Mixture Models

A mixture model is a probabilistic model for representing the presence of subpopulations within an overall population.

The discussion of k-means elucidated the difficulty in picking the number of clusters. If k is small, then clusters are heterogeneous. If k is too large, then similar documents are split between clusters. While there exists a heuristic to helping to pick k, there is no precise algorithmic method for picking k. So we know that picking the number of clusters is difficult, but we also know that the cluster assignments are 'hard'; a document can only be assigned to one cluster. This seems like a problem, given that we know that cluster assignment is imprecise. These hard assignments don't provide the full story.

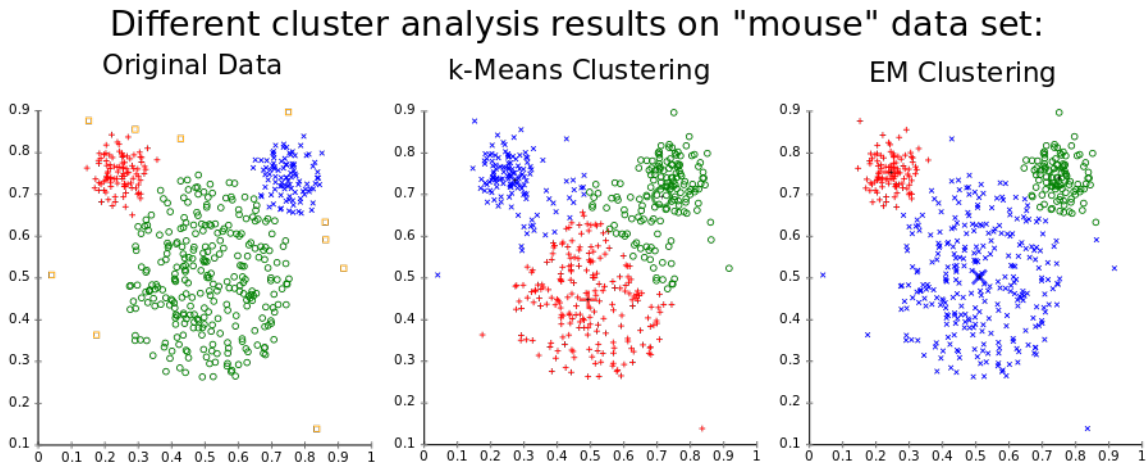
There are other methods for clustering that provide a probabilistic assignment to cluster.

K-means also does not take the cluster shape into account. The vanilla algorithm uses un-weighted Euclidean distance and effectively assumes spherically symmetric cluster shapes around a center. It is possible to use weights for features and so essentially change the shape of the cluster, but this affects all clusters in the same way, rather than letting each cluster be sized and shaped appropriately. In addition, the weights would have to be pre-specified – we would somehow have to know the cluster shape ahead of doing the clustering.

So k-means does not handle these situations well

- Clusters that have different spreads
  (Here we are not referring to the number of points in a clusters, but the range of points in the cluster)
- Overlapping clusters
- Clusters with different shapes

To illustrate the issue with spread, there is a data set called the ‘mouse’ data set, with one large cluster and two smaller clusters.



Taken from Wikipedia.

k-means, because it uses Euclidean distance from a center, will make the small clusters larger than they should be.

With overlapping clusters, k-means still makes a hard cluster assignment. In reality, in overlapping areas, the observation exists in more than one cluster, but k-means cannot model this. What would be more appropriate is to model the uncertainty.

Mixture models address these problems by:

- Providing soft assignment (probabilistic assignment) to clusters. So we can say a document has a 60% chance of being in cluster A and a 40% change of being in cluster B.
- Modeling cluster shape, not just center
- Allow for learning the weightings of each dimension per cluster.

Aggregating over unknown classes in an image dataset

For a motivating application, we will be talking about categorizing photos. The data that we will use is simplistic; the overall red, green and blue color components in the photo. Within the corpus of

photos we have subpopulations like pictures of sunsets, pictures of blue sky and pictures of forests.

So a sunset will have a lot of red. A picture of the sky will have a lot of blue. A picture of a forest will have a lot of green. So intuitively, we should be able to look at the three color components of a picture and put them into one of these categories. However, some photos will not be so easy; a photo of a forest may include some blue sky.

Most importantly remember that in real applications we are doing unsupervised learning; we don't know what kinds are clusters are in the corpus to begin with – we want to discover them. The photo application ties our intuition about photos and their colors (something we can easily see) to the operation of the algorithm, but we should not get confused; in real clustering applications we don't start with labeled data, instead we seek to learn the subpopulations inherent in the data.

Univariate Gaussian distributions

Within the corpus of photos, we assume a normal (Gaussian) distribution (a bell shaped distribution) of each of these 3 components in any subpopulation.

A 1D (Univariate) Gaussian distribution is a probability distribution of a scalar variable along one axis. It is defined by its center, called the mean (μ), and it's spread. Spread is specified by variance (σ^2) or the standard deviation (σ) which is the square root of the variance. The normal distribution for a random variable (x) is notated as:

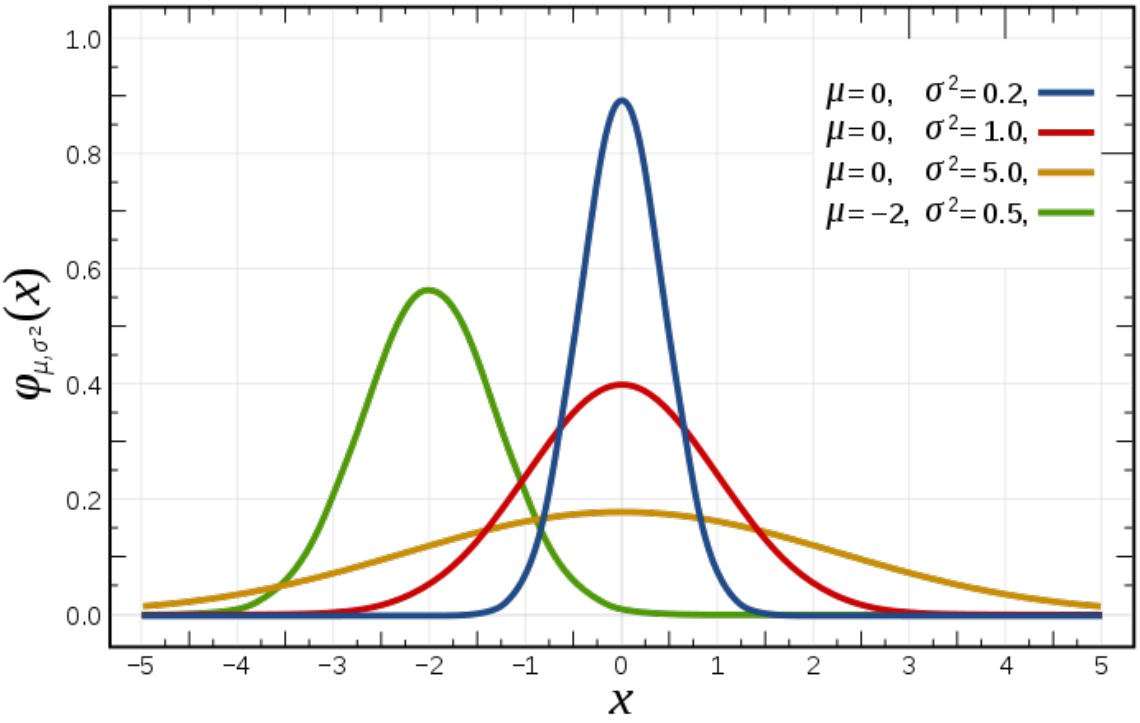
$$N(x \mid \mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-\mu)^2}{2\sigma^2}}$$

where:

x the random variable

μ	mean of the distribution. Also it's median, mode and expected value, since the normal distribution is symmetrical around the mean.
σ^2	Variance of the distribution
σ	Standard deviation (square root of the variance)
$N(x \mid \mu, \sigma^2)$	the normal distribution of random variable x, given mean and variance

A larger variance makes the distribution wider and flatter. Moving the mean moves the center of the distribution along the axis.

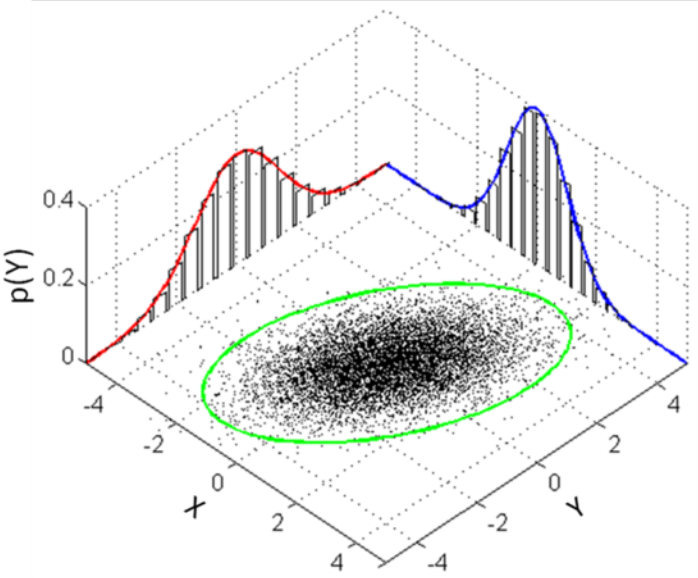


Examples of Normal distributions with different mean and variance, from Wikipedia

Here is a live example in Desmos that let's you change the mean and variance to see the result; https://www.desmos.com/calculator/2kmx0enkkz.

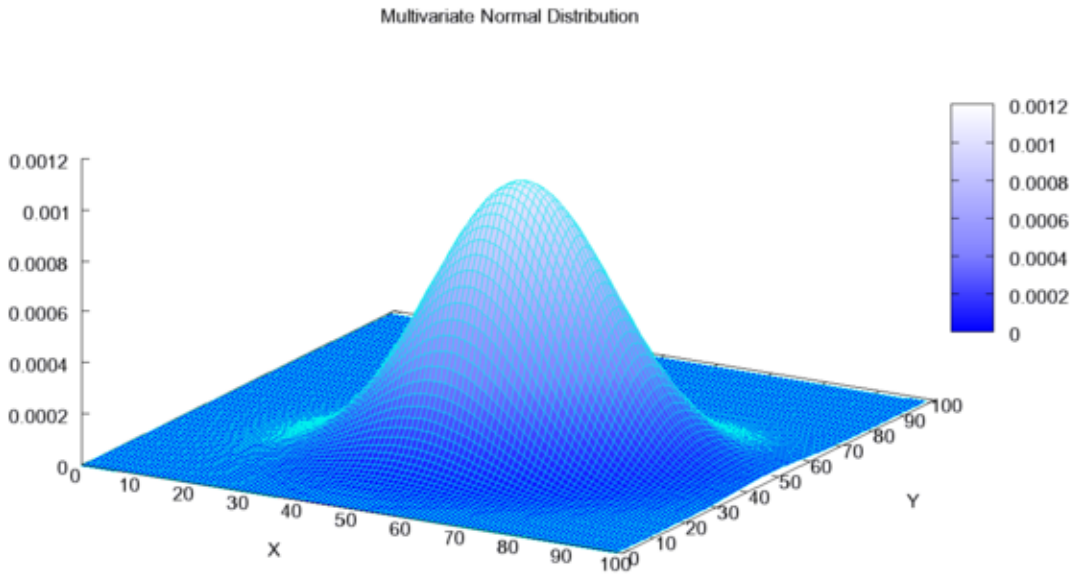
Bivariate and multivariate Gaussians

A Bivariate Gaussian is a generalization of the Univariate Gaussian to 2 dimensions. Each of the two dimensions is itself normally distributed. This can be seen below.



Bivariate Normal Distributions is a linear combination of Univariate Normal Distributions. From Wikipedia

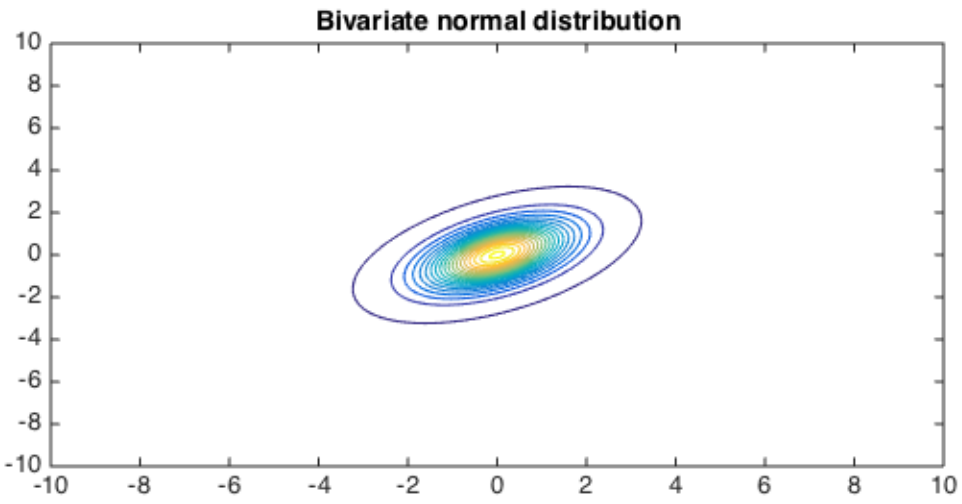
The resulting linear combination of the two univariate normal distributions along orthogonal axis produces a 'hump' with an elliptical base and a peak at it's center.



Bivariate Normal Distribution as a mesh plot, from Wikipedia

By looking at these two illustrations, we can see that the mean on both axis creates a peak; this then is the most likely expected value in the probability distribution (since it is the most likely expected value of both along both axis).

In our application, we will look 'down' on this distribution (perpendicular to both axis) using contour plots where contour lines are created along regions of equal probability. This can be seen in the illustration below.



From Alex Townsend on Chebfun.org

The center of the ellipse is the highest probability area (the area where the two random variables are most likely to be valued).

The generalization of univariate Gaussian distribution to 2 dimensions changes the notation slightly. The center of the distribution is now specified by a mean vector, which is a 2d vector. The spread of the distribution (which now includes spread in two dimensions and orientation) is specified by the Covariance Matrix, which is a 2x2 matrix. **A bivariate normal distribution is given by**:

$$N(x \mid \mu, \Sigma) = \frac{1}{2\pi\sqrt{|\Sigma|}} \times exp\left(-\frac{1}{2}(x-\mu)^T\Sigma^{-1}(x-\mu)\right)$$

$$\mu = (\mu 1, \mu 2)$$

$$\Sigma = \begin{bmatrix} {\sigma_1}^2 & \sigma_{1,2} \\ \sigma_{2,1} & {\sigma_2}^2 \end{bmatrix}$$

$N(x\|\mu, \Sigma)$	the normal distribution for random vector x given the mean vector μ and the covariance matrix Σ.
x	the observed value; a 2d vector
μ	mean of the distribution. This specifies the center of the distribution as a 2d vector. It is the point of highest likelihood in the distribution. It can be thought of as the intersection of the 2 axis aligned lines specified by the vector components $\mu = (\mu_1, \mu_2)$ where μ_1 is the mean of the first component and μ_2 is the mean of the second component. In our example μ_1 might be the mean value for blue in set of photos and μ_2 might be the mean value for green in the set of photos.
n	the number of observations
$\sigma_{1,2}$	Covariance between μ_1 and μ_2
Σ	Covariance matrix. This is a 2x2 matrix for a bivariate distribution.
$\|\Sigma\|$	the determinant of the covariance matrix
d	the dimension of the space of x; in the bivariate case this is 2.

Note that the matrix is symmetrical because $\sigma_{1,2} = \sigma_{2,1}$

$$\sigma_{x,y} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)$$

where:

$\sigma_{x,y}$	covariance between dimension x and dimension y
x_i	i-th obseration in dimension x (or dimension x of i-th observation) (*don't be confused with prior use of x as the entire observation vector, here we just mean this is a value from one of the dimensions of the observation vector*)
y_i	i-th obseration in dimension y (or dimension y of i-th observation)
μ_x	mean in dimension x of observations
μ_y	mean in dimension y of observations

n number of observations

Note: don't get confused with the previous use of x as the entire 2-dimensional observation; in this case we are using x as just one of the dimensions and y as the other.

From Wikipedia:
In probability theory and statistics, a covariance matrix (also known as dispersion matrix or variance–covariance matrix) is a matrix whose element in the i, j position is the covariance between the i th and j th elements of a random vector. A random vector is a random variable with multiple dimensions.

Intuitively, the covariance matrix generalizes the notion of variance to multiple dimensions. As an example, the variation in a collection of random points in two-dimensional space cannot be characterized fully by a single number, nor would the variances in the x and y directions contain all of the necessary information; a 2×2 matrix would be necessary to fully characterize the two-dimensional variation.

Because the covariance of the ith random variable with itself is simply that random variable's variance, each element on the principal diagonal of the covariance matrix is the variance of one of the random variables. Because the covariance of the ith random variable with the jth one is the same thing as the covariance of the jth random variable with the ith one, every covariance matrix is symmetric. In addition, every covariance matrix is positive semi-definite.

Covariance Structures
The structure of the Covariance matrix affects the shape of the resulting probability distributions and can be described as an effect on the contour plot of the distribution.

$$\Sigma = \begin{bmatrix} \sigma^2 & 0 \\ 0 & \sigma^2 \end{bmatrix}$$

Circular Covariance Structure

Here the variance of each random variable is exactly the same ($\sigma_1^2 = \sigma_2^2$) and there is no correlation between them ($\rho = 0$). When the first random variable is high, it does not affect the probability of the second random variable being high or low at all. Because the variance of each random variable is the same, so the spread on each axis is exactly the same, a contour plot of a bivariate distribute with this kind of covariance would show concentric circles.

$$\Sigma = \begin{bmatrix} {\sigma_1}^2 & 0 \\ 0 & {\sigma_2}^2 \end{bmatrix}$$

Axis Aligned Ellipse Covariance Structure

Here each random variable has a different spread, but those spreads are completely independent of each other (there are zeroes in the upper right and lower left, indicating there is no correlation between the two variables). This would allow different spreads along each axis. A contour plot of a bivariate distribution with this kind of covariance would results in an axis aligned ellipse.

$$\Sigma = \begin{bmatrix} {\sigma_1}^2 & \sigma_{1,2} \\ \sigma_{2,1} & {\sigma_2}^2 \end{bmatrix}$$

Arbitrary Ellipse Covariance Structure

Here the random variables are correlated in some way. If they are positively correlated, then when the first random variable is high, then it is more likely that the second random variable is high and when the first is low, it is more likely that the second is low. In a negative correlation, when the first is high, the second is more likely to be low and visa versa. A positively correlated distribution would produce a contour plot with an ellipse whose axis goes from lower-left to upper-right. A negatively correlated distribution would produce a contour plot with an ellipse whose axis goes from upper-left to lower-right.

This extension of univariate Gaussian distribution specified by it's mean and variance to a bivariate Gaussian distribution specified by a mean vector and covariance matrix can be further generalized into d dimensions. A d-dimensional multivariate Gaussian is specified by a d-dimensional mean vector and a dxd covariance matrix. Specifically, the normal distribution for d-dimensional random variables is given by:

$$N(x \mid \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \times exp\left(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right)$$

where

$\|\Sigma\|$	the determinant of the covariance matrix
d	the dimension of the space of x; in the bivariate case this is 2.

Aside: This component of the multivariate Gaussian distribution:

$$(x- \mu)^T \Sigma^{-1}(x- \mu)$$

is the square of the Mahalanobis distance (also called generalized squared interpoint distance). It is a measure of distance between a point and a distribution.

The Mahalanobis distance of an observation $x = [x_1, x_2, \ldots, x_N]$ from a set of observations with mean $\mu = [\mu_1, \mu_1, \ldots, \mu_N]$ is defined as:

$$D_M(x) = \sqrt{(x - \mu)^T \Sigma^{-1} (x - \mu)}$$

This can also be defined as a dissimilarity measure between to random vectors x and y in the same distribution:

$$d(x, y) = \sqrt{(x - y)^T \Sigma^{-1} (x - y)}$$

So the Mahalanobis distance (and it's square) is a useful distance/similarity metric between a distribution and an observation or between observations within a distribution.

In our example of specifying photos using their overall red, green and blue color components, the mean vector would have 3 elements and the covariance matrix would be a 3x3 matrix. The mean vector elements would be the mean for red in the corpus, the mean for green in the corpus and the mean for blue in the corpus. The covariance matrix would have a diagonal where [1,1] is the variance for red in the corpus, [2,2] is the variance for green in the corpus and [3,3] is the variance for blue in the corpus. The other elements specify the correlations between these elements.

Mixtures of Gaussians for Clustering

A Mixture of Gaussians is a special case of a mixture model.

If we look at the histogram of one variable across all observations in our corpus (for instance, just look at the blue component, effectively taking a slice out of the distribution density function along the blue axis), then we will get some complex shape that represents the histogram on that axis. Our goal is to model that histogram as a set of Gaussians that are combined together using a weighted average such that the overall probability function still integrates to 1; the sum of all probabilities is 1.

So we find Gaussians that when combined using a weighted average best model the data. Each Gaussian represents a cluster. The weights are then called cluster weights, 1 per cluster (one per Gaussian). Each cluster weight must be between zero and one inclusive and the sum of the cluster weights must be one. For K clusters;

$$\pi = [\pi_{1,} \ldots \pi_{k}]$$

$$0 \leq \pi_{k} \leq 1$$

$$\sum_{k=1}^{K} \pi_k = 1$$

π vector of cluster weights

π_k weight of the k^{th} cluster in the mixture model. This is based on the contribution of the cluster to the overall probability (basically, the proportion of this cluster in the whole)

K number of clusters (Gaussians) in the mixture model

Each cluster represents a Mixture component specified by the cluster weight, cluster mean and cluster covariance matrix, also called the cluster parameters:

$$\{\pi_k, \mu_k, \Sigma_k\}$$

Expectation Maximization (EM) Building Blocks

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters comprising the means and covariances of the components and the mixing coefficients). The expectation maximization algorithm proceeds after initialization by iterating over two steps until convergence:

0. Initialize the cluster parameters (means, covariances and mixing coefficients)
1. Estimation step: Calculate the soft cluster assignments (the cluster responsibilities) using the current cluster parameter values
2. Maximization Step: Maximize the likelihood over the cluster parameters; estimate the cluster parameters using the current soft cluster assignments.
3. Compare the log Likelihood for the estimated cluster parameters to the previous cluster parameters. If there is no convergence, repeat steps 1, 2 and 3.

Calculating the soft cluster assignments given the parameter values

We can compute soft assignments from known cluster parameters. First, let's assume we know the cluster parameters for each cluster;

$$\{\pi_k, \mu_k, \Sigma_k\}$$

Given the cluster parameters for K clusters, and the observed values, the soft cluster assignment of observation i to the K clusters is quantified using a responsibility vector:

$$r_i = [r_{i1}, r_{i2}, \ldots, r_{iK}]$$

where each element, r_{ik}, is the responsibility that the kth cluster takes for the ith observation.

$$r_{ik} = p\left(z_i = k \middle| \{\pi_j, \mu_j, \Sigma_j\}_{j=1}^{K}, x_i\right)$$

where

r_{ik}	The responsibility that cluster k takes for observation i. The responsibility corresponds to the certainty that the observation came from that cluster.
$z_i = k\|$	The random variable that the distribution is over; the probability of assignment of observation i to cluster k given the following fixed values the define the probability distribution
$\{\pi_j, \mu_j, \Sigma_j\}_{j=1}^{K}$	The given probability distributions (the cluster parameters for the K clusters)
x_i	The given observation

Each element of the responsibility vector is between zero and one and the total responsibility across all clusters for a given observation sums to 1 (since it is 1 observation);

$$1 = r_{i1} + r_{i2} + \cdots + r_{iK}$$

So this says that responsibility that cluster k takes for observation i equals the probability of assignment of observation i to cluster k given the cluster parameters for the K clusters and the observation itself. But it does not yet give a direct way to calculate the responsibility vector from the parameters, but there is a way to do that.

There is intuition behind the notion of which cluster an observation should be assigned to. Given two clusters of equal size, if an observation is closer to one than the other, then the closer of the two clusters takes more responsibility. If an observation is exactly between two equal size clusters, then responsibility is split. However, if an observation is exactly between to different size

clusters, then the mass of the cluster weighs of the probability such that it is more likely that the observation belongs to the large cluster than the small cluster.

So when calculating the responsibility vector, we need to weigh the prior probability of observation i being from cluster k (or in the first step, the initial probability of observation I being from cluster k), with how likely the observed value is under this cluster assignment (how likely it is to see such a value in the given cluster).

$$r_{ik} = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \, N\left(x_i|\mu_j, \Sigma_j\right)}$$

where

π_k	Prior (or initial) probability of observation being in cluster k
$N(x_i\|\mu_k, \Sigma_k)$	Likelihood of the observed value x_i under this cluster assignment.
$\sum_{j=1}^{K} \pi_j \, N\left(x_i\|\mu_j, \Sigma_j\right)$	All possible cluster assignments (used to normalize such that the probability is between zero and one.)

Prior Probability Term: Without knowing anything about an observation; what is the chance that it is from cluster k? This is given by the prior probability term;

$$p(z_i = k) = \pi_k$$

Likelihood Term: Given that observation x_i is from cluster k; what is the likelihood of seeing an observation with those features in cluster k? This is given by the likelihood term:

$$p(x_i|z_i = k, \mu_k, \Sigma_k) = \; N(x_i|\mu_k, \Sigma_k)$$

So, given cluster parameters and an observation; computing soft cluster assignments for the observation (the responsibility of each cluster for the observation) is straightforward:

- For each one of the k clusters, compute the prior probability term times the likelihood term.
- Calculate the sum of all of these.
- Normalize each of the vectors from step one using the sum of the vectors from step two. This creates a set of responsibility vectors, one per cluster, such that the total responsibility is one and each individual responsibility is between zero and 1.

So if we know the cluster parameters, we can calculate the cluster assignments for an observation. However, we don't actually know the cluster parameters. That's something that we have to infer from the unlabeled observations.

Estimating cluster parameters from hard cluster assignments

If we start with hard cluster assignments, then assigned observations from one cluster have no influence on the parameters of the other clusters. This decouples the clusters – we only need to take into account the observations assigned to the cluster. Below is an example using our RGB model for photos where we split 6 observations among 3 clusters.

observation	R	G	B	cluster
x1	x1[1]	x1[2]	x1[3]	3
x2	x2[1]	x2[2]	x2[3]	3
x3	x3[1]	x3[2]	x3[3]	3
x4	x4[1]	x4[2]	x4[3]	1
x5	x5[1]	x5[2]	x5[3]	2
x6	x6[1]	x6[2]	x6[3]	2

Hard Assignments

We will use a Maximum Likelihood estimation, which searches over all settings (cluster parameters in this case) to find the setting that

maximizes the likelihood of our observed data (the hard cluster assignments) under the specified model (the mixture of gaussians). Using ML estimation, we estimate the mean and covariance matrix.

$$\hat{\mu}_k = \frac{1}{N_k} \sum_{i \; in \; k} x_i$$

$$\hat{\Sigma}_k = \frac{1}{N_k} \sum_{i \; in \; k} (x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

where

$\hat{\mu}_k$	estimated mean vector
$\hat{\Sigma}_k$	estimated covariance matrix
$\hat{\pi}_k$	the cluster proportion – the initial estimated probability of any observation being in cluster k
N_k	number of observations in cluster k
N	total number of observations in the data
x_i	the ith observation in cluster k
$\sum_{i \; in \; k} x_i$	the sum of the observations in cluster k

The estimate of the initial probability of an observation being in a cluster is simply the proportion of observations that we have assigned to the cluster. This is often called the cluster proportion for short.

$$\hat{\pi}_k = \frac{N_k}{N}$$

where

$\hat{\pi}_k$ the cluster proportion – the initial estimated probability of any observation being in cluster k

N_k number of observations in cluster k

N total number of observations in the data

The summation for "all i in k" is the switch that only includes those observations with hard assignments to the given cluster k. So it makes it like we are doing this calculation independently on 3 separate cluster tables.

observation	R	G	B	cluster
x1	x1[1]	x1[2]	x1[3]	3
x2	x2[1]	x2[2]	x2[3]	3
x3	x3[1]	x3[2]	x3[3]	3

observation	R	G	B	cluster
x4	x4[1]	x4[2]	x4[3]	1

observation	R	G	B	cluster
x5	x5[1]	x5[2]	x5[3]	2
x6	x6[1]	x6[2]	x6[3]	2

Estimating cluster parameters from soft cluster assignments

How does this change is we have known soft assignments (responsibilities) rather than hard assignments? Now, rather than an observation being in only one cluster, we build a model that assumes that the observation is split between all clusters in some way. Instead of putting the observation in a single cluster, some fraction of the observation, between zero and one, is assigned to each cluster, with the total assignment across all clusters being exactly one. Below is an example using our RGB model for photos where we split 6 observations among 3 clusters.

observation	R	G	B	ri1 (cluster1)	ri2 (cluster2)	ri3 (cluster3)	
x1	x1[1]	x1[2]	x1[3]	0.3	0.18	0.52	1.00
x2	x2[1]	x2[2]	x2[3]	0.01	0.26	0.73	1.00
x3	x3[1]	x3[2]	x3[3]	0.002	0.008	0.99	1.00
x4	x4[1]	x4[2]	x4[3]	0.75	0.1	0.15	1.00
x5	x5[1]	x5[2]	x5[3]	0.05	0.93	0.02	1.00
x6	x6[1]	x6[2]	x6[3]	0.13	0.86	0.01	1.00
				1.242	2.338	2.42	6.00

Soft Assignment Responsibilities

Note that each row sums to one; so the observation is split between the 3 nodes based on the responsibility vector. The totals at the bottom are the effective number of observations in each cluster and these totals sum to the total number of observations.

Now when we estimate the cluster parameters for a given cluster, we use all of the observations. Each row calculation is weighted by the cluster's entry in the observation's responsibility vector. For the first cluster, we can form a table to do the calculations that looks like this:

observation	R	G	B	ri1 (cluster1)
x1	x1[1]	x1[2]	x1[3]	0.3
x2	x2[1]	x2[2]	x2[3]	0.01
x3	x3[1]	x3[2]	x3[3]	0.002
x4	x4[1]	x4[2]	x4[3]	0.75
x5	x5[1]	x5[2]	x5[3]	0.05
x6	x6[1]	x6[2]	x6[3]	0.13
				1.242

Soft Assignment Cluster 1

The calculation of the cluster parameters that were used for the hard cluster assignment model modified slightly to use soft assignments:

- include all observations in the cluster

- apply the cluster responsibility to each observation as a weighting factor
- use the effective cluster weight to normalize

The soft count is the effective number of observations in cluster k; the sum of all cluster responsibilities for that cluster:

$$N^{soft}{}_k = \sum_{i=1}^{N} r_{ik}$$

The responsibility of cluster k for observation i is the average of all data points weighted by the cluster responsibilities, divided by the effective number of observation in the cluster:

$$\hat{\mu}_k = \frac{1}{N^{soft}{}_k} \sum_{i=1}^{N} r_{ik} x_i$$

The covariance of each cluster is set to the weighted average of all outer products, weighted by the cluster responsibilities. The "outer product" in this context refers to the matrix product $(xi\text{-}\mu k)(xi\text{-}\mu k)^T$, Letting $(xi\text{-}\mu k)$ to be k×1 column vector, this product is a k×k matrix. Taking the weighted average of all outer products gives us the covariance matrix, which is also k×k:

$$\hat{\Sigma}_k = \frac{1}{N^{soft}{}_k} \sum_{i=1}^{N} r_{ik}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

where (for 3 equations above)

$\hat{\mu}_k$	estimated mean vector of length k
$\hat{\Sigma}_k$	estimated covariance matrix
r_{ik}	the cluster responsibility – the responsibility of cluster k for observation i
$N^{soft}{}_k$	the soft count; the effective number of observations in cluster k; the sum of all cluster responsibilities for that cluster
N	total number of observations in the data
x_i	the ith observation

Likewise, the calculation of the cluster proportion (the cluster weight) is modified; the weight of cluster k is given by the ratio of the soft count to the total number of data points:

$$\hat{\pi}_k = \frac{N^{soft}{}_k}{N}$$

Notice that N is equal to the sum over the soft counts of all clusters. The cluster weights show us how much each cluster is represented over all data points.

This model is consistent with our hard assignments model if we give full responsibility for an observation to a single cluster only. We model a hard assignment as a responsibility of 1.0, where the non-assigned clusters have a responsibility of 0.0. When we do this, the calculation of cluster proportion, mean and covariance simplify to the hard assignment calculation.

$$r_{ik} = \begin{cases} 1 & i \; in \; k \\ 0 & otherwise \end{cases}$$

observation	R	G	B	ri1 (cluster1)	ri2 (cluster2)	ri3 (cluster3)	
x1	x1[1]	x1[2]	x1[3]	0	0	1	1.00
x2	x2[1]	x2[2]	x2[3]	0	0	1	1.00
x3	x3[1]	x3[2]	x3[3]	0	0	1	1.00
x4	x4[1]	x4[2]	x4[3]	1	0	0	1.00
x5	x5[1]	x5[2]	x5[3]	0	1	0	1.00
x6	x6[1]	x6[2]	x6[3]	0	1	0	1.00
				1	2	3	6.00

Hard Assignment as Responsibilities

The Expectation Maximization (EM) Algorithm

Now we can fill in two of the EM algorithm steps with the necessary math. The expectation maximization algorithm proceeds after initialization by iterating over two steps until convergence:

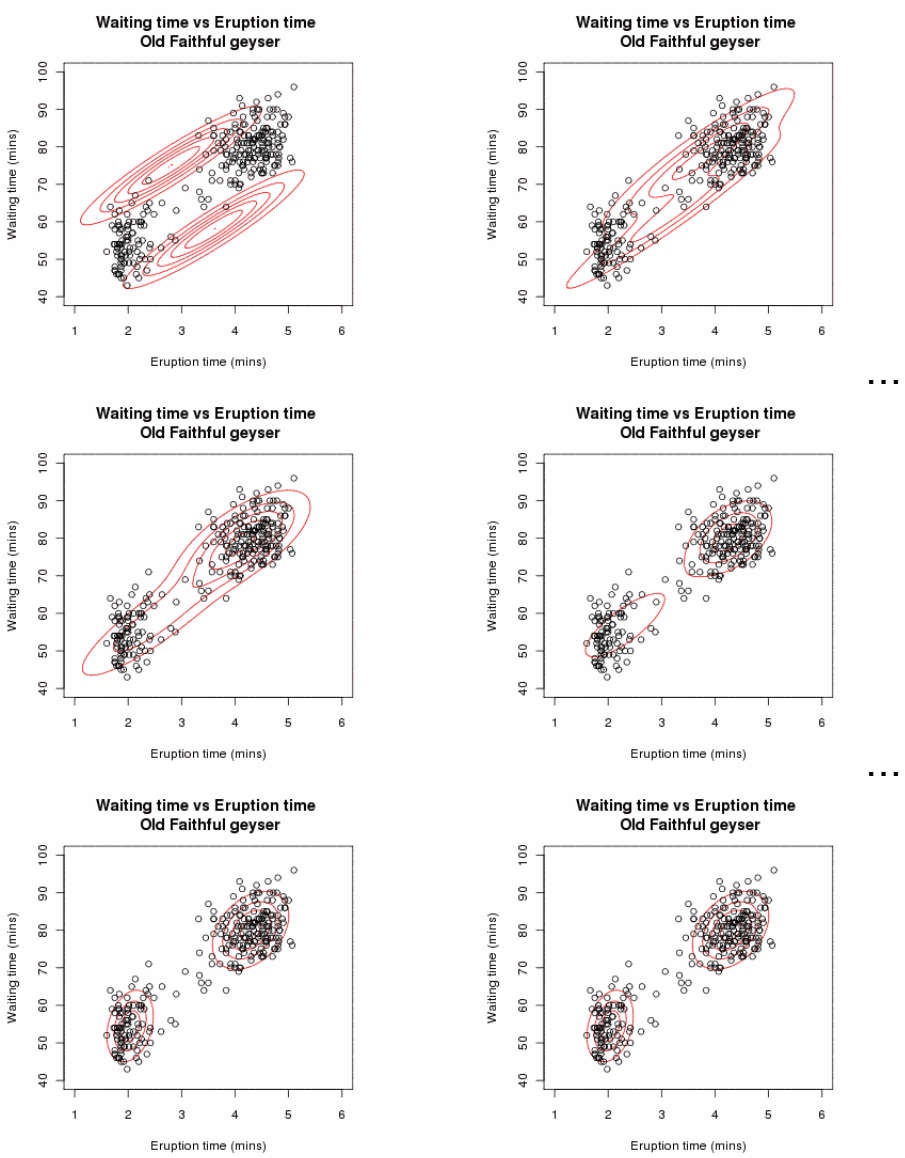
0. Initialize the cluster parameters (means, covariances and mixing coefficients)
1. Estimation step: Calculate the soft cluster assignments (the cluster responsibilities) using the current cluster parameter values:

$$r_{ik} = \frac{\pi_k N(x_i|\mu_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j N(x_i|\mu_j, \Sigma_j)}$$

2. Maximization Step: Estimate the cluster parameters using the current soft cluster assignments.

$$\hat{\mu}_k = \frac{1}{N^{soft}{}_k} \sum_{i=1}^{N} r_{ik} x_i$$

3. Compare the log Likelihood for the estimated cluster parameters to the previous cluster parameters. If there is no convergence, repeat steps 1, 2 and 3.



Frames taken from an animation of the EM algorithm to a 2 component Gaussian mixture model on the Old Faithful dataset. From Wikipedia

Remember that the resulting clusters are based on soft assignments; each observation exists to some extent between 0 and 1 in every cluster and the total of the soft assignments for a given observation across all clusters is 1. The cluster responsibility vectors that we get at convergence tell us to what extent each observation is associated with each cluster. We have the converged the cluster parameters (mean vector, covariance matrix and cluster proportion) for each cluster to tell us where the cluster is centered and how it is shaped.

Convergence, initialization, and overfitting of EM

The Expectation Maximization algorithm is a coordinate ascent algorithm that converges to a local mode.

- we can equate the E and M steps with alternating maximizations of an objective function.
- convergence is assessed using the log likelihood of the data under the current estimated cluster parameters and responsibilities

Initialization

Initialization is critical to both convergence and runtime and can be done in a number of ways:

- Choose K random observations to use as centroids and assign observations to the nearest centroid to form initial parameter estimates
- Pick centers sequentially to provide good coverage of data like in k-means++
- Run k-means and use the result as the initial clustering.
- Grow the mixture model by splitting (and sometimes removing) clusters until K clusters are formed. (this is popular in speech applications)

Overfitting

Overfitting in EM is a real problem in practice and the algorithm must be tweeked to avoid it. To visualize this, think about a system of two clusters, but one of the clusters is assigned a single point only. When we calculate the Expectation Maximum, what happens is that

- the mean exactly equals the observation
- the variance goes to zero, such that the shape of the clusters is infinitesimally small around the observation
- the cluster responsibility for this observation becomes 1 for this cluster and zero for all other clusters

The clue that the model is over-fit is that the variance has gone to zero; now no other observation can possibly fit in the cluster (since it is infinitesimally small) and the observation cannot possibly be assigned to another cluster, since the cluster holds all of the

responsibility for the observation (it's likelihood of being in that cluster is 1).

In high dimensional data, like the clustering of large text documents, this problem is much more likely because it is more likely that a document contains a unique word. If the document contains a unique word, then it will become a cluster of one and force the remaining observations into K-1 clusters.

Also, at least initially, some words may be entirely absent from a cluster, causing the M step to produce zero mean and variance for those words. This means any data point with one of those words will have 0 probability of being assigned to that cluster since the cluster allows for no variability (0 variance) around that count being 0 (0 mean). Since there is a small chance for those words to later appear in the cluster, we assign a small positive variance (~1e-10) rather than zero. Doing so also prevents numerical overflow.

The fix is simple; we can regularize the M-step's estimated covariance matrix each time it is calculated by adding a very small value along the diagonal to guarantee that it does not go to zero.

There is another approach where each of the estimated parameters is smoothed (not just the covariance) using a formal Bayesian approach involving pseudo-observations (place prior on parameters).

Convergence

To assess convergence, a likelihood function is used to describe the plausibility of the parameter values given the observed data. The likelihood of the set of parameter values given the observed values is the probability of the observed values given the set of parameter values.

We denote the probability density function (pdf) associated with the observed values (the outcomes) O as $f(O|\theta)$. θ is the set of estimated parameters (for our mixture of Gaussians model, this is $\{\pi_k, \mu_k, \Sigma_k\}$). Then the likelihood function for a set of estimated distribution parameters θ given a known observation O, is $L(\theta|O)$.

Simply put, the likelihood of the estimated distribution parameters given the known observation O is the probability of seeing the value O given the probability distribution:

$$L(\theta|O) = f(O|\theta)$$

We want to maximize this function. So to calculate the likelihood, we plug our known observations into our estimated model parameters and iterate on the parameters until we find a maximum. In practice, we use the natural logarithm of the likelihood function.

The log likelihood (the natural logarithm of the likelihood function, like the likelihood function, quantifies the probability of observing a given set of data under a particular set of the parameters in our model. Specifically, we will keep looping through EM update steps until the log likelihood ceases to increase at a certain rate.

The log likelihood for an estimated probability distribution θ, given an observed value, is the sum of the natural logarithm of the likelihood of each observed value given our estimated parameters:

$$F(\theta|x) = \sum_{i=1}^{n} lnf(x_i|\theta)$$

where

F(θ) log-likelihood of distribution θ, given the set of observations x
θ probability distribution parameters
x_i the i-th observation

So for our mixture of Gaussians;

$$F(\pi_k, \mu_k, \Sigma_k|x) = \sum_{i=1}^{N} lnf(x_i|\pi_k, \mu_k, \Sigma_k)$$

Scaling mixtures of Gaussians for document clustering

When using mixture models for clustering text documents, documents are represented as vectors of TF-IDF data. The length of the vector is the number of words in the vocabulary, which can be very large. So the vector space is high dimensional. We can model high-dimensional data, like the Wikipedia dataset, with a mixture of Gaussians, though with increasing dimension we run into two important issues associated with using a full covariance matrix for each component.

- Computational cost becomes prohibitive in high dimensions: score calculations have complexity cubic in the number of dimensions M if the Gaussian has a full covariance matrix (includes all symmetric correlation parameters).
- A model with many parameters requires more data: μ_k will be a vector of length M, the number of words in the vocabulary. The full covariance matrix, with symmetric correlation parameters, will be an MxM matrix. It's a symmetric matrix, so there are M(M+1) / 2, or on the order of M^2 unique parameters to fit. With the number of parameters growing roughly as the square of the dimension, it can quickly become impossible to find a sufficient amount of data to make good inferences.

So we need to simplify in such a high dimensional space. We can do this by assuming that there is no correlation between the variances in each dimension. That means that all of the elements, other than the variance terms on the diagonal, are zero:

$$\Sigma = \begin{bmatrix} {\sigma_1}^2 & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & {\sigma_k}^2 \end{bmatrix}$$

By requiring the covariance matrix of each component to be diagonal, each covariance matrix has only M parameters to fit and the score computation decomposes into M univariate score calculations.

The full covariance is given by:

$$\hat{\Sigma}_k = \frac{1}{N^{soft}{}_k} \sum_{\mathrm{i}=1}^{N} r_{ik}(x_i - \hat{\mu}_k)(x_i - \hat{\mu}_k)^T$$

This is a square matrix with M rows and M columns, and the above equation implies that the (v, w) element is computed by:

$$\hat{\Sigma}_{k,v,w} = \frac{1}{N^{soft}{}_k} \sum_{\mathrm{i}=1}^{N} r_{ik}(x_{iv} - \hat{\mu}_{kv})(x_{iw} - \hat{\mu}_{kw})$$

When assuming a diagonal matrix, we treat all correlation parameters as zero; we only need to calculate each of the M diagonal elements. **Each diagonal element (v, v) is given by**:

$$\hat{\sigma}^2_{k,v} = \hat{\Sigma}_{k,v,v} = \frac{1}{N^{soft}{}_k} \sum_{\mathrm{i}=1}^{N} r_{ik}(x_{iv} - \hat{\mu}_{kv})^2$$

So we will just be learning the variances and ignoring correlations. That then means that we will be learning clusters as axis aligned ellipses, since it is the correlation terms that 'rotate' the ellipse and they are all zero in the simplified model.

This is not a big limitation; since within each cluster we will learn weights for each dimension, which means each dimension in each cluster has it's own spread. This is still much more flexible than the k-means model, even when weights are applied to k-means, since those apply to all clusters and we had to specify the weights, rather than learn them. In addition, the mixture model provides us with soft assignments.