**Complexity.** In step 1 of the $k$-means algorithm, we find the nearest neighbor to each of $N$ $n$-vectors, over the list of $k$ centroids. This requires approximately $3Nkn$ flops. In step 2 we average the $n$-vectors over each of the cluster groups. For a cluster with $p$ vectors, this requires $n(p-1)$ flops, which we approximate as $np$ flops; averaging all clusters requires a total of $Nn$ flops. This is less than the cost of partitioning in step 1. So $k$-means requires around $(3k+1)Nn$ flops per iteration. Its order is $Nkn$ flops.

Each run of $k$-means typically takes fewer than a few tens of iterations, and usually $k$-means is run some modest number of times, like 10. So a very rough guess of the number of flops required to run $k$-means 10 times (in order to choose the best partition found) is $1000Nkn$ flops.

As an example, suppose we use $k$-means to partition $N = 100000$ vectors with size $n = 100$ into $k = 10$ groups. On a 1 Gflop/s computer we guess that this will take around 100 seconds. Given the approximations made here (for example, the number of iterations that each run of $k$-means will take), this is obviously a crude estimate.

## 4.4 Examples

### 4.4.1 Image clustering

The MNIST (Mixed National Institute of Standards) database of handwritten digits is a data set containing $N = 60000$ grayscale images of size $28 \times 28$, which we represent as $n$-vectors with $n = 28 \times 28 = 784$. Figure 4.6 shows a few examples from the data set. (The data set is available from Yann LeCun at `yann.lecun.com/exdb/mnist`.)

We use the $k$-means algorithm to partition these images into $k = 20$ clusters, starting with a random assignment of the vectors to groups, and repeating the experiment 20 times. Figure 4.7 shows the clustering objective versus iteration number for three of the 20 initial assignments, including the two that gave the lowest and the highest final values of the objective.

Figure 4.8 shows the representatives with the lowest final value of the clustering objective. Figure 4.9 shows the set with the highest value. We can see that most of the representatives are recognizable digits, with some reasonable confusion, for example between '4' and '9' or '3' and '8'. This is impressive when you consider that the $k$-means algorithm knows nothing about digits, handwriting, or even that the 784-vectors represent $28 \times 28$ images; it uses only the distances between 784-vectors. One interpretation is that the $k$-means algorithm has 'discovered' the digits in the data set.

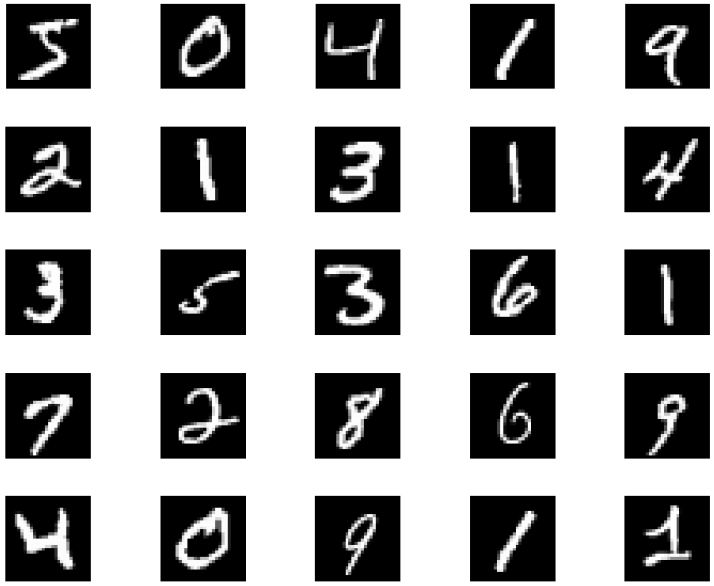

**Figure 4.6** 25 images of handwritten digits from the MNIST data set. Each image has size $28 \times 28$, and can be represented by a 784-vector.

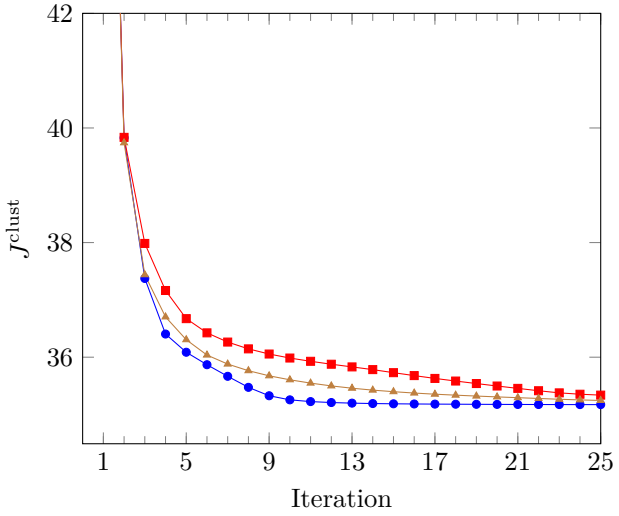


**Figure 4.7** Clustering objective $J^{\text{clust}}$ after each iteration of the $k$-means algorithm, for three initial partitions, on digits of the MNIST set.

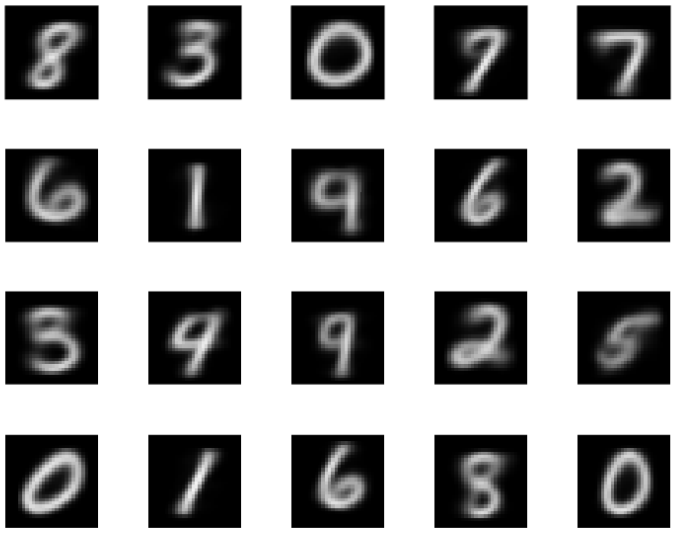

**Figure 4.8** Group representatives found by the $k$-means algorithm applied to the MNIST set.

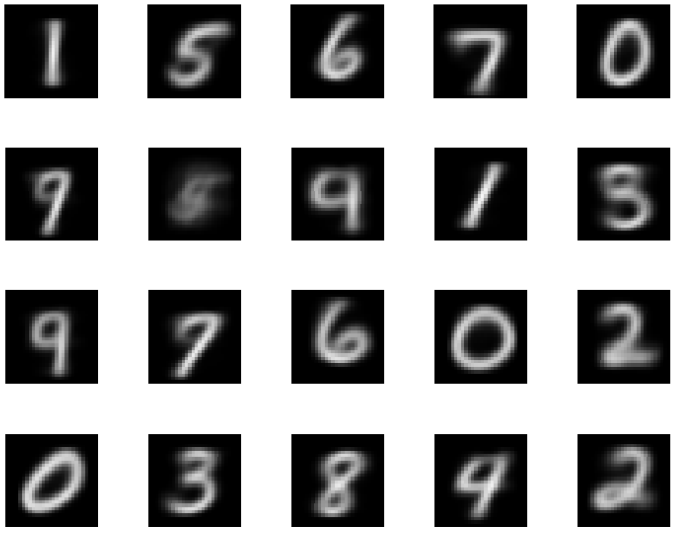

**Figure 4.9** Group representatives found by the $k$-means algorithm applied to the MNIST set, with a different starting point than in figure 4.8.

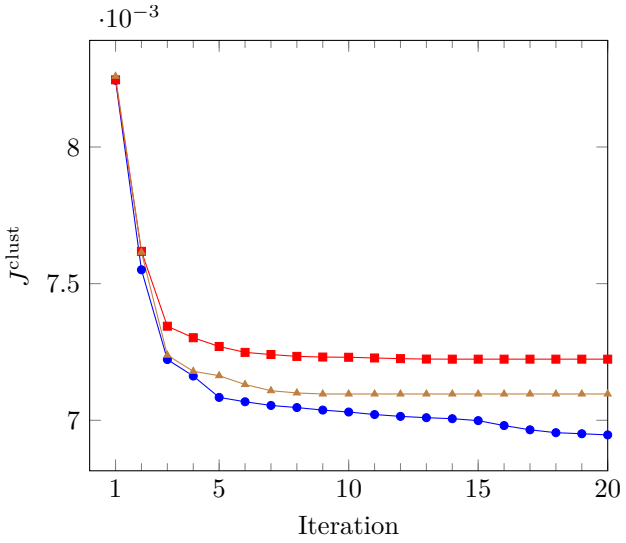


**Figure 4.10** Clustering objective $J^{\text{clust}}$ after each iteration of the $k$-means algorithm, for three initial partitions, on Wikipedia word count histograms.

### 4.4.2 Document topic discovery

We start with a corpus of $N = 500$ Wikipedia articles, compiled from weekly lists of the most popular articles between September 6, 2015, and June 11, 2016. We remove the section titles and reference sections (bibliography, notes, references, further reading), and convert each document to a list of words. The conversion removes numbers and stop words, and applies a stemming algorithm to nouns and verbs. We then form a dictionary of all the words that appear in at least 20 documents. This results in a dictionary of 4423 words. Each document in the corpus is represented by a word histogram vector of length 4423.

We apply the $k$-means algorithm with $k = 9$, and 20 randomly chosen initial partitions. The $k$-means algorithm converges to similar but slightly different clusterings of the documents in each case. Figure 4.10 shows the clustering objective versus iteration of the $k$-means algorithm for three of these, including the one that gave the lowest final value of $J^{\text{clust}}$, which we use below.

Table 4.1 summarizes the clustering with the lowest value of $J^{\text{clust}}$. For each of the nine clusters we show the largest ten coefficients of the word histogram of the cluster representative. Table 4.2 gives the size of each cluster and the titles of the ten articles closest to the cluster representative.

Each of the clusters makes sense, and mostly contains documents on similar topics, or similar themes. The words with largest coefficients in the group representatives also make sense. It is interesting to see that $k$-means clustering has assigned movies and TV series (mostly) to different clusters (9 and 6). One can also note that clusters 8 and 9 share several top key words but are on separate topics (actors and movies, respectively).

| Cluster 1 | | Cluster 2 | | Cluster 3 | |
|---|---|---|---|---|---|
| Word | Coefficient | Word | Coefficient | Word | Coefficient |
| fight | 0.038 | holiday | 0.012 | united | 0.004 |
| win | 0.022 | celebrate | 0.009 | family | 0.003 |
| event | 0.019 | festival | 0.007 | party | 0.003 |
| champion | 0.015 | celebration | 0.007 | president | 0.003 |
| fighter | 0.015 | calendar | 0.006 | government | 0.003 |
| bout | 0.015 | church | 0.006 | school | 0.003 |
| title | 0.013 | united | 0.005 | American | 0.003 |
| Ali | 0.012 | date | 0.005 | university | 0.003 |
| championship | 0.011 | moon | 0.005 | city | 0.003 |
| bonus | 0.010 | event | 0.005 | attack | 0.003 |

| Cluster 4 | | Cluster 5 | | Cluster 6 | |
|---|---|---|---|---|---|
| Word | Coefficient | Word | Coefficient | Word | Coefficient |
| album | 0.031 | game | 0.023 | series | 0.029 |
| release | 0.016 | season | 0.020 | season | 0.027 |
| song | 0.015 | team | 0.018 | episode | 0.013 |
| music | 0.014 | win | 0.017 | character | 0.011 |
| single | 0.011 | player | 0.014 | film | 0.008 |
| record | 0.010 | play | 0.013 | television | 0.007 |
| band | 0.009 | league | 0.010 | cast | 0.006 |
| perform | 0.007 | final | 0.010 | announce | 0.006 |
| tour | 0.007 | score | 0.008 | release | 0.005 |
| chart | 0.007 | record | 0.007 | appear | 0.005 |

| Cluster 7 | | Cluster 8 | | Cluster 9 | |
|---|---|---|---|---|---|
| Word | Coefficient | Word | Coefficient | Word | Coefficient |
| match | 0.065 | film | 0.036 | film | 0.061 |
| win | 0.018 | star | 0.014 | million | 0.019 |
| championship | 0.016 | role | 0.014 | release | 0.013 |
| team | 0.015 | play | 0.010 | star | 0.010 |
| event | 0.015 | series | 0.009 | character | 0.006 |
| style | 0.014 | appear | 0.008 | role | 0.006 |
| raw | 0.013 | award | 0.008 | movie | 0.005 |
| title | 0.011 | actor | 0.007 | weekend | 0.005 |
| episode | 0.010 | character | 0.006 | story | 0.005 |
| perform | 0.010 | release | 0.006 | gross | 0.005 |

**Table 4.1** The 9 cluster representatives. For each representative we show the largest 10 coefficients of the word histogram.

| Cluster | Size | Titles |
|---|---|---|
| 1 | 21 | Floyd Mayweather, Jr; Kimbo Slice; Ronda Rousey; José Aldo; Joe Frazier; Wladimir Klitschko; Saul Álvarez; Gennady Golovkin; Nate Diaz; Conor McGregor. |
| 2 | 43 | Halloween; Guy Fawkes Night; Diwali; Hanukkah; Groundhog Day; Rosh Hashanah; Yom Kippur; Seventh-day Adventist Church; Remembrance Day; Mother's Day. |
| 3 | 189 | Mahatma Gandhi; Sigmund Freud; Carly Fiorina; Frederick Douglass; Marco Rubio; Christopher Columbus; Fidel Castro; Jim Webb; Genie (feral child); Pablo Escobar. |
| 4 | 46 | David Bowie; Kanye West; Celine Dion; Kesha; Ariana Grande; Adele; Gwen Stefani; Anti (album); Dolly Parton; Sia Furler. |
| 5 | 49 | Kobe Bryant; Lamar Odom; Johan Cruyff; Yogi Berra; José Mourinho; Halo 5: Guardians; Tom Brady; Eli Manning; Stephen Curry; Carolina Panthers. |
| 6 | 39 | The X-Files; Game of Thrones; House of Cards (U.S. TV series); Daredevil (TV series); Supergirl (U.S. TV series); American Horror Story; The Flash (2014 TV series); The Man in the High Castle (TV series); Sherlock (TV series); Scream Queens (2015 TV series). |
| 7 | 16 | Wrestlemania 32; Payback (2016); Survivor Series (2015); Royal Rumble (2016); Night of Champions (2015); Fastlane (2016); Extreme Rules (2016); Hell in a Cell (2015); TLC: Tables, Ladders & Chairs (2015); Shane McMahon. |
| 8 | 58 | Ben Affleck; Johnny Depp; Maureen O'Hara; Kate Beckinsale; Leonardo DiCaprio; Keanu Reeves; Charlie Sheen; Kate Winslet; Carrie Fisher; Alan Rickman. |
| 9 | 39 | Star Wars: The Force Awakens; Star Wars Episode I: The Phantom Menace; The Martian (film); The Revenant (2015 film); The Hateful Eight; Spectre (2015 film); The Jungle Book (2016 film); Bajirao Mastani (film); Back to the Future II; Back to the Future. |

**Table 4.2** Cluster sizes and titles of 10 documents closest to the cluster representatives.

The identification of these separate topics among the documents is impressive, when you consider that the $k$-means algorithm does not understand the meaning of the words in the documents (and indeed, does not even know the order of the words in each document). It uses only the simple concept of document dissimilarity, as measured by the distance between word count histogram vectors.

## 4.5 Applications

Clustering, and the $k$-means algorithm in particular, has many uses and applications. It can be used for exploratory data analysis, to get an idea of what a large collection of vectors 'looks like'. When $k$ is small enough, say less than a few tens, it is common to examine the group representatives, and some of the vectors in the associated groups, to interpret or label the groups. Clustering can also be used for more specific directed tasks, a few of which we describe below.

**Classification.** We cluster a large collection of vectors into $k$ groups, and label the groups by hand. We can now assign (or classify) *new* vectors to one of the $k$ groups by choosing the nearest group representative. In our example of the handwritten digits above, this would give us a rudimentary digit classifier, which would automatically guess what a written digit is from its image. In the topic discovery example, we can automatically classify new documents into one of the $k$ topics. (We will see better classification methods in chapter 14.)

**Recommendation engine.** Clustering can be used to build a *recommendation engine*, which suggests items that a user or customer might be interested in. Suppose the vectors give the number of times a user has listened to or streamed each song from a library of $n$ songs over some period. These vectors are typically sparse, since each user has listened to only a very small fraction of the music library. Clustering the vectors reveals groups of users with similar musical taste. The group representatives have a nice interpretation: $(z_j)_i$ is the average number of times users in group $j$ listened to song $i$.

This interpretation allows us to create a set of recommendations for each user. We first identify which cluster $j$ her music listening vector $x_i$ is in. Then we can suggest to her songs that she has not listened to, but others in her group (*i.e.*, those with similar musical taste) have listened to most often. To recommend 5 songs to her, we find the indices $l$ with $(x_i)_l = 0$, with the 5 largest values of $(z_j)_l$.

**Guessing missing entries.** Suppose we have a collection of vectors, with some entries of some of the vectors missing or not given. (The symbol '?' or '*' is sometimes used to denote a missing entry in a vector.) For example, suppose the vectors collect attributes of a collection of people, such as age, sex, years of education, income, number of children, and so on. A vector containing the symbol '?' in the age entry means that we do not know that particular person's age. Guessing missing entries of vectors in a collection of vectors is sometimes called