Gaussian Mixture Models

COS 424/524, SML 302: Fundamentals of Machine Learning

Professor Engelhardt

COS 424/524 SML 302

Lecture 12

Probabilistic models for clustering

In the last lecture, we learned about k-means analysis for clustering unlabeled data.

There were a number of drawbacks; most substantially, that there was not an associated probabilistic model.

Today we will discuss

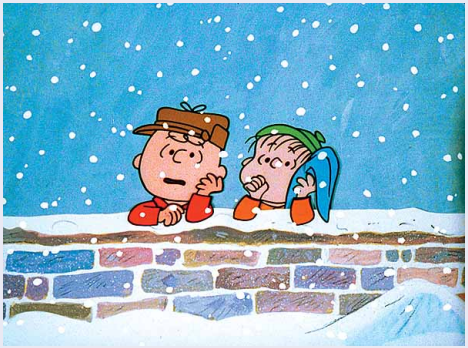
- a probabilistic model for clustering (a Gaussian mixture model);
- an associated parameter estimation method (expectation-maximization)

Before we do that, an example of how clustering can be used for dimension reduction.

Example: Charlie Brown and image compression

Image compression with vector quantization

- Each pixel is associated with a red, green, and blue value (RGB)
- A 1024×1024 image includes $1,048,576$ values $\langle x_1, x_2, x_3 \rangle$, requiring 3M of storage



How can we use clustering to compress this image?

Example: Charlie Brown and image compression

Image compression with vector quantization

- Let's cluster the RGB values $\langle x_1, x_2, x_3 \rangle$ into $K = 2$ clusters.

- Replace each pixel with cluster assignment (i.e., "paint by numbers").

Example: Charlie Brown and image compression

Image compression with vector quantization

- Each cluster location is a point in RGB. The set is the *codebook*.
- Here: $K = 2$. We need 1 bit per pixel + 2×3 bits $\approx$ 131KB.

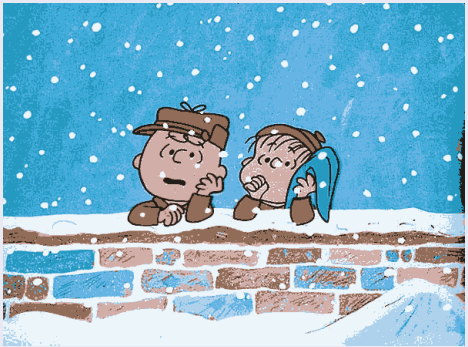
Example: Charlie Brown and image compression

Image compression with vector quantization

Here: $K = 4$. We need 2 bits per pixel $+ 4 \times 3$ bits $\approx$ 262KB.

Example: Charlie Brown and image compression

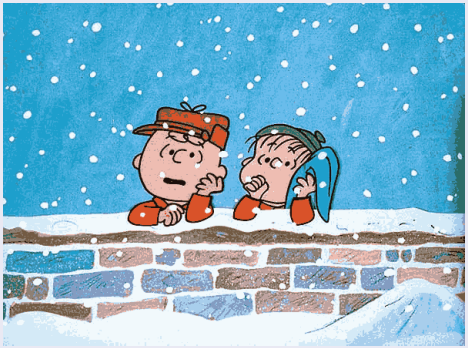
Image compression with vector quantization



Here: $K = 8$. We need 3 bits per pixel $+ 8 \times 3$ bits ≈ 393KB.

Example: Charlie Brown and image compression

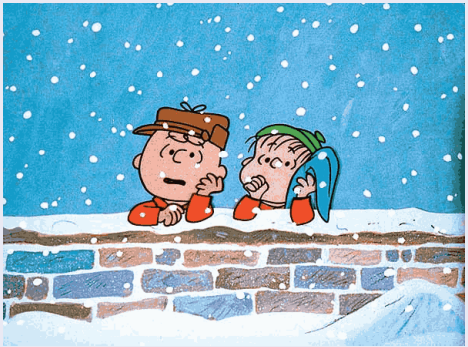
Image compression with vector quantization



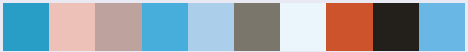
Here: $K = 16$. We need 4 bits per pixel $+ 16 \times 3$ bits $\approx$ 524KB.

Example: Charlie Brown and image compression

Image compression with vector quantization

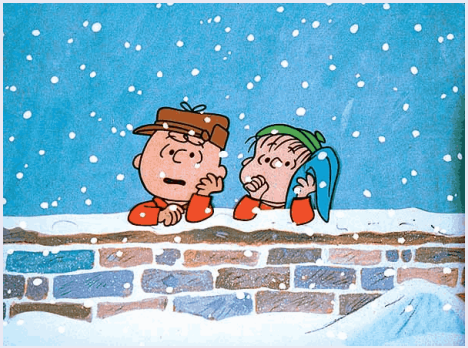


Here: $K = 32$. We need 5 bits per pixel $+ 32 \times 3$ bits ≈ 655KB.



Example: Charlie Brown and image compression

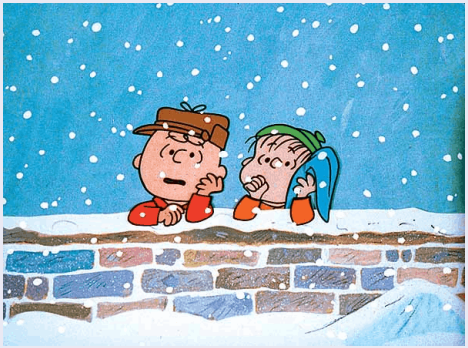
Image compression with vector quantization



Here: $K = 64$. We need 6 bits per pixel $+ 64 \times 3$ bits ≈ 786KB.

Example: Charlie Brown and image compression

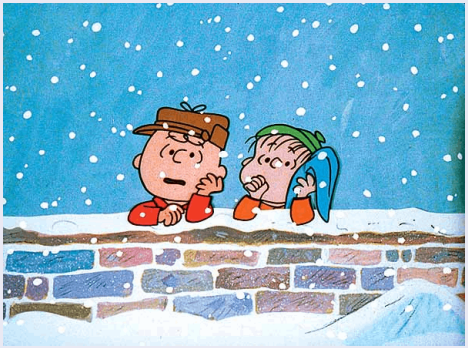
Image compression with vector quantization



Here: $K = 128$. We need 7 bits per pixel + 128×3 bits ≈ 0.92MB.

Example: Charlie Brown and image compression

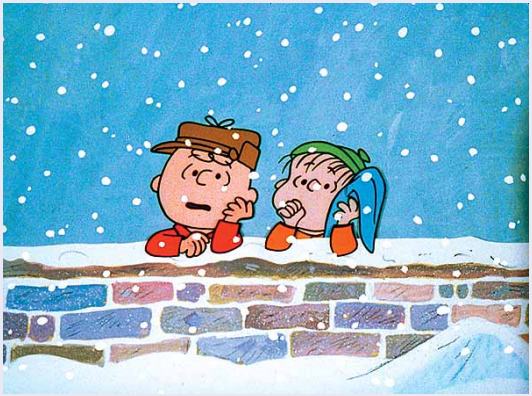
Image compression with vector quantization

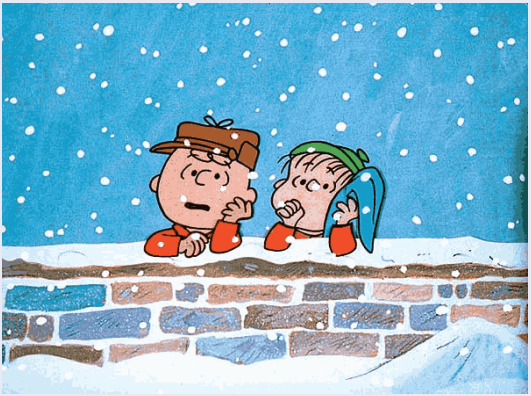


Here: $K = 256$. We need 8 bits per pixel + 256×3 bits $\approx$ 1048KB, or 1MB

Example: Charlie Brown and image compression

Image compression with vector quantization





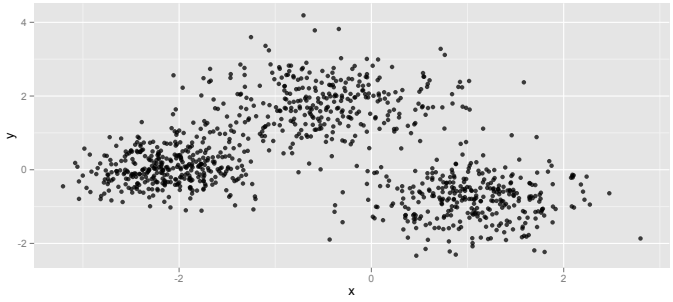
With $K = 32$, minimal changes in picture quality, but one sixth of the size.

Idea behind mixture models

- With mixture models, we can take any generative model and use it to cluster samples.
- We imagine each sample being generated by one of a collection of models (each with different parameter settings).
- To cluster, we simultaneously learn the parameter values and the clustering assignments.
- In K-means, there is a *distance measure* underlying cluster assignments.
- In mixture models, that measure is defined via generative distributions.

Mixture model framework

- Data are n samples $\mathcal{D} = \{\mathbf{x_1}, \ldots, \mathbf{x_n}\}$, where each x_i is a p-vector
- Each sample has a *latent cluster assignment* z_i, a K-indicator vector
- Each mixture component k has *parameters* θ_k, which is $O(p)$
- Each component k has *probability* $\pi_k \geq 0$, where $\sum_{k=1}^{K} \pi_k = 1$



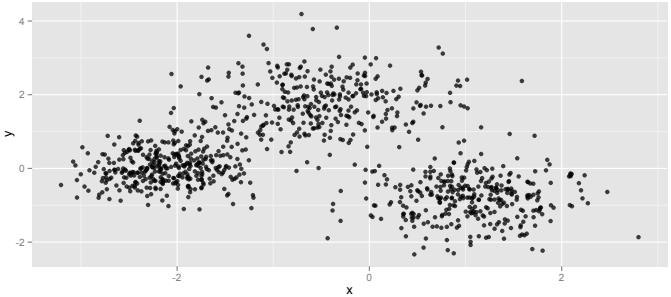
Mixture model distribution

- The joint distribution for sample i is

$$p(x_i, z_i \mid \pi, \theta_{1:K}) = p(z_i \mid \pi) p(x_i \mid z_i, \theta_{1:K})$$

where

$$p(x_i \mid z_i, \theta_{1:K}) = \prod_{k=1}^{K} p(x_i \mid \theta_k)^{z_i^{(k)}}$$

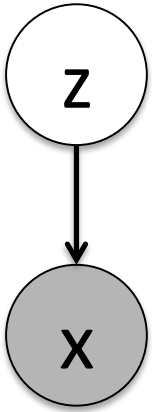


Generative model for a mixture model

Mixture model generative process

Generative process for each sample

1. Choose $z_i \sim \text{Mult}(\pi)$
2. Choose $x_i \mid z_i \sim p(x_i \mid \theta_{z_i})$



Recall that this generative graphical model factorizes the joint probability of x_i, z_i as:

$$\begin{aligned} p(x_i, z_i \mid \theta_{1:K}) &= p(z_i \mid \pi) p(x_i \mid z_i, \theta) \\ &= p(z_i \mid \pi) \prod_{k=1}^{K} p(x_i \mid \theta_k)^{z_i^k} \end{aligned}$$

Generative processes

The generative process is a useful way to think about models and inference.

Inference (or model fitting) can be thought of as "reversing" the process, finding the hidden variables and parameters that are the best candidates to generate the observed data.

Also, generative processes give us a framework to generate data.

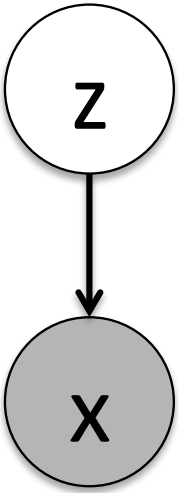
Exercise: Draw from a 2D Gaussian mixture model in R.

Parameters and latent variables

- This generative model is *identical* to generative model for classification using Naive Bayes:

$$p(x_i, z_i \mid \theta_{1:K}) = p(z_i \mid \pi) \prod_{k=1}^{K} p(x_i \mid \theta_k)^{z_i^k}$$

- Component-specific distributions are parameterized by $\theta_{1:K}$
- The vector π is the component proportions
- The critical difference is that we have not observed the class labels z_i in the training data



Examples of mixture models

$$p(x_i, z_i \mid \theta_{1:K}) = p(z_i \mid \pi) \prod_{k=1}^{K} p(x_i \mid \theta_k)^{z_i^k} = \sum_{k=1}^{K} z_i^k \pi_k p(x_i \mid \theta_k)$$

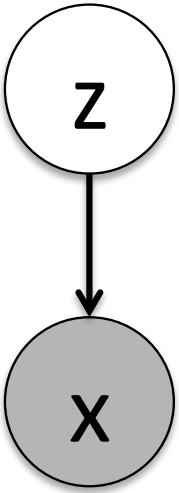
Possible types of mixture models

- Multinomial mixtures: mixture components are multinomial
- Gaussian mixtures: mixture components are Gaussian
- Spike-and-slab prior: mixture components are spikes (noise) or slabs (signal) prior distributions:
$$\beta \sim \pi\delta(0) + (1 - \pi)\mathcal{N}(0, \sigma^2).$$
- Hidden Markov models (HMMs): component labels change with respect to time.

Building mixture models from simple distributions

Just as we can take any generative model and make it into a model for classification, we can take any model and make it into a mixture model.

In a mixture model, the "distance measure" is defined by the likelihood of the sample with respect to a parametric distribution, $p(x_i \mid \theta_k)$.



Gaussian mixture model

Let's start our study of mixture models with the canonical *Gaussian mixture model*. For simplicity, let's consider GMMs with unit variance; denote the cluster means as $\mu_{1:K}$.

The likelihood for a 1D Gaussian mixture (with unit variance):

$$\begin{aligned} p(x \mid \pi, \mu_{1:K}) &= \sum_{z=1}^{K} p(x, z \mid \pi, \mu_{1:K}) \\ &= \sum_{z=1}^{K} p(z \mid \pi) p(x \mid \mu_z) \\ &= \sum_{z=1}^{K} \pi_z \mathcal{N}(x; \mu_z, 1) \end{aligned}$$

Note here that we marginalize over the latent z variables.

Gaussian mixture model (GMM)

The density of a 1D Gaussian mixture (with unit variance)

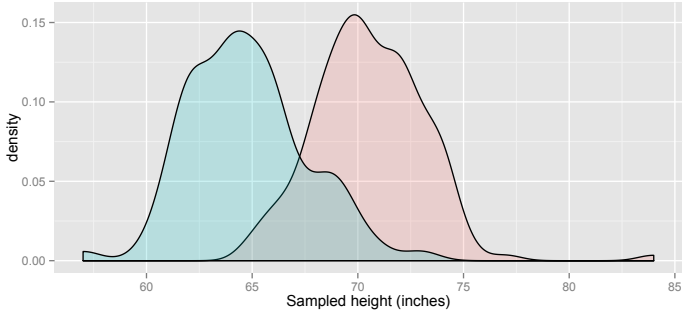
$$p(x \mid \pi, \mu_{1:K}) = \sum_{z=1}^{K} \pi_z \mathcal{N}(x; \mu_z, 1)$$

Here, $\mathcal{N}(x; \mu_z, 1)$ is the Gaussian density with mean μ_z and unit variance.

These are K Gaussian "bumps", each weighted by their corresponding proportion π_k.

You can think about latent variable models in three ways.

- One is as a way to estimate a complex, multimodal distribution for the marginal distribution of x.
- Another is as a way to uncover latent structure in the data.
- Still another is as dimension reduction.



The difficulty with the likelihood

We fit a mixture model with maximum likelihood.

Let's write down the log likelihood

$$\begin{aligned}
\log p(x_{1:n} \mid \pi, \mu_{1:K}) &= \log \prod_{i=1}^{n} p(x_i \mid \pi, \mu_{1:K}) \\
&= \sum_{i=1}^{n} \log p(x_i \mid \pi, \mu_{1:K}) \\
&= \sum_{i=1}^{n} \log \sum_{z_i=1}^{K} p(z_i \mid \pi) p(x_i \mid z_i, \mu_{1:K})
\end{aligned}$$

In line 3, the log cannot be pushed through the summation. We are stuck.

The difficulty with the likelihood

- The summation is in the likelihood because z_i is a *hidden variable*, and we are marginalizing over it.
- Suppose z_i were observed. Then log likelihood decomposes into sums over simple functions. This is what let us fit classification models.
- Here, log likelihood does not decompose: taking derivative wrt log summation, all parameters depend on value of all others
- Fitting latent variable models with max likelihood methods is hard: often no closed form expression for MLE, computationally difficult, possibly non-convex.

How can we estimate the parameters of this latent variable model?

EM for the Gaussian mixture model

- The Expectation-Maximization (EM) algorithm is general-purpose technique for fitting parameters in models with latent variables.
- Note that there are exceptions both ways:
  - some latent variable models do not require EM to fit parameters
  - for some latent variable models, EM is does not work
- There are obvious parallels between K-means and EM, both of which have a coordinate ascent interpretation.

EM algorithm in pseudocode

Expectation-Maximization algorithm pseudocode

Initialize component parameters

Repeat until convergence:

1. **Soft cluster assignment:** For each sample i, compute conditional distribution of cluster assignment z_i given current setting of parameters $\pi^{(t)}$ and $\mu_{1:K}^{(t)}$.
2. **Pretend assignments are observations and fit model with MLE**: Obtain $\pi^{(t+1)}$ and $\mu_{1:K}^{(t+1)}$ by computing approximate MLEs for each cluster, weighting each sample's contribution by its probability of being in that cluster.

EM for Gaussian mixture model

EM for GMM

Initialize component parameters.

Repeat until convergence:

1. **E-step**: For each sample i, compute $\lambda_i = p(z_i \mid x_i, \pi^{(t)}, \mu_{1:K}^{(t)})$
2. **M-step**: For each cluster k's distribution

$$\mu_k^{(t+1)} = \frac{\sum_{i=1}^n \lambda_{i,k} x_i}{\sum_{i=1}^n \lambda_{i,k}}$$

and for the cluster proportions

$$\pi_k^{(t+1)} = \frac{\sum_{i=1}^n \lambda_{i,k}}{n}$$

How does each step compare with K-means?

E-step, in more detail

The E-step involves assigning a point probabilistically to a cluster.

$$p(z_i = k \mid x_i, \pi^{(t)}, \mu_{1:K}^{(t)}) \propto p(z_i \mid \pi^{(t)}) p(x_i \mid \mu_k^{(t)})$$

This is the same computation for classifying new observations (e.g., "What class is this image?") in the naive Bayes classifier.

The parameters are fixed at their most recent values from the M-step.

For a multinomial vector z, this looks like a point on the K simplex.

M-step, in more detail

How about the M-step for component-specific means? μ_k?

- MLE is a weighted empirical average of all samples, where each sample is weighted by its posterior probability of being in cluster k.

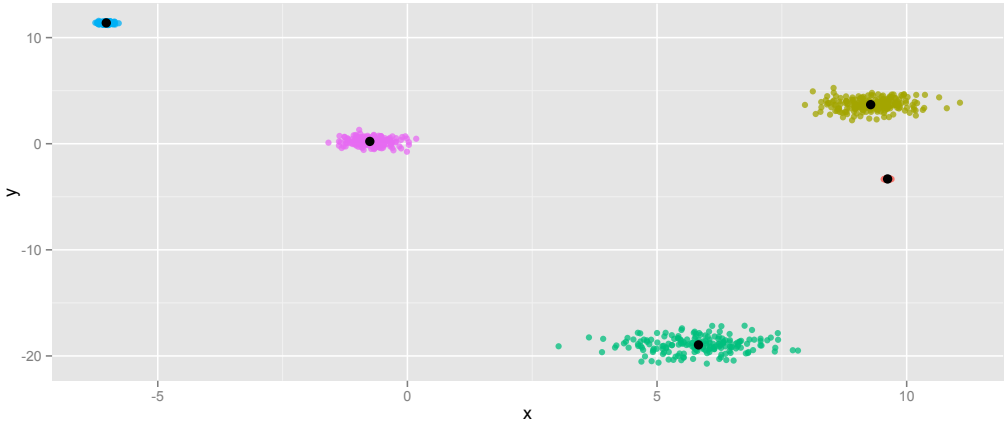
How about M-step for component proportions π?

- The numerator is expected number of samples in cluster k.
- The denominator is the total number of samples n.

All these M-steps are like traditional MLEs, but with *expected assignment* instead of observed class labels.

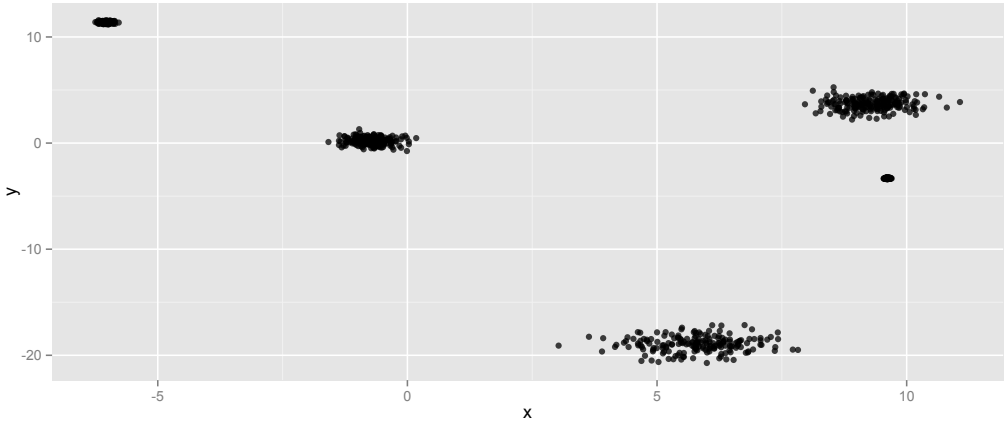
Example of EM for GMM

Let's consider 1000 data points from five unequal clusters.



Example of EM for GMM

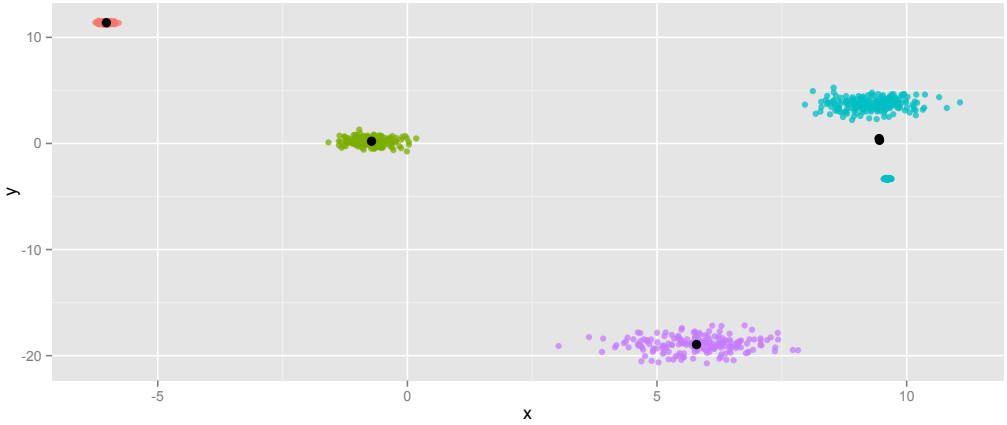
Here is what the (unlabeled) data look like:



Let's try to fit a GMM using EM with $K = 5$.

Example of EM for GMM

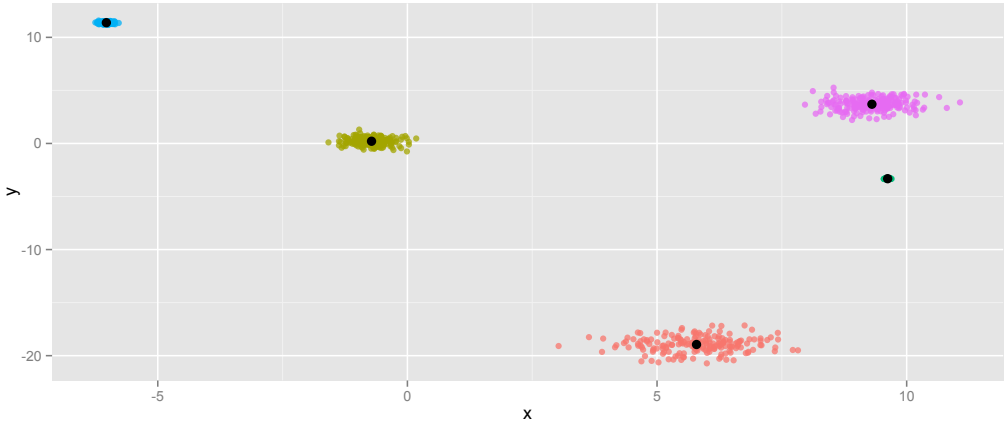
Our first try:



Let's try again...

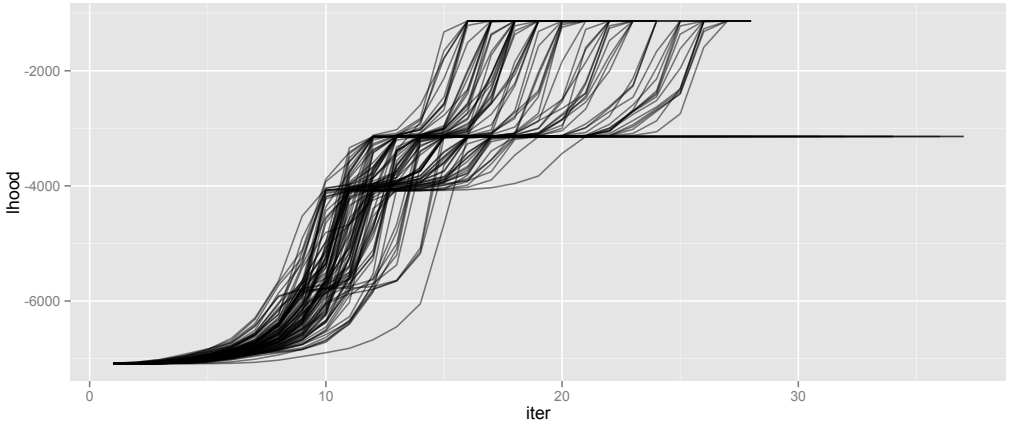
Example of EM for GMM

Our next try:



Let's try 100 times and plot the log likelihood over these runs...

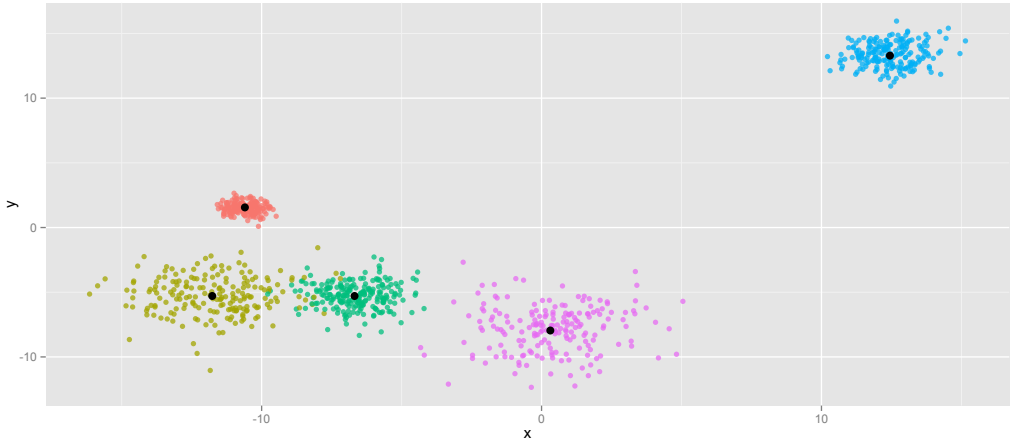
Example of EM for GMM



Notice the convergence to local optima *even for an easy problem*

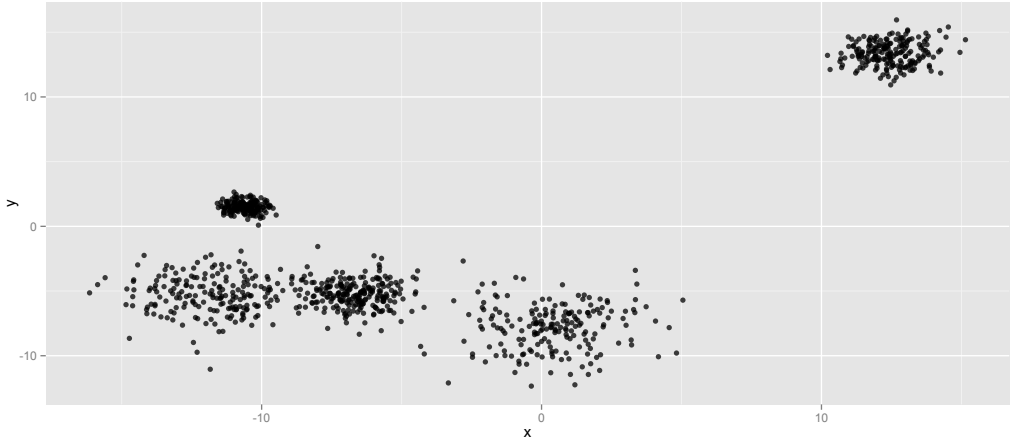
Example of EM for GMM

Let's consider a second example of 1000 data points from five unequal clusters.



Example of EM for GMM

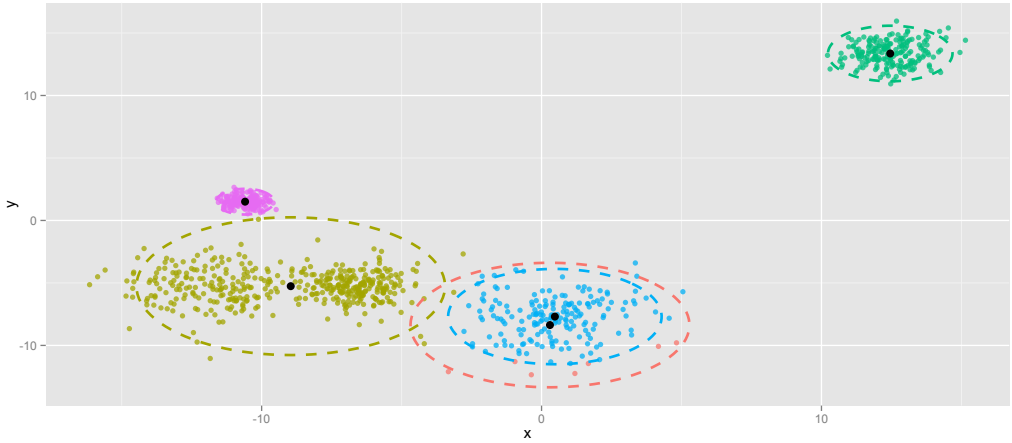
Here is what the (unlabeled) data look like:



Let's try to fit a GMM using EM with $K = 5$.

Example of EM for GMM

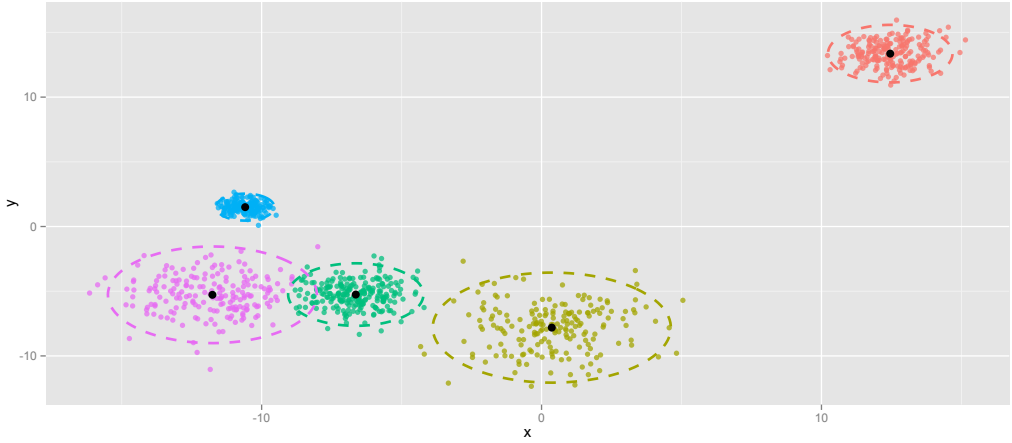
Our first try:



Let's try again...

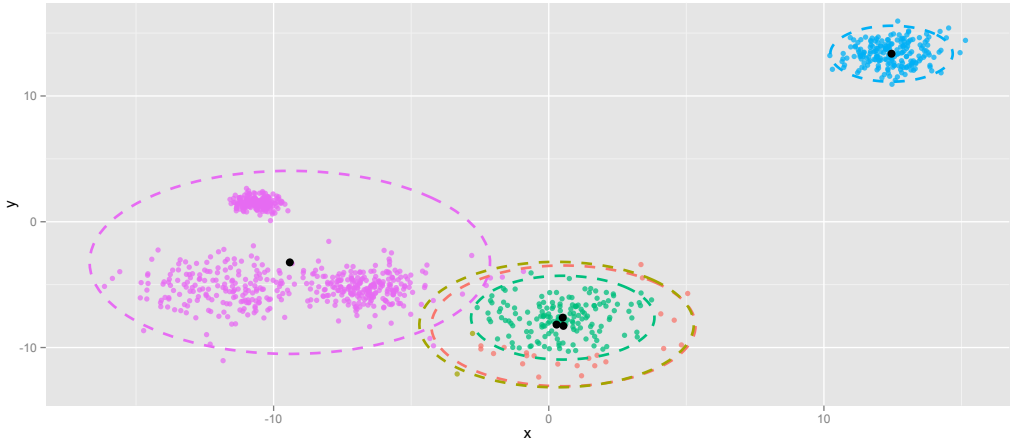
Example of EM for GMM

Our next try:



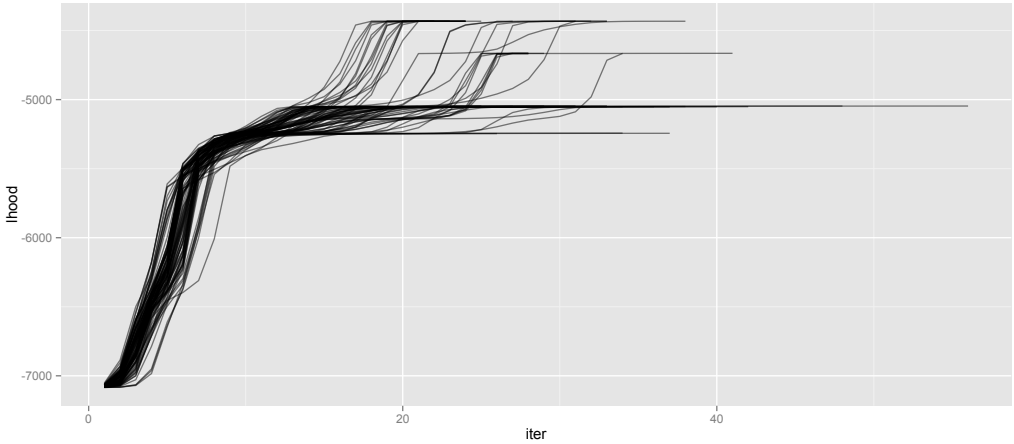
Example of EM for GMM

Our worst try:



Let's try 100 times and plot the log likelihood over these runs...

Example of EM for GMM



Notice the convergence to local optima...

Example of EM for GMM: iterations of EM

http://www.cs.princeton.edu/∼bee/demos/gmm5.ani.html

Bernoulli mixture model

We can also consider the *Bernoulli mixture model*. Denote the cluster biases as as $\mu_{1:K}$.

The likelihood for a Bernoulli mixture:

$$\begin{aligned} p(x \mid \pi, \mu_{1:K}) &= \sum_{z=1}^{K} p(x, z \mid \pi, \mu_{1:K}) \\ &= \sum_{z=1}^{K} p(z \mid \pi) Bern(x \mid \mu_z) \\ &= \sum_{z=1}^{K} \pi_z Bern(x; \mu_z) \end{aligned}$$

Here again we marginalize over the latent z variables.

Bernoulli mixture model

The density of a 1D Bernoulli mixture

$$p(x \mid \pi, \mu_{1:K}) = \sum_{z=1}^{K} \pi_z Bern(x; \mu_z)$$

Here, $Bern(x; \mu_z)$ is the Bernoulli density with cluster-specific bias μ_z.

Interpretation: randomly select one of K coins, each with their own bias, flip it once, and record the value. Which coin did you choose?

Bernoulli mixture model

The density of a 1D Bernoulli mixture

$$p(x \mid \pi, \mu_{1:K}) = \sum_{z=1}^{K} \pi_z Bern(x; \mu_z)$$

Here, $Bern(x; \mu_z, 1)$ is the Bernoulli density with cluster-specific bias μ_z.

While this problem is not particularly interesting, we may consider a binomial or multinomial distribution (i.e., multiple coin flips).

Then the problem becomes clustering n samples of p coin flips with m heads and $p - m$ tails.

General mixture models

- We can use this clustering framework for many kinds of data.
- One course theme: statistical models allow us to analyze disparate data types jointly.
- Clustering has a latent categorical variables (cluster labels) and features with arbitrary distributions.
- We've seen Gaussian, Bernoulli, binomial, and, in a previous class, spike-and-slab.
- Again, consider this framework for any simple generative model, or independent draws from simple generative models.
- How do you handle missing data in this framework? Ignore it.

Key point for mixture models

- The generative distribution tells us that each sample x_i is generated from *exactly one* mixture component.
- When we perform *soft assignment* we will find that we converge to a local minima for which every sample has a probabilistic assignment to each of the K mixture components.
- Do not confuse uncertainty in cluster assignment with the assumption that a sample is generated from more than one mixture component.

Mixture models versus K-means

	mixture model	K-means
assignment	soft (probabilistic)	hard
cluster proportions	modeled π	not modeled
likelihood	available	not available
domain-specific info	generative model	distance function

Mixture models: summary and assumptions

- A mixture model is a generative model of an observation where the observation may come from one of K possible clusters, each with their own generative distribution
- A mixture model assumes:
  - each sample is generated from a single mixture component
  - whatever assumptions are made by the generative models specific to the cluster-specific generative distribution
  - the cluster labels have not been observed
- We can use Expectation-Maximization to fit the latent variables and the parameters in this class of model.

Additional Resources

- MLAPA: Chapter 11
- *Pattern Recognition and Machine Learning*, Chapter 9
- Dempster et al. (1977): "Maximum likelihood from incomplete data via the EM algorithm"
- Metacademy: *Expectation-Maximization algorithm*
- (video) Mathematical Monk: *Expectation-Maximization (EM)*