16 Flat clustering

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

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

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 – a partition that is given in classification and computed in clustering. But as we will see the two problems are fundamentally different. In supervised classification (Chapter 13, page 196), 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 teacher that would guide us.

The key input to a clustering algorithm is the similarity measure. In Figure 16.1, it is the different degrees of closeness of points to each other that define three different clusters. In document clustering, the similarity measure is usually vector space similarity or distance (Chapter 7). Different similarity measures give rise to different clusterings. Thus, the similarity measure is the main means by which we can influence the outcome of clustering.

Flat clustering creates a flat set of clusters without any explicit structure that would relate clusters to each other. Hierarchical algorithms create a hierarchy of clusters and will be covered in Chapter 17. Chapter 17 also addresses the difficult problem of labeling clusters automatically.

A second important distinction is between partitional and non-partitional clustering algorithms. Partitional algorithms compute a hard assignment – each document is a member of exactly one cluster. The assignment of non-partitional algorithms is soft – a document’s assignment is a distribution over all clusters. In a soft assignment, a document has fractional membership (or in some clustering formalisms full membership) in several clusters.

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 partitional algorithm, and the Expectation-Maximization (or EM) algorithm (Section 16.5), a non-partitional algorithm. K-means is perhaps the most widely used flat clustering algorithm due to 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

The cluster hypothesis states the fundamental assumption we make when using clustering in information retrieval:

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

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 words. The cluster hypothesis is essentially identical to the
16.1 Clustering in information retrieval

<table>
<thead>
<tr>
<th>Application</th>
<th>What is clustered?</th>
<th>Benefit</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>Result set clustering</td>
<td>result set</td>
<td>more effective information presentation to user</td>
<td>Figure 16.2</td>
</tr>
<tr>
<td>Scatter-Gather</td>
<td>(subsets of)</td>
<td>alternative user interface: “search without typing”</td>
<td>Figure 16.3</td>
</tr>
<tr>
<td>Collection clustering</td>
<td>collection</td>
<td>effective information presentation for exploratory browsing</td>
<td>McKeown et al. (2002), <a href="http://news.google.com">http://news.google.com</a></td>
</tr>
<tr>
<td>Language modeling</td>
<td>collection</td>
<td>increased precision and/or recall</td>
<td>Liu and Croft (2004)</td>
</tr>
<tr>
<td>Cluster-based retrieval</td>
<td>collection</td>
<td>higher efficiency: faster search</td>
<td>Salton (1971a)</td>
</tr>
</tbody>
</table>

Table 16.1 Some applications of clustering in information retrieval.

The contiguity hypothesis in Chapter 14 (page 217). 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 information retrieval. They differ in the set of documents that they cluster – result set, collection or subsets of the collection – and the aspect of an information retrieval 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.

The first application mentioned in Table 16.1 is result set clustering. The default presentation of search results in information retrieval is a simple list. Users scan the list from top to bottom until they have found the information they are looking for. Instead, result set clustering clusters the result set, 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 Vivisimo search engine is a more effective user interface for understanding what is in the search result than a simple list of documents.

A better user interface is also the goal of Scatter-Gather, the second application in Figure 16.2. Scatter-Gather clusters the whole collection to get groups of documents that the user can select (“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.

Automatically generated clusters like those in Figure 16.3 will never be as neatly organized as a manually constructed hierarchical tree like the online...
Clustering of search results to improve user 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).

Yahoo directory. Also, finding descriptive labels for clusters automatically is a difficult problem (Section 17.5, page 289). But cluster-based navigation is an interesting alternative to keyword searching, the standard information retrieval 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 since 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.
16.1 Clustering in information retrieval

Figure 16.3 The Scatter-Gather user interface. 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).

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 since 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 (12.5), page 186, 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 words 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 words in \( d \).

Clustering can also speed up search. As we saw in Section 7.1.2, page 100, 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, e.g., 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. Since there are many fewer clusters than documents, finding the closest cluster is fast; and since the documents matching a query are all similar to each other, they tend to be in the same clusters. While this algorithm is inexact, the expected decrease in search quality is small. This is essentially the application of clustering that was covered in Section 7.2.1 (page 104).

### 16.2 Problem statement

We can define the goal in partitional 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.

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 average similarity between documents and their centroids. The discussion of similarity measures and distance metrics in Chapter 14 (page 219) 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, which we approximate using the now familiar vector space representation. If the intended similarity is something else, for example, 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, say, English and French documents.

A difficult issue in clustering is determining the cardinality of a clustering, the number \( K \) of clusters. 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 of computer monitors in the 1990s.

Since 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, for $N$ documents and $K$ clusters, there are $K^N / K!$ different partitions, so this approach is not feasible. 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 intra-cluster similarity (documents within a cluster are similar) and low inter-cluster 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 performance in an application. An alternative to internal criteria is direct evaluation in the application of interest. For result set 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 121, and Section 13.6, page 210). The gold standard is ideally produced by human judges with a good level of inter-judge agreement (see Chapter 8, page 122). 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 jaguar result set in Figure 16.2 consists of three classes corresponding to the three senses car, animal, and operating system.

This section introduces four external criteria of clustering quality. Purity is a simple and transparent evaluation measure. Normalized mutual information is information-theoretically motivated and can therefore be more easily interpreted than other measures. 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.

The first measure is purity. Suppose we have a set of clusters $\Omega = \{\omega_1, \omega_2, \ldots, \omega_K\}$ and a set of classes $\Gamma = \{c_1, c_2, \ldots, c_J\}$. Then the purity of a clustering is defined as the proportion of documents that are in the majority class of their
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); circle, 4 (cluster 2); and diamond, 3 (cluster 3). Purity is \((1/17) \times (5 + 4 + 3) \approx 0.71\).

We present an example of how to compute purity in Figure 16.4. High purity is easy to achieve for a large number of clusters – 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.

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

\[
\text{NMI}(\Omega, \Gamma) = \frac{I(\Omega; \Gamma)}{\left(H(\Omega) + H(\Gamma)\right)/2}
\]

where \(I(\Omega; \Gamma)\) is mutual information or MI (cf. Chapter 13, page 204):

\[
I(\Omega; \Gamma) = \sum_k \sum_j \hat{P}(\omega_k \cap c_j) \log \frac{\hat{P}(\omega_k \cap c_j)}{\hat{P}(\omega_k) \hat{P}(c_j)}
\]

where \(\hat{P}(\omega_k), \hat{P}(c_j), \) and \(\hat{P}(\omega_k \cap c_j)\) are the estimated probabilities (or relative frequencies) of a document being in \(\omega_k, c_j\), and in the intersection of \(\omega_k\) and \(c_j\), respectively.

\(I(\Omega; \Gamma)\) in Equation (16.1) measures the amount of information by which our knowledge about the classes increases when we are told what the clusters are. This term reaches a minimum of 0 if the clustering is random with respect to the classes. In that case, knowing that a document is in a particular cluster does not give us any new information about what class it might be in. Maximum mutual information is reached for a clustering \(\Omega_{\text{exact}}\) that perfectly recreates the classes – but also if clusters in \(\Omega_{\text{exact}}\) are further subdivided into smaller clusters (Exercise 16.4). So MI has the same problem as...
16.3 Evaluation of clustering

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 \([H(Ω) + H(Γ)]/2\) in Equation (16.1) fixes this problem since entropy tends to increase with the number of clusters. For example, \(H(Ω)\) reaches its maximum \(\log_2 N\) 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.

An alternative to the information-theoretic interpretation is to view clustering 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 percentage of decisions that are correct.

\[
RI = \frac{TP + TN}{TP + FP + FN + TN}
\]

As an example, we compute RI for Figure 16.4. We first compute TP + FP. The three clusters contain 6, 6, and 5 points, respectively, so the total number of “positives” or pairs of documents that are in the same cluster is:

\[
TP + FP = \binom{6}{2} + \binom{6}{2} + \binom{5}{2} = 40
\]

Of those, the X pairs in cluster 1, the circle pairs in cluster 2, the diamond pairs in cluster 3, and the X pair in cluster 3 are true positives:

\[
TP = \binom{5}{2} + \binom{4}{2} + \binom{3}{2} + \binom{2}{2} = 20
\]

Thus, FP = 40 - 20 = 20.

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

<table>
<thead>
<tr>
<th></th>
<th>same cluster</th>
<th>different clusters</th>
</tr>
</thead>
<tbody>
<tr>
<td>same class</td>
<td>TP = 20</td>
<td>FN = 24</td>
</tr>
<tr>
<td>different classes</td>
<td>FP = 20</td>
<td>TN = 72</td>
</tr>
</tbody>
</table>

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

The Rand index gives equal weight to false positives and false negatives. In some situations, separating similar documents is worse than putting pairs of dissimilar documents in the same cluster. We can use the F measure, the harmonic mean of precision and recall (Section 8.3, page 114), to penalize
false negatives more strongly than false positives by selecting a value $\beta > 1$, thus giving more weight to recall.

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

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 information retrieval, evaluating clustering with $F$ has the added advantage that the measure is already familiar to the research community.

### 16.4 K-means

K-means is the most important flat clustering algorithm. Its objective is to minimize the average squared distance of documents from their cluster centers where a cluster center is defined as the mean or centroid $\bar{\mu}$ of the documents in a cluster $\omega$:

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

This definition assumes that documents are represented as vectors in a real-valued space in the familiar way. We used centroids for Rocchio classification in Chapter 14 (page 219). They play a very 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 and initially do not know which documents will end up in the same cluster.

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:

$$RSS_k = \sum_{x \in \omega_k} \|x - \bar{\mu}(\omega_k)\|^2 = \sum_{x \in \omega_k} \sum_{m=1}^{M} (x_m - \mu_m(\omega_k))^2$$

(16.2)

$$RSS = \sum_{k=1}^{K} RSS_k$$

where $x_m$ and $\mu_m(\omega_k)$ are the $m^{th}$ components of $x$ and $\bar{\mu}(\omega_k)$, respectively. This is the objective function in k-means and our goal is to minimize it. Since $N$ is fixed, minimizing RSS is equivalent to minimizing the average squared distance, a measure of how well centroids represent their documents.
16.4 K-means

Given:

- $D$: a set of $N$ vectors
- $K$: desired number of clusters

Select $K$ random seeds $\{s_1, s_2, \ldots, s_K\}$ from $D$

Let $\mu(\omega_k) := s_k$, $1 \leq k \leq K$

Repeat until stopping criterion is met:

- **Reassignment step**
  - Assign each $x_n$ to cluster $\omega_k$ s.t. $\|x_n - \mu(\omega_k)\|$ is minimal
- **Recomputation step**: For each $\omega_k$:
  - $\mu(\omega_k) = \frac{1}{|\omega_k|} \sum_{x \in \omega_k} x$

$\Rightarrow$ **Figure 16.5** The k-means algorithm.

$\Rightarrow$ **Figure 16.6** One iteration of the k-means algorithm in $\mathbb{R}^2$. The position of the two cluster centers (shown as circles) converges after one iteration.

K-means usually starts with selecting as initial cluster centers $K$ randomly selected documents, the seeds. It then moves the cluster centers around in space in order 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 one iteration of the k-means algorithm for a set of four points. The third column of Table 17.2 (page 290) shows examples of centroids.

Several termination conditions can be used as stopping criteria:

- A fixed number of iterations $I$ has been completed.
• Assignment of documents to clusters (the partitioning function \( \gamma \)) does not change between iterations.

• Centroids \( \vec{\mu}_k \) do not change between iterations.

We now show that k-means converges by proving that RSS monotonically decreases in each iteration. First, RSS decreases in the reassignment step since each vector is assigned to the closest centroid, so the distance it contributes to RSS decreases (or does not change). Secondly, it decreases in the recomputation step because the new centroid is the vector \( \vec{v} \) for which RSS reaches its minimum.

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

(16.3)

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

(16.4)

Setting the partial derivative to zero, we get:

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

(16.5)

which is the componentwise definition of the centroid. Thus, we minimize \( \text{RSS}_k \) when the old centroid is replaced with the new centroid. RSS, the sum of the \( \text{RSS}_k \), must then also decrease during recomputation.

Since there is only a finite set of possible clusterings, a monotonically decreasing algorithm will eventually arrive at a minimum. Take care, however, to break ties consistently, e.g., 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.

This proves the convergence of k-means, but there is unfortunately no guarantee that a global minimum will be reached. This is a particular problem if a document set contains many outliers. Frequently, if an outlier is chosen as an initial seed, then no other vector is assigned to it during subsequent iterations. We end up 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.

Effective heuristics for seed selection include excluding outliers from the seed set; trying out multiple starting points and choosing the clustering with lowest cost; and obtaining seeds from another method such as hierarchical clustering (see Chapter 17). Since hierarchical clustering methods are deterministic and therefore more predictable than k-means, a hierarchical clustering of a small random sample of size \( iK \) (e.g., for \( i = 5 \) or \( i = 10 \)) will often
provide good seeds (see description of the Buckshot algorithm, Chapter 17, page 292). If seeds are provided by a method other than random selection, they can be arbitrary vectors and need not be members of the set of documents to be clustered.

What is the time complexity of k-means? Most of the time is spent on computing vector distances. One such operation costs \(O(M)\). The reassignment step computes \(O(KN)\) distances, so its overall complexity is \(O(KNM)\). In the recomputation step, each vector gets added to a centroid once, so the complexity of this step is \(O(NM)\). For a fixed number of iterations \(I\), the overall complexity is therefore \(O(INM)\). 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 \(I\), but this rarely does harm in practice. In most cases, k-means converges quickly.

There is one subtlety in the preceding argument. Even a linear algorithm can be quite slow if one of the arguments of \(O(\ldots)\) 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, on average, a small fraction of the theoretically possible \(M\) componentwise differences need to be computed. Centroids, however, are dense since 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., \(k = 1000\)) hardly decreases cluster quality while achieving a significant speedup of the reassignment step (see references in Section 16.6).

The same efficiency problem is addressed by \textit{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 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 number of $K$ according to the objective function, i.e., the number $K$ that minimizes RSS. But $\text{RSS}_{\text{min}}(K)$ is a monotonically decreasing function in $K$ (see Exercise 16.11), which reaches its minimum 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 inspect the $\text{RSS}_{\text{min}}$ values as the number of clusters increases and find the “knee” in the curve – the point where successive decreases in $\text{RSS}_{\text{min}}$ 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 $K$, 4 and 9 in this case.

A second type of criterion for cluster cardinality imposes a penalty for each new cluster. To do this, we create a generalized objective function that combines two elements: distortion, a measure of how much documents deviate from the prototype of their clusters (e.g., RSS for k-means); and a measure of the complexity of the clustering, which is usually a function of the number of clusters. For k-means, we get this selection criterion for $K$:

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

(16.6)

where $\lambda$ is a weighting factor. A large value of $\lambda$ 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 (16.6) is that we need to determine $\lambda$. If this is not easier than determining $K$ directly, then we are back to square one. In some cases, we can choose values of $\lambda$ 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 $\lambda$ 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 since $K$ changes.

A theoretical justification for Equation (16.6) is the Akaike Information Criterion or AIC. For k-means it can be stated as follows:

$$\text{AIC}: K = \arg \min_K [\text{RSS}_{\min}(K) + 2 \cdot M \cdot K]$$

(16.7)

Equation (16.7) is a special case of Equation (16.6) for $\lambda = 2M$.

The general form of AIC is:

$$\text{AIC}: K = \arg \min_K [-2L(K) + 2q(K)]$$

(16.8)

where $-L(K)$, the negative maximum log likelihood of the data for $K$ clusters, is a measure of distortion and $q(K)$ is the number of parameters of a model with $K$ clusters. To derive Equation (16.7) from Equation (16.8) observe that $q(K) = KM$ in k-means since each element of the $K$ centroids is a parameter that can be varied independently; and that $L(K) = -(1/2)\text{RSS}_{\min}(K)$ (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 16.17).

The derivation of AIC is based on a number of assumptions, e.g., that the data are independently and identically distributed. These assumptions are
only approximately true for data sets in information retrieval. As a consequence, the AIC can rarely be applied as is in text clustering. In Figure 16.8, the dimensionality of the vector space is $M \approx 50,000$. Thus, $q(K) > 50,000$ dominates the smaller RSS-based term ($\text{RSS}_{\min}(1) < 5000$, 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 (16.6) is often more useful than Equation (16.7) – with the caveat that we need to come up with an estimate for $\lambda$.

16.5 Model-based clustering

In k-means, we attempt to find centroids that are good representatives. We can view the 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 the data were generated by a model and then tries to recover the original model from the data.

A commonly used criterion for selecting the model is maximum likelihood. In k-means, the quantity $\exp(-\text{RSS})$ 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 identical criteria.

More generally, we write $\Theta$ for the parameters that describe the model. In k-means, $\Theta = \{\vec{\mu}_1, \ldots, \vec{\mu}_K\}$. The maximum likelihood criterion is then to select the model $\Theta$ that maximizes the log likelihood of generating the data $D$:

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

$L(D|\Theta)$ 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 we took in Chapter 12 (page 181) for language modeling and in Chapter 13 (page 197) for text classification. In text classification, we chose the class that maximizes the likelihood of generating a particular document. Here, we choose the clustering $\Theta$ that maximizes the likelihood of generating the a set of documents that is given. Once we have $\Theta$, we can compute an assignment probability $P(d|\omega_k; \Theta)$ for each document-cluster pair.

Because cluster membership is a probability distribution in model-based clustering, assignment to clusters is soft (as defined earlier in this chapter
A document about Chinese cars may get soft assignments of 0.5 to 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 binomial (as in the example below), Gaussian (another case that often occurs in document clustering) or a member of a different family.

A commonly used algorithm for model-based clustering is the Expectation-Maximization algorithm or EM algorithm. EM clustering is an iterative algorithm that maximizes $L(D|\Theta)$. In principle, EM can be applied to any probabilistic modeling of the data. We will work with a mixture of multivariate binomials, the distribution we know from Section 11.3 (page 170) and Section 13.3 (page 202):

$$P(d|\omega_k, \Theta) = \left(\prod_{w_m \in d} q_{mk}\right) \left(\prod_{w_m \notin d} (1 - q_{mk})\right)$$

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. $P(U_m = 1|\omega_k)$ is the probability that a document from cluster $k$ contains word $w_m$. $\alpha_k$ is the prior probability of cluster $\omega_k$: the probability that a document $d$ is in $\omega_k$ if we have no information about $d$.

The mixture model then is:

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

In this model, we generate a document by first picking a cluster $k$ with probability $\alpha_k$ and then generating the words of the document according to the parameters $q_{mk}$. Recall that the document representation of the multivariate binomial 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 $\Theta$ that maximize $L(D|\Theta)$? EM is quite 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 models. The parameters in k-means are the centroids, the parameters in EM are the $\alpha_k$ and $q_{mk}$.

The maximization step recomputes the conditional parameters $q_{mk}$ and the priors $\alpha_k$ as follows:
Maximization Step:  
\[ q_{mk} = \frac{\sum_{n=1}^{N} r_{nk} I(w_m \in d_n)}{\sum_{m=1}^{N} r_{nk}} \]

\[ \alpha_k = \frac{\sum_{n=1}^{N} r_{nk}}{N} \]  

(16.10)  

\( r_{nk} \) 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 binomial from Table 13.2 (page 202) 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 \( \alpha_k \):

Expectation Step:  
\[ r_{nk} = \frac{\alpha_k (\prod_{w_m \in d_n} q_{mk}) (\prod_{w_m \not\in d_n} (1 - q_{mk}))}{\sum_{k=1}^{K} \alpha_k (\prod_{w_m \in d_n} q_{mk}) (\prod_{w_m \not\in d_n} (1 - q_{mk}))} \]  

(16.11)  

This expectation step applies Equation (16.9) to computing the normalized likelihood that \( \omega_k \) generated document \( d_n \). It is the classification procedure for the multivariate Bernoulli in Table 13.2. Thus, the expectation step is nothing else but Naive Bayes classification.

We clustered a set of 11 documents using EM in Table 16.2. After convergence in iteration 25, the first 5 documents are assigned to cluster 1 (\( r_{i,1} = 1.00 \)) and the last 6 to cluster 2 (\( r_{i,1} = 0.00 \)). Somewhat untypically, the final assignment is a hard assignment here. EM usually converges to a soft assignment. In iteration 25, the prior for cluster 1 is \( 5/11 \approx 0.45 \) because 5 of the 11 documents are in cluster 1. Some words 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 because they share sugar. For parameters of words occurring in ambiguous contexts, convergence takes longer. Seed documents 6 and 7 both contain sweet. As a result, it takes 25 iterations for the word to be unambiguously associated with one cluster.

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 with EM which also occurs in applications of EM the reader may have encountered in other contexts: hidden markov models, probabilistic grammars, and machine translation. 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.”
16.5 Model-based clustering

<table>
<thead>
<tr>
<th>docid</th>
<th>document text</th>
<th>docid</th>
<th>document text</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>hot chocolate cocoa beans</td>
<td>7</td>
<td>sweet sugar</td>
</tr>
<tr>
<td>2</td>
<td>cocoa ghana africa</td>
<td>8</td>
<td>sugar cane brazil</td>
</tr>
<tr>
<td>3</td>
<td>beans harvest ghana</td>
<td>9</td>
<td>sweet sugar beet</td>
</tr>
<tr>
<td>4</td>
<td>cocoa butter</td>
<td>10</td>
<td>sweet cake icing</td>
</tr>
<tr>
<td>5</td>
<td>butter truffles</td>
<td>11</td>
<td>cake black forest</td>
</tr>
<tr>
<td>6</td>
<td>sweet chocolate</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>parameter</th>
<th>iteration of clustering</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\alpha_1$</td>
<td>0.50 0.45 0.52 0.56 0.55 0.53 0.45</td>
</tr>
<tr>
<td>$r_{1,1}$</td>
<td>1.00 1.00 1.00 1.00 1.00 1.00 1.00</td>
</tr>
<tr>
<td>$r_{2,1}$</td>
<td>0.50 0.73 0.97 1.00 1.00 1.00 1.00</td>
</tr>
<tr>
<td>$r_{3,1}$</td>
<td>0.50 0.80 0.99 1.00 1.00 1.00 1.00</td>
</tr>
<tr>
<td>$r_{4,1}$</td>
<td>0.50 0.71 0.87 0.98 1.00 1.00 1.00</td>
</tr>
<tr>
<td>$r_{5,1}$</td>
<td>0.50 0.53 0.61 0.78 0.96 1.00 1.00</td>
</tr>
<tr>
<td>$r_{6,1}$</td>
<td>1.00 1.00 1.00 1.00 1.00 0.74 0.00</td>
</tr>
<tr>
<td>$r_{7,1}$</td>
<td>1.00 0.00 0.00 0.00 0.00 0.00 0.00</td>
</tr>
<tr>
<td>$r_{8,1}$</td>
<td>0.00 0.00 0.00 0.00 0.00 0.00 0.00</td>
</tr>
<tr>
<td>$r_{9,1}$</td>
<td>0.00 0.00 0.00 0.00 0.00 0.00 0.00</td>
</tr>
<tr>
<td>$r_{10,1}$</td>
<td>0.50 0.42 0.19 0.02 0.00 0.00 0.00</td>
</tr>
<tr>
<td>$r_{11,1}$</td>
<td>0.50 0.56 0.53 0.28 0.02 0.00 0.00</td>
</tr>
<tr>
<td>$q_{africa,1}$</td>
<td>0.000 0.036 0.046 0.057 0.061 0.064 0.071</td>
</tr>
<tr>
<td>$q_{africa,2}$</td>
<td>0.000 0.031 0.019 0.002 0.000 0.000 0.000</td>
</tr>
<tr>
<td>$q_{brazil,1}$</td>
<td>0.000 0.000 0.000 0.000 0.000 0.000 0.000</td>
</tr>
<tr>
<td>$q_{brazil,2}$</td>
<td>0.000 0.062 0.071 0.077 0.074 0.070 0.062</td>
</tr>
<tr>
<td>$q_{cocoa,1}$</td>
<td>0.000 0.143 0.152 0.167 0.181 0.191 0.214</td>
</tr>
<tr>
<td>$q_{cocoa,2}$</td>
<td>0.000 0.063 0.040 0.012 0.001 0.000 0.000</td>
</tr>
<tr>
<td>$q_{sugar,1}$</td>
<td>0.000 0.000 0.000 0.000 0.000 0.000 0.000</td>
</tr>
<tr>
<td>$q_{sugar,2}$</td>
<td>0.500 0.187 0.214 0.231 0.221 0.210 0.187</td>
</tr>
<tr>
<td>$q_{sweet,1}$</td>
<td>0.500 0.107 0.089 0.070 0.062 0.054 0.000</td>
</tr>
<tr>
<td>$q_{sweet,2}$</td>
<td>0.500 0.156 0.185 0.216 0.219 0.220 0.250</td>
</tr>
</tbody>
</table>

Table 16.2 The EM clustering algorithm. The table shows a set of documents (above) and parameter values for selected iterations during EM clustering (below). Parameters shown are prior $\alpha_1$, soft assignment scores $r_{nk}$ (both omitted for cluster 2), and lexical parameters $q_{mk}$ for a few words. 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 16.10 were replaced with $r_{nk} + \epsilon$ where $\epsilon = 0.0001$. 

Preliminary draft (c) 2007 Cambridge UP
16.6 References and further reading

A more general introduction to clustering, covering both k-means and EM, but without reference to text-specific issues, can be found in Duda et al. (2000). Rasmussen (1992) gives an introduction to clustering from an information retrieval perspective. The cluster hypothesis is due to Jardine and van Rijsbergen (1971), where it was stated as follows: Associations between documents convey information about the relevance of documents to requests. Croft (1978) shows that cluster-based retrieval can be more accurate as well as more efficient than regular search. However, Voorhees (1985a) presents evidence that accuracy does not improve consistently across collections.

There is good evidence that clustering of result sets improves user experience and search result quality (Hearst and Pedersen 1996, Zamir and Etzioni 1999, Tombros et al. 2002, Käki 2005), although not as much as result set structuring based on carefully edited category hierarchies (Hearst 2006). The Scatter-Gather interface for browsing collections was presented by Cutting et al. (1992). The cluster-based language modeling approach was pioneered by Liu and Croft (2004). Schütze and Silverstein (1997) evaluate two methods for avoiding inefficient dense centroids in 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. See also (Chen and Lin 2000, Radev et al. 2001). Other applications of clustering in information retrieval are duplicate detection (Yang and Callan 2006) and metadata discovery on the semantic web (Alonso et al. 2006).

The discussion of external evaluation measures is partially based on Strehl (2002). Dom (2001) 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 the kappa measure in Chapter 8, page 121). Basu et al. (2004) argue that the three evaluation measures NMI, Rand index and F measure give very similar results.

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). The k-medoid algorithm was presented by Kaufman and Rousseeuw (1990). The EM algorithm was originally introduced by Dempster et al. (1977).

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). BIC is applied to k-means by 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 (Ji and Xu 2006) and how to incorporate interactive feedback during clustering (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)) or contexts of words (e.g., Schütze and Pedersen (1995)).

16.7 Exercises

Exercise 16.1
Define two documents as similar if they have $n$ content words in common (for, say, $n = 5$). What are some examples of documents for which the cluster hypothesis does not hold?

Exercise 16.2
Why are documents that do not use the same word for the concept car likely to end up in the same cluster?

Exercise 16.3
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.

Exercise 16.4
Let $\Omega$ be a clustering that exactly reproduces a class structure $\Gamma$ and $\Omega'$ a clustering that further subdivides some clusters. Show that $I(\Omega, \Gamma) = I(\Omega', \Gamma)$.

Exercise 16.5
Compute the mutual information between clusters and classes in Figure 16.4.

Exercise 16.6
Replace every point $d$ in Figure 16.4 with two points in the same class as $d$. Compute purity, NMI, $RI$, and $F_1$ for this clustering. Do the measures increase or decrease after doubling the number of points? Does this correspond to your intuition as to the relative difficulty of clustering a set twice as large?

Exercise 16.7
Which of the four evaluation measures introduced above are symmetric in the sense that the measure’s value does not change if the roles of clusters and classes are switched?
Exercise 16.8
Two of the possible termination conditions for k-means were (1) assignment does not change, (2) centroids do not change (page 261). Do these two conditions imply each other?

Exercise 16.9
Compute RSS for the two clusterings in Figure 16.7.

Exercise 16.10
Download Reuters-21578. (i) Compute a k-means clustering of the entire collection (training and test sets) into 10 clusters. There are a number of software packages that implement k-means, e.g., WEKA (Witten and Frank 2005) and R (R Development Core Team 2005). (ii) Compute purity, normalized mutual information, $F_1$ and RI for the classes acquisitions, corn, crude, earn, grain, interest, money-fx, ship, trade, and wheat. (iii) Compile a confusion matrix (Table 14.4, page 233) for these 10 classes and 10 clusters. Identify classes that give rise to false positives and false negatives.

Exercise 16.11
Show that $\text{RSS}_{\text{min}}(K)$ as defined above is monotonically decreasing.

Exercise 16.12
There is a soft version of k-means that computes the assignment strength of a document to a cluster as the distance from its centroid. Modify reassignment and recomputation steps of hard k-means for this soft version.

Exercise 16.13
In the last iteration in Table 16.2, document 6 is in cluster 2 even though it was the initial seed for cluster 1. Why does the document change membership?

Exercise 16.14
The values of the parameters $q_{mk}$ in iteration 25 in Table 16.2 are rounded. What are the exact values that EM will converge to?

Exercise 16.15
Perform a k-means clustering for the documents in Table 16.2. After how many iterations does k-means converge? Compare the result to the EM clustering in Table 16.2 and discuss the differences.

Exercise 16.16
Modify the expectation and maximization steps of EM for a Gaussian mixture. As with Naive Bayes, the maximization step computes the maximum likelihood parameter estimates $\alpha_k$, $\mu_k$, and $\Sigma_k$ 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 corresponding equations.

Exercise 16.17
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.

Exercise 16.18
We saw above that the time complexity of k-means is $O(INKM)$. What is the time complexity of EM?
Exercise 16.19
The within-point scatter of a cluster $\omega$ is defined as $\frac{1}{2} \sum_{x_i \in \omega} \sum_{x_j \in \omega} \| x_i - x_j \|^2$. Show that minimizing RSS and minimizing within-point scatter are equivalent.

Exercise 16.20
Derive an AIC criterion for the multivariate binomial mixture model from Equation (16.8).