Clustering, Mixtures, and the EM Algorithm

Machine Learning: Module 1

Sean Norton; Simon Hoellerbauer

September 23, 2018

The Problem: Finding Groups

In social science, we often believe our observations have some sort of group structure.

- Regime types
- Types of voters
- Types of legislators

However, our data doesn't (generally) come with these groupings conveniently pre-labeled.

E.g. Regime Types

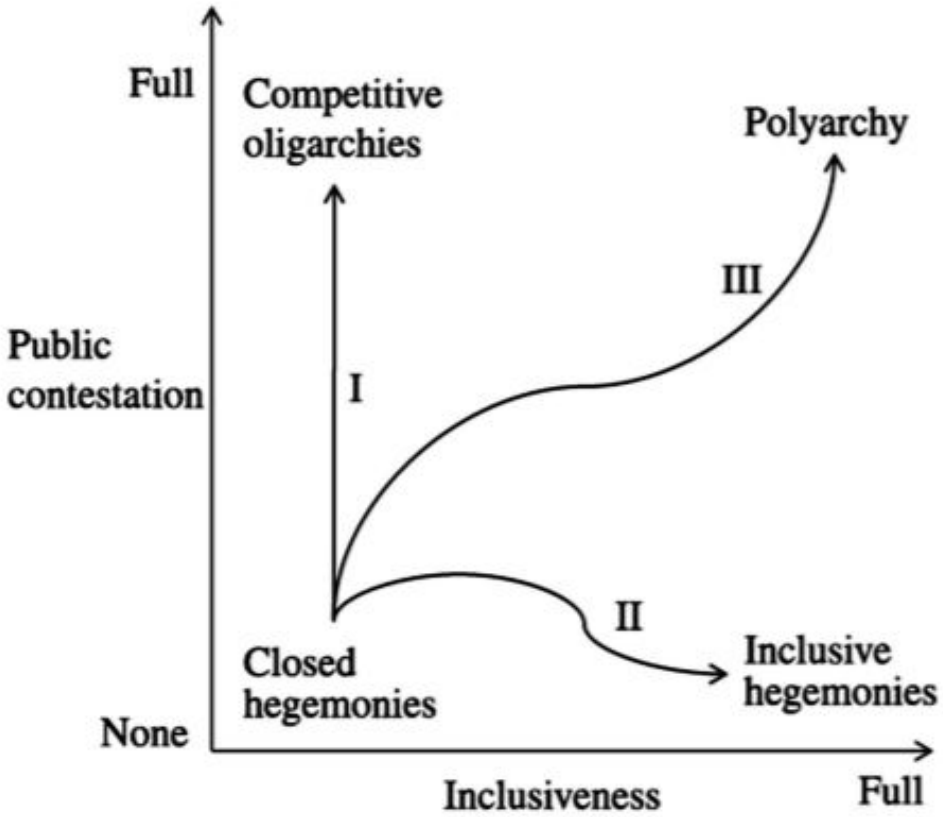


Figure: Dahl: Regime Types

The Problem Continued

The general approach to dealing with this problem has been to hand-label cases. This is problematic because:

- It's time-consuming
- Humans aren't able to consider all available data at once
- It relies on researcher discretion
- For large datasets (e.g. all countries in the world), this is impossible.

What if there was a way to find latent groupings between our cases quickly and with as little researcher discretion as possible?

Enter Cluster Analysis

This is exactly what cluster analysis is intended to do!
Given:

- Data
- Number of clusters
- Variables
- Similarity measure

A cluster analysis algorithm finds groupings, or clusters, that maximize the similarity between observations within a cluster.

K-Means and Notation

One of the most common similarity measures is the squared distance between the center (mean) of a cluster.
This is known as *k-means* or *k-nearest neighbor* clustering.
Before we dive into the math, some notation:

- k: total number of clusters
- r_{nk}: indicator vector of cluster membership for observation x_n
- μ_k: the centroid of cluster k

K-Means: The Math

Cluster analysis relies on a measure of similarity, which in k-means is:

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|x_n - \mu_k\|^2$$

This number J is also known as a *distortion measure*.
What (hopefully) familiar thing does this measure look like?

The Math cont.

Q: But how do we choose values of μ_k given that we don't actually know the cluster assignments?
A: We don't!
We can find k_m through a version of the *expectation-maximization algorithm*:

1. Initialize some random values for μ_k
2. Minimize J w.r.t. r_{nk}; i.e. assign cluster memberships in order to minimize distortion
3. Using the previous r_{nk}, minimize J w.r.t to μ_k; i.e., assign new means that minimize distortion
4. Repeat until convergence (J does not change, or the change falls below some threshold)

The Math: Visualized

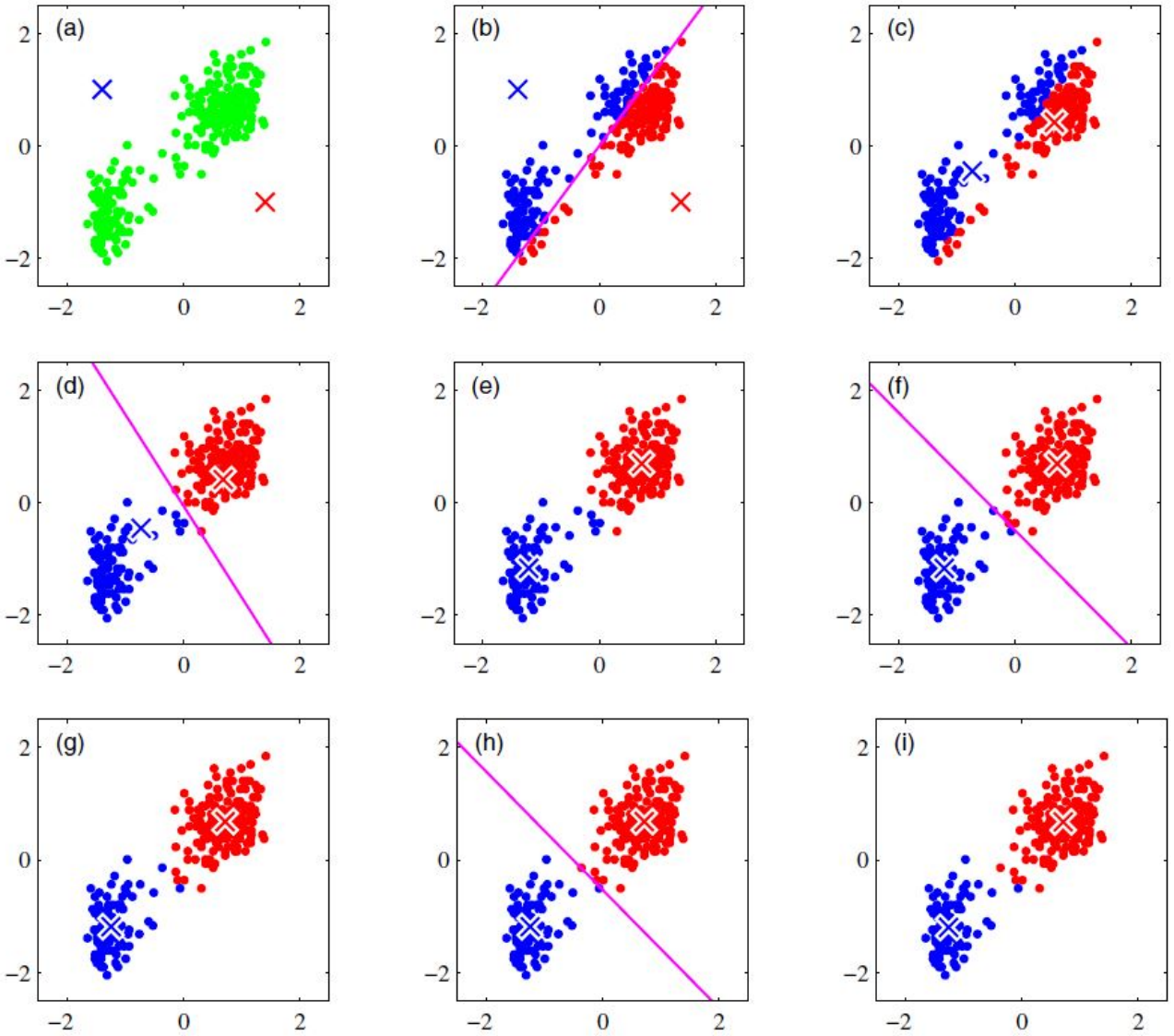


Figure: Optimizing Clusters

Problems with Clustering

- Choosing k: this requires trying a lot of different k w.r.t. some similarity measure
  - Plot the WCSS against the number of clusters, and look for a "bend" in the plot; this is known as the elbow method
  - Use average silhouette width: the silhouette is a measure how similar an observation is to its own cluster (consistency) and how dissimilar it is to other clusters (separation)
  - Gap statistics: compare multiple values of k to a simulated reference distribution of datasets with clusters varying from $k = 1$ to $k = max$
- Sensitivity to outliers; luckily, there are clustering methods other than k-means
  - Partitioning around medoids (PAM): uses median instead of mean
  - Hierarchical clustering: creates a tree-based representation of the data without specifying k; clusters are created by "cutting" the tree.
- Fundamentally descriptive; clusters will not necessarily be the same given different data

Problems cont.

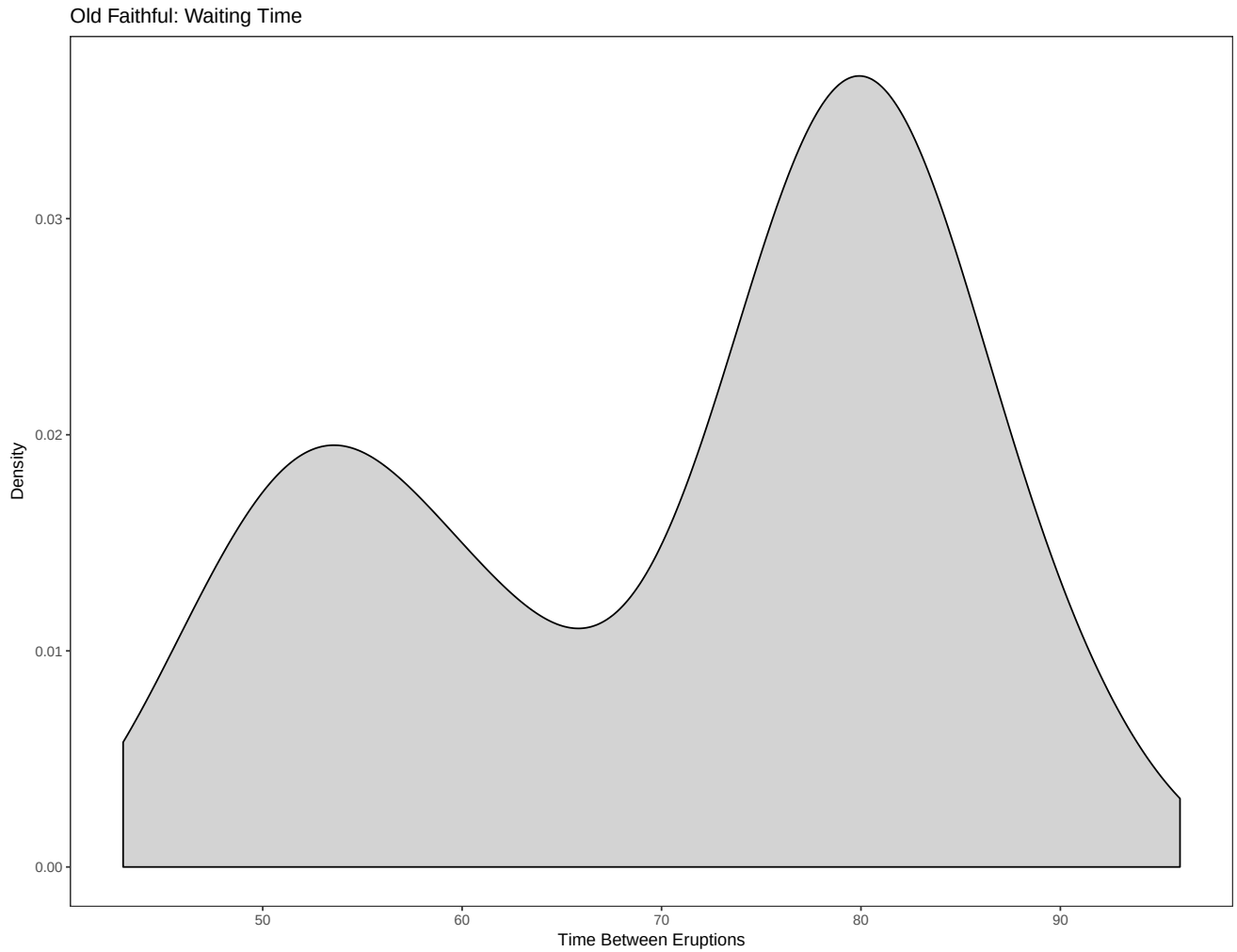
Figure: My Cluster Plot

Mixture Models: Motivation

What if instead of "hard" cluster assignments, we could assign a probability that cases belong to a particular group?
That is precisely what *mixture models* are intended to do!
Mixture models allow observations to belong to different distributions - either different parameterizations of the same distribution, or different distributions entirely.
This allows us to:

- Sort cases into groups, much like cluster models, but do so in a way that quantifies uncertainty
- Estimate a model in the same step as the grouping, which most out-of-the-box clustering packages do not do
- Work with data that has a multimodal distribution without having to discard a substantial amount of variation.

Motivating Example: Old Faithful



Motivating Example: Old Faithful

Clearly, this data appears to be normally distributed, but not in a single normal distribution.
Instead of fitting a single Gaussian, we can fit a *mixture* of Gaussians of the following form:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

Where π_k is the probability that an observation belongs to cluster k.
This is a Gaussian mixture model (GMM)

Deriving the Model

To be able to estimate π_k, let us define a K-dimensional vector $\mathbf{z}$, which is 1 if an observation is in group k and 0 otherwise. $\mathbf{z}$ is a latent variable - deriving the model this way will be of use later.

$$p(z_k = 1) = \pi_k$$

Or alternatively, since this is a 1-of-K vector:

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}$$

What must be true of π

Deriving the Model

The conditional distribution $(\mathbf{x}|\mathbf{z})$ is then:

$$p(x|z) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\mu_k, \Sigma_k)^{z_k}$$

Marginalizing the distribution over $\mathbf{z}$ returns the original formulation:

$$p(x) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x|\mu_k, \Sigma_k)$$

While this seems like a useless exercise, we can now work with the joint distribution of our observations and the latent variable, $p(x|z)$, which will make estimation considerably easier.

More for Later

Another important quantity for later will be $\gamma(z_k)$.
For now, think of π_k as the *prior* probability of group membership. We are also interested in the posterior probability ($p(z_k = 1|x)$), which can be calculated using Bayes' Rule:

$$\frac{p(z_k = 1)p(x|z_k = 1)}{\sum_{j=1}^{k} p(z_j = 1)p(x|z_j = 1)}$$

Do this terms look familiar? How would you put this in terms of the joint PDF we just derived?

Attempting MLE

The log-likelihood of the GMM is given by:

$$ln\ p(x|\pi, \mu, \Sigma) = \sum_{n=1}^{N} ln\{\sum_{k=1}^{k}(x_n|\mu_k, \Sigma_k)\}$$

What do you notice is strange about this log-likelihood compared to ones we've seen before?

Attempting MLE

In the log likelihood of the GMM, the logarithm does not act directly on the Gaussian distribution. When we set the derivatives of likelihood to 0 in order to maximize the parameters, this results in there not being a closed form solution.

$$0 = -\sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k (x_n - \mu_k)$$

Where the mean of group k is the mean of each observation in the group, weighted by the posterior probability of its membership in that group:

$$\mu_k = \frac{1}{N_k} \sum_{n=1} N \gamma(z_{nk}) x_n$$

$$N_k = \sum_{n=1} \gamma z_{nk} = \#\text{ of points in cluster k}$$

Attempting MLE

Similarly, the covariance is given by:

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1} N\gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^\mathsf{T}$$

And the mixing coefficients are (intuitively):

$$\pi_k = \frac{N_k}{N}$$

Problems with MLE

- As can be seen from the previous partial derivatives, the mean and variance of the GMM estimates depend on the posterior probabilities, which themselves depend on the mean and variance. Therefore, there is not a closed form solution.
- Because we now have multiple components, a situation can arise where a component collapses on a single point when an observation is equal to the component mean, contributing infinitely to the log likelihood:

$$\mathcal{N}(x_n|\mu_k, \Sigma_j) = \frac{1}{2\pi^{\frac{1}{2}}} \frac{1}{\sigma_j}$$

$$\lim_{\sigma_j \to \infty} = \infty$$

Problems with MLE

- MLE also suffers from an *identifiability* issue: in a K component mixture, there are $K!$ possible solutions for assigning K parameters to K components; for any chosen parameter in the π_k parameter space there are $k-1!$ points that produce the exact same distribution
- This creates a multimodal distribution of the log likelihood in which there is no unique solution; more generally for all latent variable models, this is known as *label nonidentifiability*

Non-Gaussian Mixtures

Mixture models can be generalized to fit any distribution:

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|\theta)$$

Or even to mix *different* distributions:

$$p(x) = \sum_{k=1}^{K} \pi_k p(x|z_i = k, \theta)$$

Or even mixtures of non-parametric distributions - the mixing coefficient places no restrictions on the form of the distribution after it.

Mixtures of Bernoullis

Another common mixture is a mixture of Bernoulli distributions, also called *latent class analysis*.
As a reminder, the Bernoulli distribution is:

$$p(x|\mu) = \prod_{n=1}^{N} \mu_i^{x_i}(1-\mu_i)^{x_i}$$

And a mixture of Bernoullis is:

$$p(x|\mu, \pi = \sum_{k=1}^{K} p(x|\mu_k)$$

Mixtures of Bernoullis

This yields the following expressions of the mean and variance:

$$\mathbb{E}(x) = \sum_{k=1}^{K} \pi_k \mu_k$$

$$cov[x] = \sum_{k=1}^{K} \pi_k (\Sigma_k + \mu_k \mu_k^\intercal) - \mathbb{E}(x)\mathbb{E}(x)^\intercal$$

Where Σ_k is the variance of a single Bernoulli component.
What do you notice about the variance? How is this an improvement on a single Bernoulli?

Mixtures of Bernoullis

The log likelihood is then:

$$lnp(\mathbf{X}|\mu,\pi) = \sum_{n=1}^{N} ln(\sum_{k=1}^{K} \pi_k p(x_n|\mu_k))$$

Does this produce a closed form solution?
Like GMM, does this have the potential to produce a singularity? Why or why not?

EM with LCA

To use the EM algorithm on a mixture of Bernoulli's, we again introduce z as latent mixture assignment. The conditional distribution $p(x|z,\mu)$ is then:

$$p(x|z,\mu) = \sum_{k=1}^{K} p(x|\mu_k)^{z_k}$$

And the prior for z:

$$p(z|\mu) = \sum_{k=1}^{K} \pi_k^{z_k}$$

Just like with the GMM, if we take the product of the conditional and the prior then marginalize out z, we get the original formulation back.

EM with LCA

This then yields the complete-data log likelihood:

$$ln p(\mathbf{X}, \mathbf{Z}|\mu, \pi) = \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk} \{ln \pi_k + \sum_{i=1}^{D} [x_{ni} ln \mu_{ki} + (1 - x_{ni}) ln(1 - \mu_{ki})]\}$$

How do we derive the E step from the complete data likelihood?

EM with LCA

The E step evaluates the expectation for the latent variables w.r.t to the posterior distribution of $\mathbf{Z}$:

$$\mathbb{E}_{\mathbf{Z}}(.) = \gamma(z_{nk})\{ln\pi_k + \sum_{i=1}^{D}[x_{ni} ln\mu_{ki} + (1 - x_{ni})ln(1 - \mu_{ki})]\}$$

Where, just like with the GMM, the posterior is calculated using Bayes' rule:

$$\gamma(z_{nk}) = \frac{\pi_k p(x_n|\mu_k)}{\sum_{j=1}^{K} \pi_j p(x_n|\mu_j)}$$

What does the M step do after this?

Putting It All Together

This suggests a generalized way to put together any mixture model for estimation with EM:

1. Create a PDF, $p(x)$, that includes a mixing coefficient π
2. Re-conceptualize that PDF in terms of a latent variable, z
3. Find the complete data log-likelihood
4. Take the expectation of step 3 w.r.t. to Z
5. Calculate the expression of the posterior of Z using Baye's rule.
6. Run the EM algorithm, and hopefully get interesting results!

Extra credit question: if you write your own EM algorithm based on the above, how can you check that it's working properly?