Lecture topics:

- Mixture models and clustering, k-means
- Distance and clustering

Mixture models and clustering

We have so far used mixture models as flexible ways of constructing probability models for prediction tasks. The motivation behind the mixture model was that the available data may include unobserved (sub)groups and by incorporating such structure in the model we could obtain more accurate predictions. We are also interested in uncovering that group structure. This is a *clustering* problem. For example, in the case of modeling exam scores it would be useful to understand what types of students there are. In a biological context, it would be useful to uncover which genes are active (expressed) in which cell types when the measurements are from tissue samples involving multiple cell types in unknown quantities. Clustering problems are ubiquitous.

There are many different types of clustering algorithms. Some generate a series of nested clusters by merging simple clusters into larger ones (hierarchical agglomerative clustering), while others try to find a pre-specified number of clusters that best capture the data (e.g., k-means). Which algorithm is the most appropriate to use depends in part on what we are looking for. So it is very useful to know more than one clustering method.

Mixture models as generative models require us to articulate the type of clusters or subgroups we are looking to identify. The simplest type of clusters we could look for are spherical Gaussian clusters, i.e., we would be estimating Gaussian mixtures of the form

$$P(\mathbf{x}; \theta, m) = \sum_{j=1}^{m} P(j) N(\mathbf{x}; \mu_j, \sigma_j^2 I) \tag{1}$$

where the parameters θ include $\{P(j)\}$, $\{\mu_j\}$, and $\{\sigma_j^2\}$. Note that we are estimating the mixture models with a different objective in mind. We are more interested in finding where the clusters are than how good the mixture model is as a generative model.

There are many questions to resolve. For example, how many such spherical Gaussian clusters are there in the data? This is a model selection problem. If such clusters exist, do we have enough data to identify them? If we have enough data, can we hope to find the clusters via the EM algorithm? Is our approach robust, i.e., does our method degrade

gracefully when data contain "background samples" or impurities along with spherical Gaussian clusters? Can we make the clustering algorithm more robust? Similar questions apply to other clustering algorithms. We will touch on some of these issues in the context of each algorithm.

Mixtures and K-means

It is often helpful to try to search for simple clusters. For example, consider a mixture of two Gaussians where the variances of the spherical Gaussians may be different:

$$P(\mathbf{x};\theta) = \sum_{j=1}^{2} P(j)N(\mathbf{x};\mu_j,\sigma_j^2 I) \tag{2}$$

Points far away from the two means, in whatever direction, would always be assigned to the Gaussian with the larger variance (tails of that distribution approach zero much slower; the posterior assignment is based on the ratio of probabilities). While this is perfectly fine for modeling the density, it does not quite agree with the clustering goal.

To avoid such issues (and to keep the discussion simpler) we will restrict the covariance matrices to be all identical and equal to $\sigma^2 I$ where σ^2 is common to all clusters. Moreover, we will fix the mixing proportions to be uniform $P(j) = 1/m$. With such restrictions we can more easily understand the type of clusters we will discover. The resulting simplified mixture model has the form

$$P(\mathbf{x};\theta) = \frac{1}{m}\sum_{j=1}^{m} N(\mathbf{x};\mu_j,\sigma^2 I) \tag{3}$$

Let's begin by trying to understand how points are assigned to clusters within the EM algorithm. To this end, we can see how the mixture model partitions the space into regions where within each region a specific component has the highest posterior probability. To start with, consider any two components i and j and the boundary where $P(i|\mathbf{x},\theta) = P(j|\mathbf{x},\theta)$, i.e., the set of points for which the component i and j have the same posterior probability. Since the Gaussians have equal prior probabilities, this happens when $N(\mathbf{x};\mu_j,\sigma^2 I) = N(\mathbf{x};\mu_i,\sigma^2 I)$. Both Gaussians have the same spherical covariance matrix so the probability that they assign to points is based on the Euclidean distances to their mean vectors. The posteriors therefore can be equal only when the component means are equidistant from the points:

$$\|\mathbf{x}-\mu_i\|^2 = \|\mathbf{x}-\mu_j\|^2 \quad \text{or} \quad 2\mathbf{x}^T(\mu_j-\mu_i) = \|\mu_j\|^2 - \|\mu_j\|^2 \tag{4}$$

The boundary is therefore linear[1] in $\mathbf{x}$. We can draw such boundaries between any pair of components as illustrated in Figure 1. The pairwise comparisons induce a *Voronoi* partition of the space where, for example, the region enclosing μ_1 in the figure corresponds to all the points whose closest mean is μ_1. This is also the region where $P(1|\mathbf{x}, \theta)$ takes the highest value among all the posterior probabilities.

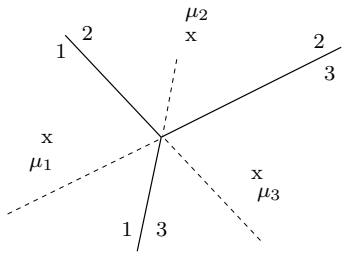

Figure 1: Voronoi regions resulting from pairwise comparison of posterior assignment probabilities. Each line corresponds to a decision between posterior probabilities of two components. The line is highlighted (solid) when the boundary is an active constraint for deciding the highest probability component.

Note that the posterior assignment probabilities $P(j|\mathbf{x}, \theta)$ evaluated in the E-step of the EM algorithm do not vanish across the boundaries. The regions merely highlight when one assignment has a higher probability than the others. The overall variance parameter σ^2 controls how sharply the posterior probabilities change when we cross each boundary. Small σ^2 results in very sharp transitions (e.g., from near zero to nearly one) while the posteriors change smoothly when σ^2 is large.

K-means. We can define a simpler and faster version of the EM algorithm by just assigning each point to the component with the highest posterior probability (its closest mean). Geometrically, the points within each Voronoi region are assigned to one component. The M-step then simply repositions each mean vector to the center of the points within the region. The updated means define new Voronoi regions and so on. The resulting algorithm for finding cluster means is known as the *K-means* algorithm:

E-step: assign each point $\mathbf{x}_t$ to its closest mean, i.e., $j_t = \arg\min_j \|\mathbf{x}_t - \mu_j\|^2$

M-step: recompute μ_j's as means of the assigned points.

[1]The linearity holds when the two Gaussians have the same covariance matrix, spherical or not.

This is a highly popular hard assignment version of the EM-algorithm. We can view the algorithm as optimizing the complete log-likelihood of the data with respect to the assignments $j_1, \ldots, j_n$ (E-step) as well as the means $\mu_1, \ldots, \mu_m$ (M-step). This is equivalent to minimizing the overall squared error (distortion) to the cluster means:

$$J(j_1, \ldots, j_n, \mu_1, \ldots, \mu_m) = \sum_{t=1}^{n} \|\mathbf{x}_t - \mu_{j_t}\|^2 \tag{5}$$

Each step of the algorithm, E or M-step, decreases the objective until convergence. The algorithm can never come back to the same assignments $j_1, \ldots, j_n$ as it would result in the same value of the objective. Since there are only $\mathcal{O}(n^k)$ possible partitions of points with k means, the algorithm has to converge in a finite number of steps. At convergence, the cluster means $\hat{\mu}_1, \ldots, \hat{\mu}_m$ are locally optimal solutions to the minimum distortion objective

$$J(\hat{\mu}_1, \ldots, \hat{\mu}_m) = \sum_{t=1}^{n} \min_j \|\mathbf{x}_t - \hat{\mu}_j\|^2 \tag{6}$$

The speed of the K-means algorithm comes at a cost. It is even more sensitive to proper initialization than a mixture of Gaussians trained with EM. A typical and reasonable initialization corresponds to setting the first mean to be a randomly selected training point, the second as the furthest training point from the first mean, the third as the furthest training point from the two means, and so on. This initialization is a greedy *packing* algorithm.

Identifiability. The k-means or mixture of Gaussians with the EM algorithm can only do as well as the maximum likelihood solution they aim to recover. In other words, even when the underlying clusters are spherical, the number of training examples may be too small for the globally optimal (non-trivial) maximum likelihood solution to provide any reasonable clusters. As the number of points increases, the quality of the maximum likelihood solution improves but may be hard to find with the EM algorithm. A large number of training points also makes it easier to find the maximum likelihood solution. We could therefore roughly speaking divide the clustering problem into three regimes depending on the number of training points: not solvable, solvable but hard, and easy. For a recent discussion on such regimes, see Srebro et al., "An Investigation of Computational and Informational Limits in Gaussian Mixture Clustering", ICML 2006.

Distance and clustering

Gaussian mixture models and the K-means algorithm make use of the Euclidean distance between points. Why should the points be compared in this manner? In many cases the vector representation of objects to be clustered is derived, i.e., comes from some feature transformation. It is not at all clear that the Euclidean distance is an appropriate way of comparing the resulting feature vectors.

Consider, for example, a document clustering problem. We might treat each document $\mathbf{x}$ as a bag of words and map it into a feature vector based on term frequencies:

$$n_w(\mathbf{x}) = \text{number of times word } w \text{ appears in } \mathbf{x} \tag{7}$$

$$f(w|\mathbf{x}) = \frac{n_w(\mathbf{x})}{\sum_{w'} n_{w'}(\mathbf{x})} = \text{term frequency} \tag{8}$$

$$\phi_w(\mathbf{x}) = f(w|\mathbf{x}) \cdot \overbrace{\log\left[\frac{\#\text{ of docs}}{\#\text{ of docs with word } w}\right]}^{\text{IDF}} \tag{9}$$

where the feature vector $\phi(\mathbf{x})$ is known as the *TFIDF* mapping (there are many variations of this). IDF stands for *inverse document frequency* and aims to de-emphasize words that appear in all documents, words that are unlikely to be useful for clustering or classification. We can now interpret the vectors $\phi(\mathbf{x})$ as points in a Euclidean space and apply, e.g., the K-means clustering algorithm. There are, however, many other ways of defining a distance metric for clustering documents.

Distance metric plays a central role in clustering, regardless of the algorithm. For example, a simple hierarchical agglomerative clustering algorithm is defined almost entirely on the basis of the distance function. The algorithm proceeds by successively merging two closest points or closest clusters (average squared distance) and is illustrated in Figure 2.

In solving clustering problems, the focus should be on defining the distance (similarity) metric, perhaps at the expense of a specific algorithm. Most algorithms could be applied with the chosen metric. One avenue for defining a distance metric is through model selection. Let's see how this can be done. The simplest model over the words in a document is a *unigram* model where each word is an independent sample from a multi-nomial distribution $\{\theta_w\}$, $\sum_w \theta_w = 1$. The probability of all the words in document $\mathbf{x}$ is therefore

$$P(\mathbf{x}|\theta) = \prod_{w \in \mathbf{x}} \theta_w = \prod_w \theta_w^{n_w(\mathbf{x})} \tag{10}$$

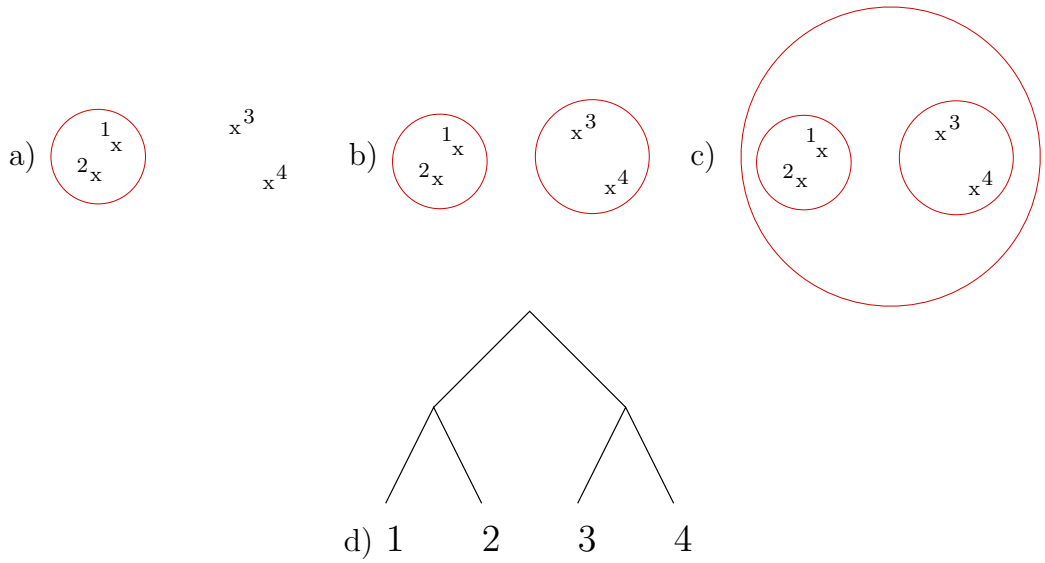

Figure 2: a-c) successive merging of clusters in a hierarchical clustering algorithm and d) the resulting cluster hierarchy.

It is often a good idea to normalize the documents so they have the same length. We could, for example, say that each document is of length one and word "counts" are given by term frequencies $f(w|\mathbf{x})$. Accordingly,

$$P(\tilde{\mathbf{x}}|\theta) = \prod_w \theta_w^{f(w|\mathbf{x})} \tag{11}$$

where $\tilde{\mathbf{x}}$ denotes a normalized version of document $\mathbf{x}$. Consider a cluster of documents C. The unigram model that best describes the normalized documents in the cluster can be obtained by maximizing the log-likelihood

$$l(C;\theta) = \sum_{\mathbf{x}\in C} \log P(\tilde{\mathbf{x}}|\theta) = \sum_{\mathbf{x}\in C}\sum_w f(w|\mathbf{x}) \log \theta_w \tag{12}$$

The maximum likelihood solution is, as before, obtained through empirical counts (represented here by term frequencies):

$$\hat{\theta}_w = \frac{1}{|C|}\sum_{\mathbf{x}\in C} f(w|\mathbf{x}) \tag{13}$$

The resulting log-likelihood value is

$$
\begin{aligned}
l(C;\hat{\theta}) &= \sum_{\mathbf{x}\in C}\sum_{w} f(w|\mathbf{x})\log\hat{\theta}_w && (14)\\
&= |C|\sum_{w}\frac{1}{|C|}\sum_{\mathbf{x}\in C} f(w|\mathbf{x})\log\hat{\theta}_w && (15)\\
&= |C|\sum_{w}\hat{\theta}_w\log\hat{\theta}_w = -|C|H(\hat{\theta}) && (16)
\end{aligned}
$$

where $H(\hat{\theta})$ is the entropy of the unigram distribution.

We can now proceed to define a distance corresponding to the unigram model. Consider any two clusters C_i and C_j and their combination $C = C_i \cup C_j$. When C_i and C_j are treated as distinct clusters, we will use a different unigram model to capture their term frequencies. The combined cluster C, on the other hand, is modeled with a single unigram model. We can now treat the documents in the two clusters as observations and devise a model selection problem for deciding whether the clusters should be viewed as separate or merged. To this end, we will evaluate the resulting log-likelihoods in two ways corresponding to whether the clusters are modeled as separate or combined. When separate:

$$
\begin{aligned}
l(C_i|\hat{\theta}_{\cdot|i}) + l(C_j|\hat{\theta}_{\cdot|j}) &= -|C_i|H(\hat{\theta}_{\cdot|i}) - |C_j|H(\hat{\theta}_{\cdot|j}) && (17)\\
&= -(|C_i| + |C_j|)\sum_{y\in\{i,j\}}\hat{P}(y)H(\hat{\theta}_{\cdot|y}) && (18)
\end{aligned}
$$

where $\hat{P}(y=i) = |C_i|/(|C_i| + |C_j|)$ is the probability that a randomly drawn document from $C = C_i \cup C_j$ is from cluster C_i. The parameter estimates $\hat{\theta}_{w|i}$ and $\hat{\theta}_{w|j}$ are obtained as before. Similarly, when the two clusters are modeled with the same unigram:

$$
l(C_i, C_j|\hat{\theta}) = -(|C_i| + |C_j|)H(\hat{\theta}) \qquad (19)
$$

where the parameter estimate $\hat{\theta}_w$ can be written as $\hat{\theta}_w = \sum_{y\in\{i,j\}}\hat{P}(y)\hat{\theta}_{w|y}$, i.e., as a cluster size weighted combination of the estimates from the individual clusters.

We can now define a (squared) distance between C_i and C_j according to how much better they are modeled as separate clusters (log-likelihood ratio statistic):

$$
\begin{aligned}
d^2(C_i, C_j) &= l(C_i|\hat{\theta}_{\cdot|i}) + l(C_j|\hat{\theta}_{\cdot|j}) - l(C_i, C_j|\hat{\theta}) && (20)\\
&= (|C_i| + |C_j|)\left[H(\hat{\theta}) - \sum_{y\in\{i,j\}}\hat{P}(y)H(\hat{\theta}_{\cdot|y})\right] && (21)\\
&= (|C_i| + |C_j|)\,\hat{I}(w;y) && (22)
\end{aligned}
$$

where $\hat{I}(w; y)$ is the mutual information between words w sampled from the combined cluster C and the identity of the cluster (i or j). (Recall a similar derivation in the feature selection context). More precisely, the mutual information is computed on the basis of $\hat{P}(w, y) = \hat{P}(y)\hat{\theta}_{w|y}$. Note that $d^2(C_i, C_j)$ is symmetric and non-negative but need not satisfy all the properties of a metric. It nevertheless compares the two clusters in a very natural way: we measure how distinguishable they are based on their word distributions. In other words, $\hat{I}(w; y)$ tells us how much a word sampled at random from a document in the combined cluster C tells us about the cluster C_i or C_j that the sample came from. Low information content means that the two clusters have nearly identical distributions over words.

The (squared) metric $d^2(C_i, C_j)$ can be immediately used, e.g., in a hierarchical clustering algorithm (the same algorithm derived from a different perspective is known as an agglomerative information bottleneck method). We could take the model selection idea further and decide when to stop merging clusters rather than simply providing a scale for deciding how different two clusters are.