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

imputing the missing entries. In our example, we might want to guess the age of the person whose age we do not know.

We can use clustering, and the k-means algorithm in particular, to guess the missing entries. We first carry out k-means clustering on our data, using only those vectors that are complete, *i.e.*, all of their entries are known. Now consider a vector x in our collection that is missing one or more entries. Since some of the entries of x are unknown, we cannot find the distances $\|x - z_j\|$, and therefore we cannot say which group representative is closest to x. Instead we will find the closest group representative to x using only the known entries in x, by finding j that minimizes

$$\sum_{i \in \mathcal{K}} (x_i - (z_j)_i)^2,$$

where $\mathcal{K}$ is the set of indices for the known entries of the vector x. This gives us the closest representative to x, calculated using only its known entries. To guess the missing entries in x, we simply use the corresponding entries in z_j, the nearest representative.

Returning to our example, we would guess the age of the person with a missing age entry by finding the closest representative (ignoring age); then we use the age entry of the representative, which is simply the average age of all the people in that cluster.