta(NM)$. For a fixed number of iterations I, the overall complexity is therefore $\Theta(IKNM)$. 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 in Chapter 17. 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 $\Theta(\ldots)$ 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 componentwise 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 (see references in Section 16.6).

K-MEDOIDS The same efficiency problem is addressed by *K-medoids*, a variant of K-means that computes medoids instead of centroids as cluster centers. We

MEDOID define the *medoid* of a cluster as the document vector that is closest to the centroid. Since medoids are sparse document vectors, distance computations are fast.

16.4.1 Cluster cardinality in K-means

We stated in Section 16.2 that the number of clusters K is an input to most flat clustering algorithms. What do we do if we cannot come up with a plausible guess for K?

A naive approach would be to select the optimal value of K according to the objective function, namely, the value of K that minimizes RSS. Defining $\text{RSS}_{\text{min}}(K)$ as the minimal RSS of all clusterings with K clusters, we observe that $\text{RSS}_{\text{min}}(K)$ is a monotonically decreasing function in K (Exercise 16.13), which reaches its minimum 0 for $K = N$ where N is the number of documents. We would end up with each document being in its own cluster. Clearly, this is not an optimal clustering.

A heuristic method that gets around this problem is to estimate $\text{RSS}_{\text{min}}(K)$ as follows. We first perform i (e.g., $i = 10$) clusterings with K clusters (each with a different initialization) and compute the RSS of each. Then we take the minimum of the i RSS values. We denote this minimum by $\widehat{\text{RSS}}_{\text{min}}(K)$. Now we can inspect the values $\widehat{\text{RSS}}_{\text{min}}(K)$ as K increases and find the "knee" in the curve – the point where successive decreases in $\widehat{\text{RSS}}_{\text{min}}$ become noticeably smaller. There are two such points in Figure 16.8, one at $K = 4$, where the gradient flattens slightly, and a clearer flattening at $K = 9$. This is typical: There is seldom a single best number of clusters. We still need to employ an external constraint to choose from a number of possible values of K (four and nine in this case).

A second type of criterion for cluster cardinality imposes a penalty for each new cluster – where conceptually we start with a single cluster containing all documents and then search for the optimal number of clusters K by successively increasing K. To determine the cluster cardinality in this way, we create a generalized objective function that combines two elements:

DISTORTION *distortion*, a measure of how much documents deviate from the prototype

MODEL of their clusters (e.g., RSS for K-means); and a measure of *model complexity*.

COMPLEXITY We interpret a clustering here as a model of the data. Model complexity in clustering is usually the number of clusters or a function thereof. For

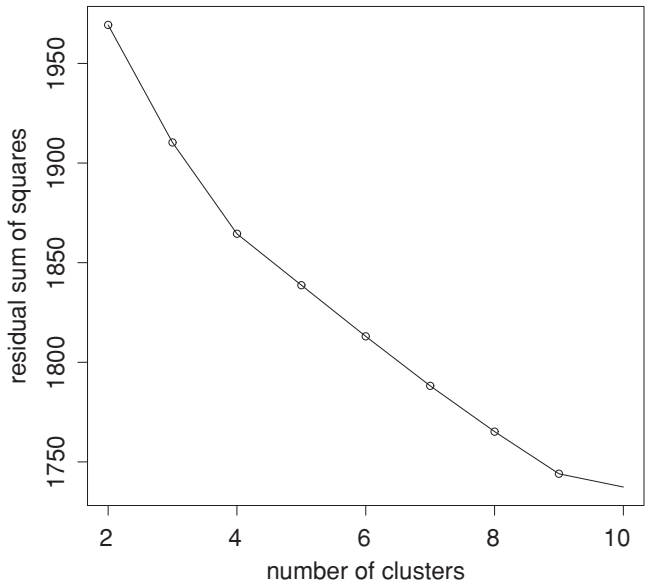

Figure 16.8 Estimated minimal residual sum of squares ($\widehat{\text{RSS}}_{\text{min}}$) as a function of the number of clusters in K-means. In this clustering of 1203 Reuters-RCV1 documents, there are two points where the $\widehat{\text{RSS}}_{\text{min}}$ curve flattens: at four clusters and at nine clusters. The documents were selected from the categories *China*, *Germany*, *Russia*, and *Sports*, so the $K = 4$ clustering is closest to the RCV1-Reuters classification.

K-means, we then get this selection criterion for K:

$$K = \arg\min_K [\text{RSS}_{\text{min}}(K) + \lambda K] \tag{16.11}$$

where λ is a weighting factor. A large value of λ favors solutions with few clusters. For $\lambda = 0$, there is no penalty for more clusters and $K = N$ is the best solution.

The obvious difficulty with Equation (16.11) is that we need to determine λ. Unless this is easier than determining K directly, then we are back to square one. In some cases, we can choose values of λ that have worked well for similar data sets in the past. For example, if we periodically cluster news stories from a newswire, there is likely to be a fixed value of λ that gives us the right K in each successive clustering. In this application, we would not be able to determine K based on past experience because K changes.

AKAIKE INFORMATION CRITERION

A theoretical justification for Equation (16.11) is the *Akaike information criterion* or AIC, an information-theoretic measure that trades off distortion against model complexity. The general form of AIC is:

$$\textbf{AIC:} \quad K = \arg\min_K [-2L(K) + 2q(K)] \tag{16.12}$$

where $-L(K)$, the negative maximum log-likelihood of the data for K clusters, is a measure of distortion and $q(K)$, the number of parameters of a model with K clusters, is a measure of model complexity. We will not attempt to derive the AIC here, but it is easy to understand intuitively. The first property of a good model of the data is that each data point is modeled well by

the model. This is the goal of low distortion. But models should also be small (i.e., have low model complexity); a model that merely describes the data (and therefore has zero distortion) is worthless. AIC provides a theoretical justification for one particular way of weighting these two factors, distortion and model complexity, when selecting a model.

For K-means, the AIC can be stated as follows:

$$\textbf{AIC:} \quad K = \arg\min_K [\text{RSS}_{\text{min}}(K) + 2MK] \tag{16.13}$$

Equation (16.13) is a special case of Equation (16.11) for $\lambda = 2M$.

To derive Equation (16.13) from Equation (16.12), observe that $q(K) = KM$ in K-means because each element of the K centroids is a parameter that can be varied independently; and that $L(K) = -(1/2)\text{RSS}_{\text{min}}(K)$ (modulo a constant) if we view the model underlying K-means as a Gaussian mixture with hard assignment, uniform cluster priors and identical spherical covariance matrices (see Exercise 16.19).

The derivation of AIC is based on a number of assumptions, for example, that the data are independent and identically distributed. These assumptions are only approximately true for data sets in information retrieval. As a consequence, the AIC can rarely be applied without modification in text clustering. In Figure 16.8, the dimensionality of the vector space is $M \approx 50{,}000$. Thus, $2\text{MK} > 50{,}000$ dominates the smaller RSS-based term ($\widehat{\text{RSS}}_{\text{min}}(1) < 5000$, not shown in the figure) and the minimum of the expression is reached for $K = 1$. But as we know, $K = 4$ (corresponding to the four classes *China*, *Germany*, *Russia*, and *Sports*) is a better choice than $K = 1$. In practice, Equation (16.11) is often more useful than Equation (16.13) – with the caveat that we need to come up with an estimate for λ.

Exercise 16.4 Why are documents that do not use the same term for the concept *car* likely to end up in the same cluster in K-means clustering?

Exercise 16.5 Two of the possible termination conditions for K-means were (i) assignment does not change, (ii) centroids do not change (page 332). Do these two conditions imply each other?

16.5 Model-based clustering

In this section, we describe a generalization of K-means, the EM algorithm. It can be applied to a larger variety of document representations and distributions than K-means.

In K-means, we attempt to find centroids that are good representatives. We can view the set of K centroids as a model that generates the data. Generating a document in this model consists of first picking a centroid at random and then adding some noise. If the noise is normally distributed, this procedure will result in clusters of spherical shape. *Model-based clustering* assumes that

MODEL-BASED CLUSTERING

the data were generated by a model and tries to recover the original model from the data. The model that we recover from the data then defines clusters and an assignment of documents to clusters.

A commonly used criterion for estimating the model parameters is maximum likelihood. In K-means, the quantity exp(−RSS) is proportional to the likelihood that a particular model (i.e., a set of centroids) generated the data. For K-means, maximum likelihood and minimal RSS are equivalent criteria. We denote the model parameters by Θ. In K-means, $\Theta = \{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$.

More generally, the maximum likelihood criterion is to select the parameters Θ that maximize the log-likelihood of generating the data D:

$$\Theta = \arg\max_{\Theta} L(D|\Theta) = \arg\max_{\Theta} \log \prod_{n=1}^{N} P(d_n|\Theta) = \arg\max_{\Theta} \sum_{n=1}^{N} \log P(d_n|\Theta).$$

$L(D|\Theta)$ is the objective function that measures the goodness of the clustering. Given two clusterings with the same number of clusters, we prefer the one with higher $L(D|\Theta)$.

This is the same approach taken in Chapter 12 (page 218) for language modeling and in Section 13.1 (page 245) for text classification. In text classification, we chose the class that maximizes the likelihood of generating a particular document. Here, we choose the clustering Θ that maximizes the likelihood of generating a given set of documents. Once we have Θ, we can compute an assignment probability $P(d|\omega_k; \Theta)$ for each document–cluster pair. This set of assignment probabilities defines a soft clustering.

An example of a soft assignment is that a document about Chinese cars may have a fractional membership of 0.5 in each of the two clusters *China* and *automobiles*, reflecting the fact that both topics are pertinent. A hard clustering like K-means cannot model this simultaneous relevance to two topics.

Model-based clustering provides a framework for incorporating our knowledge about a domain. K-means and the hierarchical algorithms in Chapter 17 make fairly rigid assumptions about the data. For example, clusters in K-means are assumed to be spheres. Model-based clustering offers more flexibility. The clustering model can be adapted to what we know about the underlying distribution of the data, be it Bernoulli (as in the example in Table 16.3), Gaussian with nonspherical variance (another model that is important in document clustering) or a member of a different family.

EXPECTATION-MAXIMIZATION ALGORITHM

A commonly used algorithm for model-based clustering is the *expectation-maximization algorithm* or *EM algorithm*. EM clustering is an iterative algorithm that maximizes $L(D|\Theta)$. EM can be applied to many different types of probabilistic modeling. We will work with a mixture of multivariate Bernoulli distributions here, the distribution we know from Section 11.3 (page 204) and Section 13.3 (page 243):

$$P(d|\omega_k; \Theta) = \left(\prod_{t_m \in d} q_{mk}\right)\left(\prod_{t_m \notin d} (1 - q_{mk})\right) \tag{16.14}$$

where $\Theta = \{\Theta_1, \ldots, \Theta_K\}$, $\Theta_k = (\alpha_k, q_{1k}, \ldots, q_{Mk})$, and $q_{mk} = P(U_m = 1|\omega_k)$ are the parameters of the model.[3] $P(U_m = 1|\omega_k)$ is the probability that a document from cluster k contains term t_m. The probability α_k is the prior of cluster ω_k: the probability that a document d is in ω_k if we have no information about d.

The mixture model then is:

$$P(d|\Theta) = \sum_{k=1}^{K} \alpha_k \left(\prod_{t_m \in d} q_{mk} \right) \left(\prod_{t_m \notin d} (1 - q_{mk}) \right). \tag{16.15}$$

In this model, we generate a document by first picking a cluster ω_k with probability α_k and then generating the terms of the document according to the parameters q_{mk}. Recall that the document representation of the multivariate Bernoulli is a vector of M Boolean values (and not a real-valued vector).

How do we use EM to infer the parameters of the clustering from the data? That is, how do we choose parameters Θ that maximize $L(D|\Theta)$? EM is similar to K-means in that it alternates between an *expectation step*, corresponding to reassignment, and a *maximization step*, corresponding to recomputation of the parameters of the model. The parameters of K-means are the centroids, the parameters of the instance of EM in this section are the α_k and q_{mk}.

EXPECTATION STEP

MAXIMIZATION STEP

The maximization step recomputes the conditional parameters q_{mk} and the priors α_k as follows:

$$\textbf{Maximization step:} \quad q_{mk} = \frac{\sum_{n=1}^{N} r_{nk} I(t_m \in d_n)}{\sum_{n=1}^{N} r_{nk}} \quad \alpha_k = \frac{\sum_{n=1}^{N} r_{nk}}{N} \tag{16.16}$$

where $I(t_m \in d_n) = 1$ if $t_m \in d_n$ and 0 otherwise and r_{nk} is the soft assignment of document d_n to cluster k as computed in the preceding iteration. (We'll address the issue of initialization in a moment.) These are the maximum likelihood estimates for the parameters of the multivariate Bernoulli from Table 13.3 (page 248) except that documents are assigned fractionally to clusters here. These maximum likelihood estimates maximize the likelihood of the data given the model.

The expectation step computes the soft assignment of documents to clusters given the current parameters q_{mk} and α_k:

$$\textbf{Expectation step:} \quad r_{nk} = \frac{\alpha_k (\prod_{t_m \in d_n} q_{mk})(\prod_{t_m \notin d_n} (1 - q_{mk}))}{\sum_{k=1}^{K} \alpha_k (\prod_{t_m \in d_n} q_{mk})(\prod_{t_m \notin d_n} (1 - q_{mk}))}. \tag{16.17}$$

This expectation step applies Equations (16.14) and (16.15) to computing the likelihood that ω_k generated document d_n. It is the classification procedure

[3] U_m is the random variable we defined in Section 13.3 (page 246) for the Bernoulli Naive Bayes model. It takes the values 1 (term t_m is present in the document) and 0 (term t_m is absent in the document).

Table 16.3 The EM clustering algorithm. The table shows a set of documents (a) and parameter values for selected iterations during EM clustering (b). Parameters shown are prior α_1, soft assignment scores $r_{n,1}$ (both omitted for cluster 2), and lexical parameters $q_{m,k}$ for a few terms. The authors initially assigned document 6 to cluster 1 and document 7 to cluster 2 (iteration 0). EM converges after 25 iterations. For smoothing, the r_{nk} in Equation (16.16) were replaced with $r_{nk} + \epsilon$ where $\epsilon = 0.0001$.

(a)

docID	document text	docID	document text
1	hot chocolate cocoa beans	7	sweet sugar
2	cocoa ghana africa	8	sugar cane brazil
3	beans harvest ghana	9	sweet sugar beet
4	cocoa butter	10	sweet cake icing
5	butter truffles	11	cake black forest
6	sweet chocolate		

(b)

	iteration of clustering							
parameter	0	1	2	3	4	5	15	25
α_1		0.50	0.45	0.53	0.57	0.58	0.54	0.45
$r_{1,1}$		1.00	1.00	1.00	1.00	1.00	1.00	1.00
$r_{2,1}$		0.50	0.79	0.99	1.00	1.00	1.00	1.00
$r_{3,1}$		0.50	0.84	1.00	1.00	1.00	1.00	1.00
$r_{4,1}$		0.50	0.75	0.94	1.00	1.00	1.00	1.00
$r_{5,1}$		0.50	0.52	0.66	0.91	1.00	1.00	1.00
$r_{6,1}$	1.00	1.00	1.00	1.00	1.00	1.00	0.83	0.00
$r_{7,1}$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
$r_{8,1}$		0.00	0.00	0.00	0.00	0.00	0.00	0.00
$r_{9,1}$		0.00	0.00	0.00	0.00	0.00	0.00	0.00
$r_{10,1}$		0.50	0.40	0.14	0.01	0.00	0.00	0.00
$r_{11,1}$		0.50	0.57	0.58	0.41	0.07	0.00	0.00
$q_{\text{africa},1}$		0.000	0.100	0.134	0.158	0.158	0.169	0.200
$q_{\text{africa},2}$		0.000	0.083	0.042	0.001	0.000	0.000	0.000
$q_{\text{brazil},1}$		0.000	0.000	0.000	0.000	0.000	0.000	0.000
$q_{\text{brazil},2}$		0.000	0.167	0.195	0.213	0.214	0.196	0.167
$q_{\text{cocoa},1}$		0.000	0.400	0.432	0.465	0.474	0.508	0.600
$q_{\text{cocoa},2}$		0.000	0.167	0.090	0.014	0.001	0.000	0.000
$q_{\text{sugar},1}$		0.000	0.000	0.000	0.000	0.000	0.000	0.000
$q_{\text{sugar},2}$		1.000	0.500	0.585	0.640	0.642	0.589	0.500
$q_{\text{sweet},1}$		1.000	0.300	0.238	0.180	0.159	0.153	0.000
$q_{\text{sweet},2}$		1.000	0.417	0.507	0.610	0.640	0.608	0.667

for the multivariate Bernoulli in Table 13.3. Thus, the expectation step is nothing else but Bernoulli Naive Bayes classification (including normalization, i.e. dividing by the denominator, to get a probability distribution over clusters).

We clustered a set of eleven documents into two clusters using EM in Table 16.3. After convergence in iteration 25, the first five documents are assigned to cluster 1 ($r_{i,1} = 1.00$) and the last six to cluster 2 ($r_{i,1} = 0.00$). Somewhat atypically, the final assignment is a hard assignment here. EM

usually converges to a soft assignment. In iteration 25, the prior α_1 for cluster 1 is $5/11 \approx 0.45$ because five of the eleven documents are in cluster 1. Some terms are quickly associated with one cluster because the initial assignment can "spread" to them unambiguously. For example, membership in cluster 2 spreads from document 7 to document 8 in the first iteration because they share sugar. ($r_{8,1} = 0$ in iteration 1.) For parameters of terms occurring in ambiguous contexts, convergence takes longer. Seed documents 6 and 7 both contain sweet. As a result, it takes 25 iterations for the term to be unambiguously associated with cluster 2 ($q_{\text{sweet},1} = 0$ in iteration 25).

Finding good seeds is even more critical for EM than for K-means. EM is prone to get stuck in local optima if the seeds are not chosen well. This is a general problem that also occurs in other applications of EM.[4] Therefore, as with K-means, the initial assignment of documents to clusters is often computed by a different algorithm. For example, a hard K-means clustering may provide the initial assignment, which EM can then "soften up."

? **Exercise 16.6** We saw above that the time complexity of K-means is $\Theta(IKNM)$. What is the time complexity of EM?

Exercise 16.7 Let Ω be a clustering that exactly reproduces a class structure $\mathbb{C}$ and Ω' a clustering that further subdivides some clusters in Ω. Show that $I(\Omega;\mathbb{C}) = I(\Omega';\mathbb{C})$.

Exercise 16.8 Show that $I(\Omega;\mathbb{C}) \leq [H(\Omega) + H(\mathbb{C})]/2$.

Exercise 16.9 Mutual information is symmetric in the sense that its value does not change if the roles of clusters and classes are switched: $I(\Omega;\mathbb{C}) = I(\mathbb{C};\Omega)$. Which of the other three evaluation measures are symmetric in this sense?

Exercise 16.10 Compute RSS for the two clusterings in Figure 16.7.

Exercise 16.11 (i) Give an example of a set of points and three initial centroids (which need not be members of the set of points) for which 3-means converges to a clustering with an empty cluster. (ii) Can a clustering with an empty cluster be the global optimum with respect to RSS?

Exercise 16.12 Download Reuters-21578. Discard documents that do not occur in one of the ten classes *acquisitions*, *corn*, *crude*, *earn*, *grain*, *interest*, *money-fx*, *ship*, *trade*, and *wheat*. Discard documents that occur in two of these ten classes. (i) Compute a K-means clustering of this subset into ten clusters. There are a number of software packages that implement K-means, such as, WEKA (Witten and Frank 2005) and R (R Development Core Team 2005). (ii) Compute purity, normalized mutual information, F_1

[4] For example, this problem is common when EM is used to estimate parameters of hidden Markov models, probabilistic grammars, and machine translation models in natural language processing (Manning and Schütze 1999).

and RI for the clustering with respect to the ten classes. (iii) Compile a confusion matrix (Table 14.5, page 283) for the ten classes and ten clusters. Identify classes that give rise to FPs and FNs.

Exercise 16.13 Prove that $\text{RSS}_{\text{min}}(K)$ is monotonically decreasing in K.

Exercise 16.14 There is a soft version of K-means that computes the fractional membership of a document in a cluster as a monotonically decreasing function of the distance Δ from its centroid, for example, as $e^{-\Delta}$. Modify reassignment and recomputation steps of hard K-means for this soft version.

Exercise 16.15 In the last iteration in Table 16.3, document 6 is in cluster 2 even though it was the initial seed for cluster 1. Why does the document change membership?

Exercise 16.16 The values of the parameters q_{mk} in iteration 25 in Table 16.3 are rounded. What are the exact values that EM will converge to?

Exercise 16.17 Perform a K-means clustering for the documents in Table 16.3. After how many iterations does K-means converge? Compare the result with the EM clustering in Table 16.3 and discuss the differences.

Exercise 16.18 [⋆ ⋆ ⋆] Modify the expectation and maximization steps of EM for a Gaussian mixture. The maximization step computes the maximum likelihood parameter estimates α_k, $\vec{\mu}_k$, and Σ_k for each of the clusters. The expectation step computes for each vector a soft assignment to clusters (Gaussians) based on their current parameters. Write down the equations for Gaussian mixtures corresponding to Equations (16.16) and (16.17).

Exercise 16.19 [⋆ ⋆ ⋆] Show that K-means can be viewed as the limiting case of EM for Gaussian mixtures if variance is very small and all covariances are 0.

WITHIN-POINT SCATTER

Exercise 16.20 [⋆ ⋆ ⋆] The *within-point scatter* of a clustering is defined as $\sum_k \frac{1}{2} \sum_{\vec{x}_i \in \omega_k} \sum_{\vec{x}_j \in \omega_k} |\vec{x}_i - \vec{x}_j|^2$. Show that minimizing RSS and minimizing within-point scatter are equivalent.

Exercise 16.21 [⋆ ⋆ ⋆] Derive an AIC criterion for the multivariate Bernoulli mixture model from Equation (16.12).

16.6 References and further reading

Berkhin (2006b) gives a general up-to-date survey of clustering methods with special attention to scalability. The classic reference for clustering in pattern recognition, covering both K-means and EM, is (Duda et al. 2000).

Rasmussen (1992) introduces clustering from an information retrieval perspective. Anderberg (1973) provides a general introduction to clustering for applications. In addition to Euclidean distance and cosine similarity, Kullback-Leibler divergence is often used in clustering as a measure of how (dis)similar documents and clusters are (Xu and Croft 1999; Muresan and Harper 2004; Kurland and Lee 2004).

The cluster hypothesis is due to Jardine and van Rijsbergen (1971) who state it as follows: *Associations between documents convey information about the relevance of documents to requests.* Salton (1971a, 1975), Croft (1978), Voorhees (1985a), Can and Ozkarahan (1990), Cacheda et al. (2003), Can et al. (2004), Singitham et al. (2004), and Altingövde et al. (2008) investigate the efficiency and effectiveness of cluster-based retrieval. Although some of these studies show improvements in effectiveness, efficiency, or both, there is no consensus that cluster-based retrieval works well consistently across scenarios. Cluster-based language modeling was pioneered by Liu and Croft (2004).

There is good evidence that clustering of search results improves user experience and search result quality (Hearst and Pedersen 1996; Zamir and Etzioni 1999; Tombros et al. 2002; Käki 2005; Toda and Kataoka 2005); although not as much as search result structuring based on carefully edited category hierarchies (Hearst 2006). The scatter-gather interface for browsing collections was presented by Cutting et al. (1992). A theoretical framework for analyzing the properties of scatter-gather and other information seeking user interfaces is presented by Pirolli (2007). Schütze and Silverstein (1997) evaluate LSI (Chapter 18) and truncated representations of centroids for efficient K-means clustering.

The Columbia NewsBlaster system (McKeown et al. 2002), a forerunner to the now much more famous and refined Google News (http://news.google.com), used hierarchical clustering (Chapter 17) to give two levels of news topic granularity. See Hatzivassiloglou et al. (2000) for details, and Chen and Lin (2000) and Radev et al. (2001) for related systems. Other applications of clustering in information retrieval are duplicate detection (Yang and Callan (2006), Section 19.6, page 400), novelty detection (see references in Section 17.9, page 367) and metadata discovery on the semantic web (Alonso et al. 2006).

The discussion of external evaluation measures is partially based on Strehl (2002). Dom (2002) proposes a measure Q_0 that is better motivated theoretically than NMI. Q_0 is the number of bits needed to transmit class memberships assuming cluster memberships are known. The Rand index is due to Rand (1971). Hubert and Arabie (1985) propose an *adjusted Rand index* that ranges between -1 and 1 and is 0 if there is only chance agreement between clusters and classes (similar to κ in Chapter 8, page 152). Basu et al. (2004) argue that the three evaluation measures NMI, Rand index, and F measure give very similar results. Stein et al. (2003) propose *expected edge density* as an

ADJUSTED RAND INDEX

internal measure and give evidence that it is a good predictor of the quality of a clustering. Kleinberg (2002) and Meilă (2005) present axiomatic frameworks for comparing clusterings.

Authors that are often credited with the invention of the K-means algorithm include Lloyd (1982) (first distributed in 1957), Ball (1965), MacQueen (1967), and Hartigan and Wong (1979). Arthur and Vassilvitskii (2006) investigate the worst-case complexity of K-means. Bradley and Fayyad (1998), Pelleg and Moore (1999), and Davidson and Satyanarayana (2003) investigate the convergence properties of K-means empirically and how it depends on initial seed selection. Dhillon and Modha (2001) compare K-means clusters with SVD-based clusters (Chapter 18). The K-medoid algorithm was presented by Kaufman and Rousseeuw (1990). The EM algorithm was originally introduced by Dempster et al. (1977). An in-depth treatment of EM is (McLachlan and Krishnan 1996). See Section 18.5 (page 383) for publications on latent analysis, which can also be viewed as soft clustering.

AIC is due to Akaike (1974) (see also Burnham and Anderson (2002)). An alternative to AIC is BIC, which can be motivated as a Bayesian model selection procedure (Schwarz 1978). Fraley and Raftery (1998) show how to choose an optimal number of clusters based on BIC. An application of BIC to K-means is (Pelleg and Moore 2000). Hamerly and Elkan (2003) propose an alternative to BIC that performs better in their experiments. Another influential Bayesian approach for determining the number of clusters (simultaneously with cluster assignment) is described by Cheeseman and Stutz (1996). Two methods for determining cardinality without external criteria are presented by Tibshirani et al. (2001).

We only have space here for classical completely unsupervised clustering. An important current topic of research is how to use prior knowledge to guide clustering (e.g., Ji and Xu (2006)) and how to incorporate interactive feedback during clustering (e.g., Huang and Mitchell (2006)). Fayyad et al. (1998) propose an initialization for EM clustering. For algorithms that can cluster very large data sets in one scan through the data see Bradley et al. (1998).

The applications in Table 16.1 all cluster documents. Other information retrieval applications cluster words (e.g., Crouch 1988), contexts of words (e.g., Schütze and Pedersen 1995) or words and documents simultaneously (e.g., Tishby and Slonim 2000, Dhillon 2001, Zha et al. 2001). Simultaneous clustering of words and documents is an example of *co-clustering* or *biclustering*.

CO-CLUSTERING