CSC321 Lecture 18: Mixture Modeling

Roger Grosse

Overview

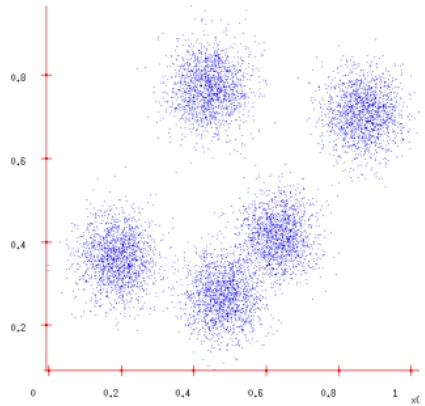
- Some examples of situations where you'd use unupservised learning
  - You want to understand how a scientific field has changed over time. You want to take a large database of papers and model how the distribution of topics changes from year to year. But what are the topics?
  - You're a biologist studying animal behavior, so you want to infer a high-level description of their behavior from video. You don't know the set of behaviors ahead of time.
  - You want to reduce your energy consumption, so you take a time series of your energy consumption over time, and try to break it down into separate components (refrigerator, washing machine, etc.).
- Common theme: you have some data, and you want to infer the causal structure underlying the data.
- This structure is latent, which means it's never observed.

Overview

- In last lecture, we looked at density modeling where all the random variables were fully observed.
- The more interesting case is when some of the variables are latent, or never observed. These are called latent variable models.
  - Today's lecture: mixture models, where the latent variable comes from a small discrete set
  - Next week: latent variable models which have distributed representations — these are much more powerful

Clustering

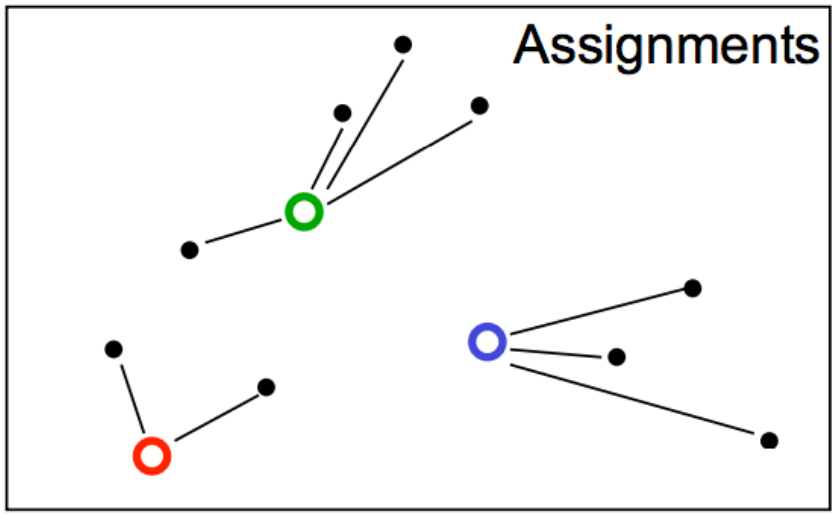
- Sometimes the data form clusters, where examples within a cluster are similar to each other, and examples in different clusters are dissimilar:

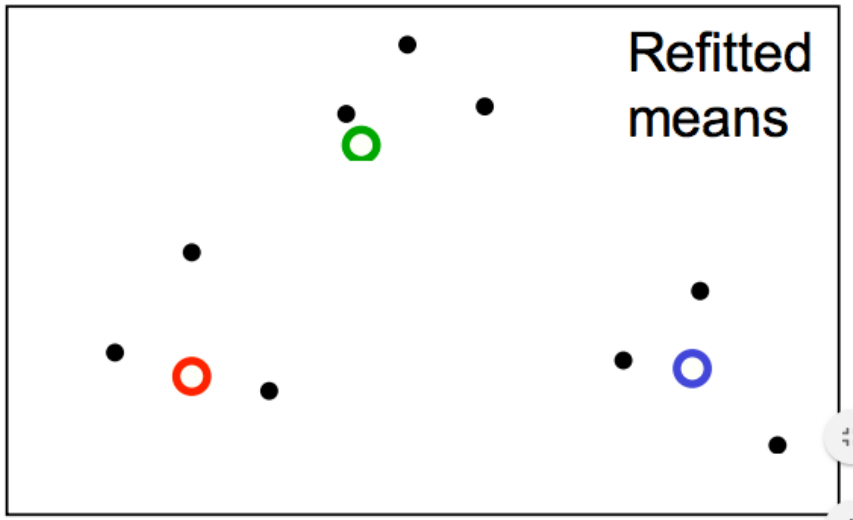


- Such a distribution is multimodal, since it has multiple modes, or regions of high probability mass.
- Grouping data points into clusters, with no labels, is called clustering
- E.g. clustering machine learning papers based on topic (deep learning, Bayesian models, etc.)
  - This is an overly simplistic model — more on that later

K-Means

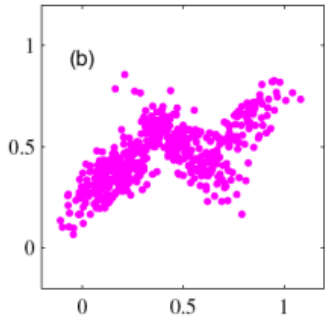
- First, let's look at a simple clustering algorithm, called k-means.
- This is an iterative algorithm. In each iteration, we keep track of:
  - An assignment of data points to clusters
  - The center of each cluster
- Start with random cluster locations, then alternate between:
  - Assignment step: assign each data point to the nearest cluster
  - Refitting step: move each cluster center to the average of its data points





K-Means

- Each iteration can be shown to decrease a particular cost function: the sum of squared distances from data points to their corresponding cluster centers.
  - More on this in CSC411.
- Problem: what if the clusters aren't spherical?



- Let's instead treat clustering as a distribution modeling problem.
  - Last lecture, we fit Gaussian distributions to data.
  - To model multimodal distributions, let's fit a mixture model, where each data point belongs to a different component.
  - E.g., in a mixture of Gaussians, each data point comes from one of several different Gaussian distributions.
  - We don't need to use Gaussians — we can pick whatever distribution best represents our data.

Mixture of Gaussians

- In a mixture model, we define a generative process where we first sample the latent variable z, and then sample the observations $\mathbf{x}$ from a distribution which depends on z.

$$p(z, \mathbf{x}) = p(z)\, p(\mathbf{x} \mid z).$$

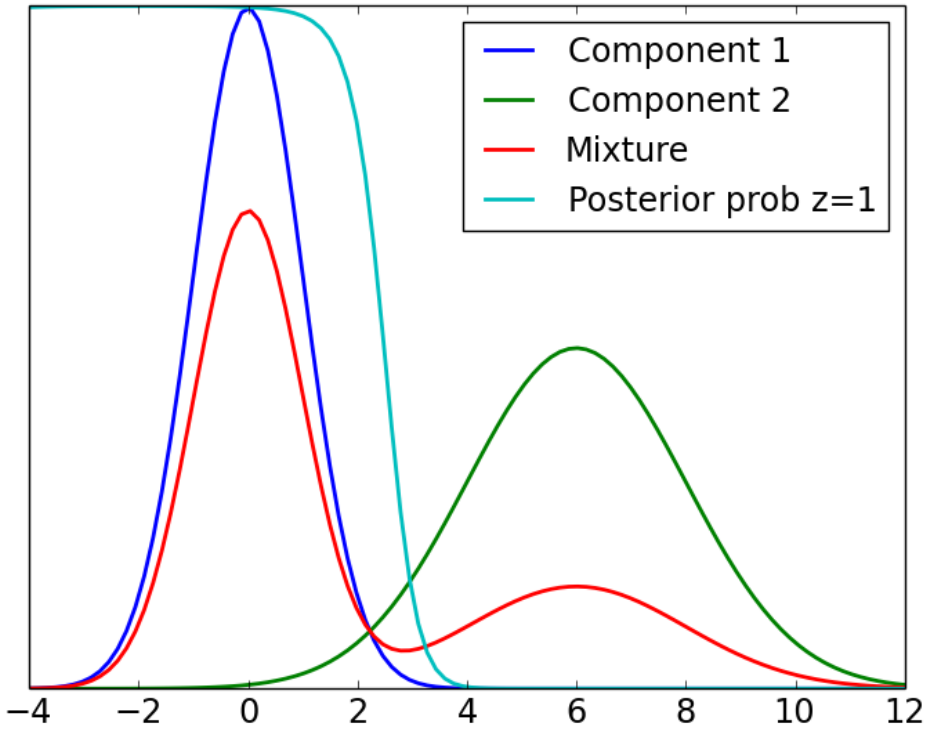
- E.g. mixture of Gaussians:

$$z \sim \text{Multinomial}(0.7, 0.3) \tag{1}$$
$$x \mid z = 1 \sim \text{Gaussian}(0, 1) \tag{2}$$
$$x \mid z = 2 \sim \text{Gaussian}(6, 2) \tag{3}$$

- The probabilities used to sample z are called the mixing proportions.

Mixture of Gaussians

Example:



- The probability density function over x is defined by marginalizing, or summing out, z:

$$p(\mathbf{x}) = \sum_{k=1}^{K} \Pr(z = k)\, p(\mathbf{x} \mid z = k)$$

Posterior Inference

- Suppose we know the model parameters (mixture probabilities and component means and variances)
- In posterior inference, we infer the posterior over z using Bayes' Rule:

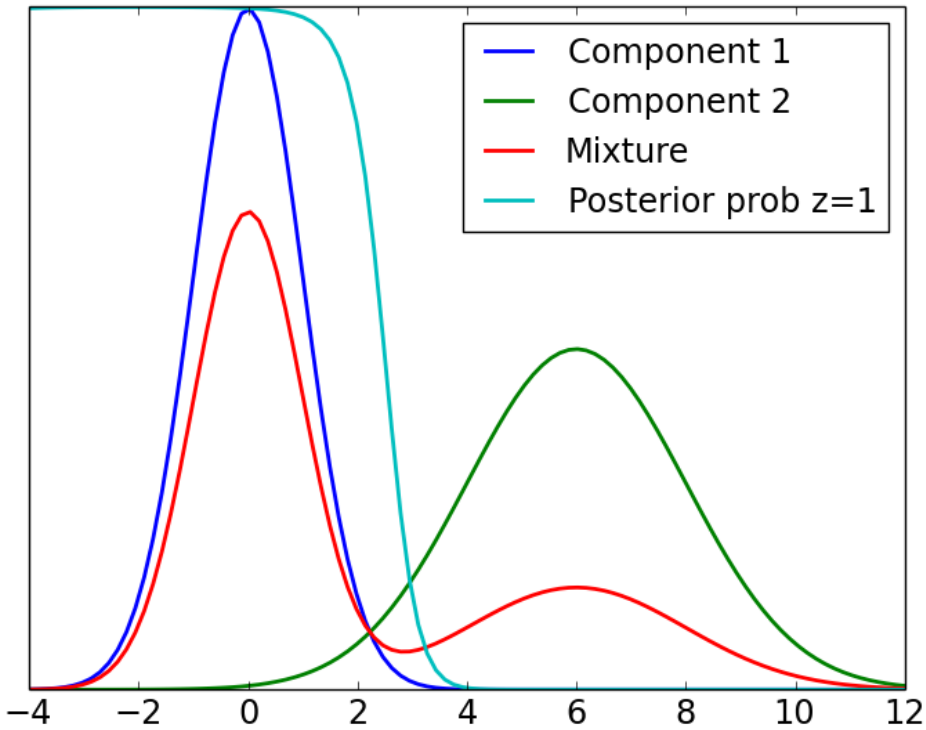
$$p(z \mid \mathbf{x}) \propto p(z)\, p(\mathbf{x} \mid z).$$

- For a univariate Gaussian mixture with mixing proportions $\boldsymbol{\pi}$

$$p(z = 1 \mid x) = \frac{\pi_1 \cdot \mathcal{N}(x; \mu_1, \sigma_1)}{\pi_1 \cdot \mathcal{N}(x; \mu_1, \sigma_1) + \pi_2 \cdot \mathcal{N}(x; \mu_2, \sigma_2)}$$

Posterior Inference

Example:



Posterior Inference

- Sometimes the observables aren't actually observed — then we say they're missing
- One use of probabilistic models is to make predictions about missing data
  - E.g. image completion, which you'll do in Assignment 4
- Analogously to Bayesian parameter estimation, we use the posterior predictive distribution:

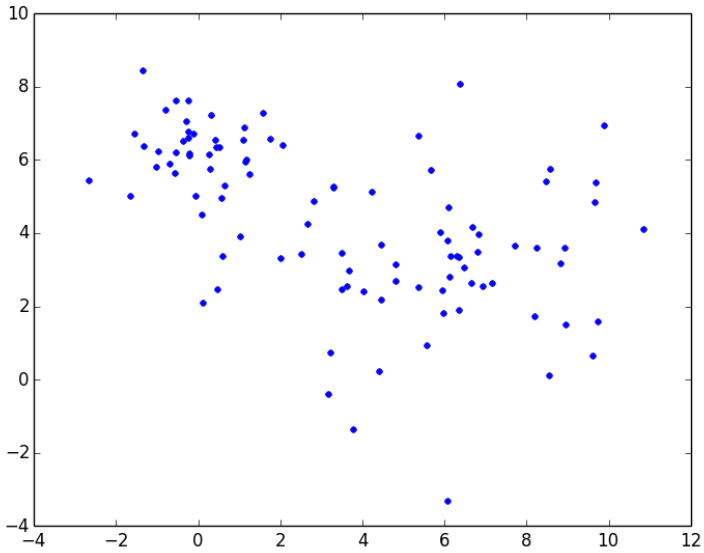
$$p(x_2 \mid x_1) = \sum_z \underbrace{p(z \mid x_1)}_{\text{posterior}} \, p(x_2 \mid z, x_1).$$

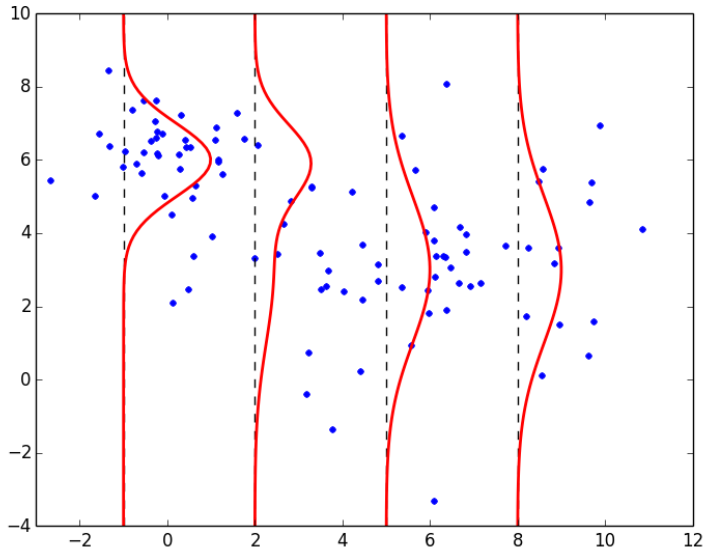
- If the dimensions of $\mathbf{x}$ are conditionally independent given z, this is just a reweighting of the original mixture model, where we use the posterior rather than the prior.

$$p(x_2 \mid x_1) = \sum_z \underbrace{p(z \mid x_1)}_{\text{posterior}} \underbrace{p(x_2 \mid z)}_{\text{component PDF}}.$$

Posterior Inference

Example:





Fully worked-through example in the lecture notes.

Parameter Learning

- Now let's talk about learning. We need to fit two sets of paramters:
  - The mixture probabilities $\pi_k = \Pr(z = k)$
  - The mean μ_k and standard deviation σ_k for each component
- If someone hands us the values of all the latent variables, it's easy to fit the parameters using maximum likelihood.

$$\begin{aligned}
\ell &= \log \prod_{i=1}^{N} p(z^{(i)}, x^{(i)}) \\
&= \log \prod_{i=1}^{N} p(z^{(i)}) p(x^{(i)} \mid z^{(i)}) \\
&= \sum_{i=1}^{N} \underbrace{\log p(z^{(i)})}_{\boldsymbol{\pi}} + \underbrace{\log p(x^{(i)} \mid z^{(i)})}_{\mu_k,\ \sigma_k}
\end{aligned}$$

Parameter Learning

- Let $r_k^{(i)}$ be the indicator variable for $z^{(i)} = k$. This is called the responsibilitiy
- Solving for the mixing probabilities:

$$\ell = \sum_{i=1}^{N} \log p(z^{(i)}) + \log p(x^{(i)} \mid z^{(i)})$$

$$= \text{const} + \sum_{i=1}^{N} \log p(z^{(i)})$$

- This is just the maximum likelihood problem for the multinomial distirbution. The solution is just the empirical proabilities, which we can write as:

$$\pi_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)}$$

Parameter Learning

- Solving for the mean parameter μ_k for component k:

$$\begin{aligned}\ell &= \sum_{i=1}^{N} \log p(z^{(i)}) + \log p(x^{(i)} \mid z^{(i)}) \\ &= \text{const} + \sum_{i=1}^{N} \log p(x^{(i)} \mid z^{(i)}) \\ &= \text{const} + \sum_{i=1}^{N} r_k^{(i)} \log \mathcal{N}(x^{(i)}; \mu_k, \sigma_k)\end{aligned}$$

- This is just maximum likelihood for the parameters of a Gaussian distribution, where only certain data points count.
- Solution:

$$\mu_k \leftarrow \frac{\sum_{i=1}^{N} r_k^{(i)} x^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}}$$

Expectation-Maximization

- We've seen how to do two things:
  - Given the model parameters, compute the posterior over the latent variables
  - Given the latent variables, find the maximum likelihood parameters
- But we don't know the parameters or latent variables, so we have a chicken-and-egg problem.
- Remember k-means? We iterated between an assignment step and a refitting step.
- Expectation-Maximization (E-M) is an analogous procedure which alternates bewteen two steps:
  - **Expectation step (E-step):** Compute the posterior expectations of the latent variables z
  - **Maximization step (M-step):** Solve for the maximum likelihood parameters given the full set of x's and z's

Expectation-Maximization

E-step:

- This is like the assignment step in k-means, except that we assign fractional responsibilities.

$$\begin{aligned} r_k^{(i)} &\leftarrow \Pr(z^{(i)} = k \mid x^{(i)}) \\ &\propto \pi_k \cdot \mathcal{N}(x^{(i)}; \mu_k, \sigma_k) \end{aligned}$$

- This is just posterior inference, which we've already talked about.

Expectation-Maximization

M-step:

- Maximum likelihood with fractional counts:

$$\boldsymbol{\theta} \leftarrow \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \sum_{k=1}^{K} r_k^{(i)} \left[\log \Pr(z^{(i)} = k) + \log p(\mathbf{x}^{(i)} \mid z^{(i)} = k) \right]$$

- The maximum likelihood formulas we already saw don't depend on the responsibilities being 0 or 1. They also work with fractional responsibilities. E.g.,

$$\pi_k \leftarrow \frac{1}{N} \sum_{i=1}^{N} r_k^{(i)}$$

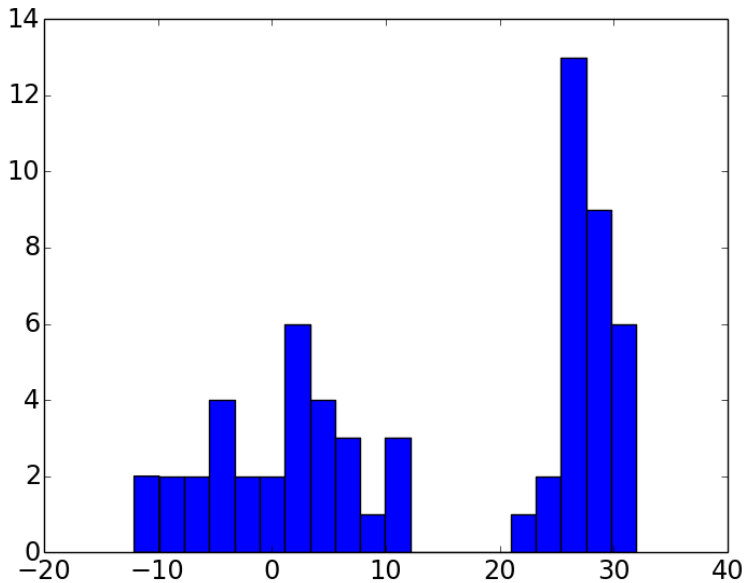
$$\mu_k \leftarrow \frac{\sum_{i=1}^{N} r_k^{(i)} x^{(i)}}{\sum_{i=1}^{N} r_k^{(i)}}$$

Expectation-Maximization

- We initialize the model parameters randomly and then repeatedly apply the E-step and M-step.
- Each step can be shown to increase the log-likelihood, but this is beyond the scope of the class.
  - Optional mathematical justification in the lecture notes, in case you're interested.
  - Also, there's a full explanation in CSC 411.
  - Next lecture, we'll fit a different latent variable model by doing gradient descent on the parameters. This will turn out to have an EM-like flavor.

Example

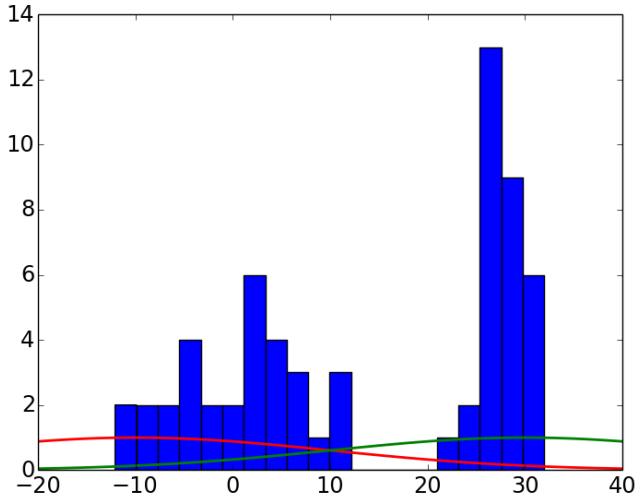
- Suppose we recorded a bunch of temperatures in March for Toronto and Miami, but forgot to record which was which, and they're all jumbled together.



- Let's try to separate them out using a mixture of Gaussians and E-M.

Example

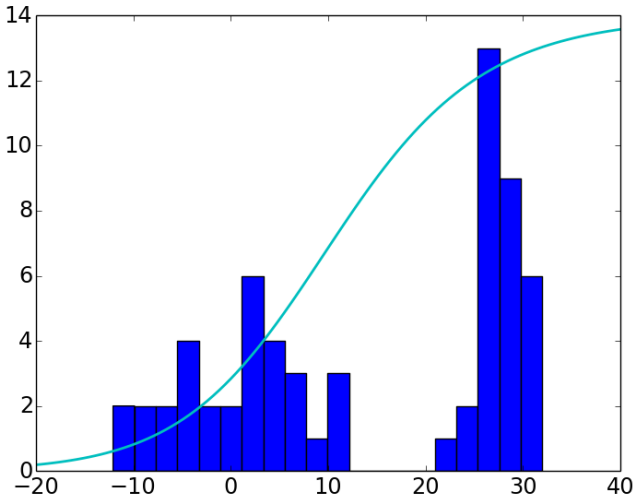
Random initialization



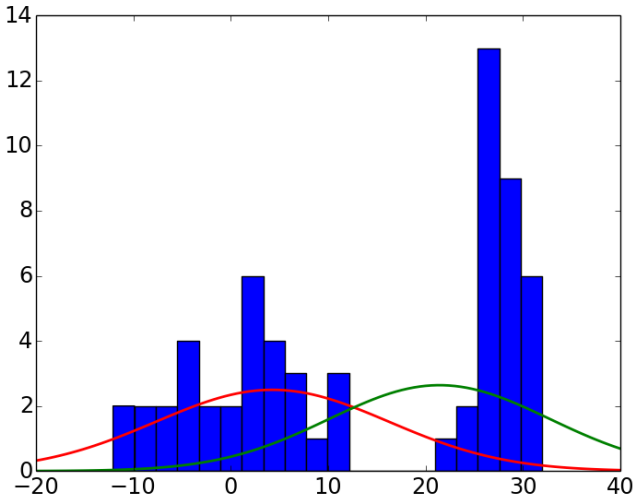
Example

Step 1:

E-step



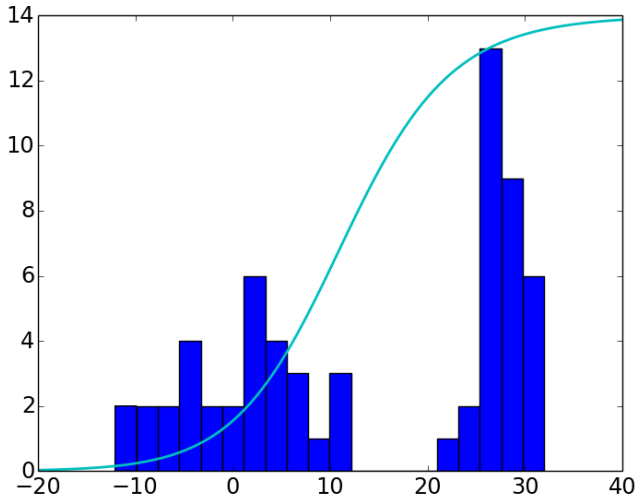
M-step



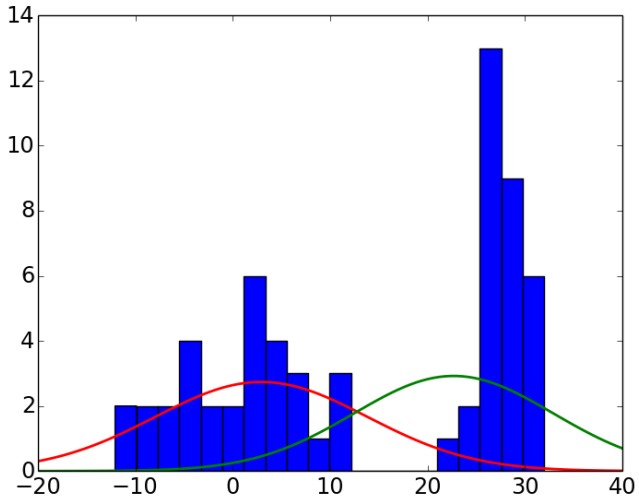
Example

Step 2:

E-step



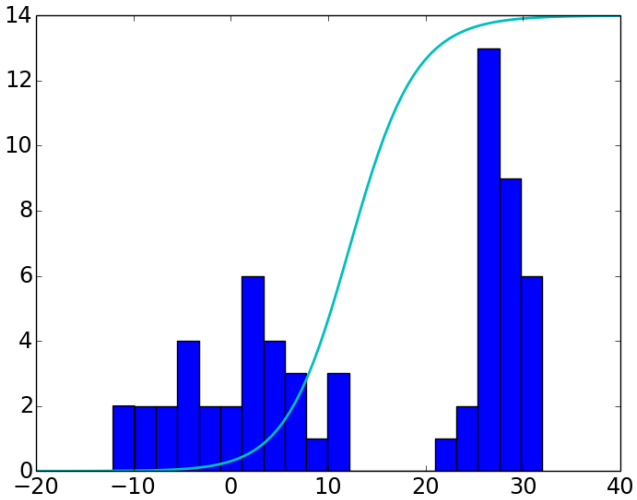
M-step



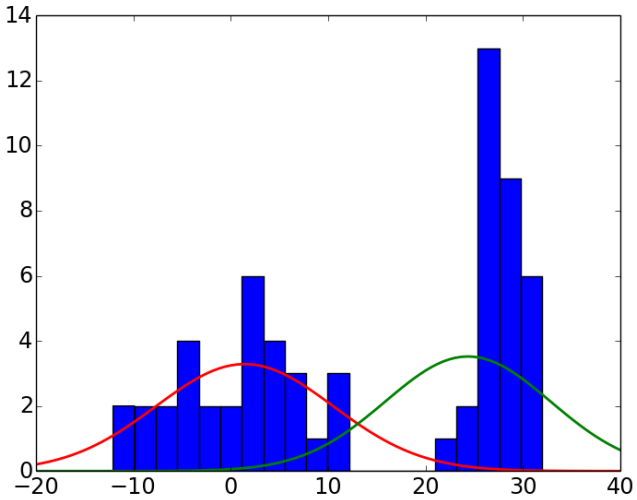
Example

Step 3:

E-step



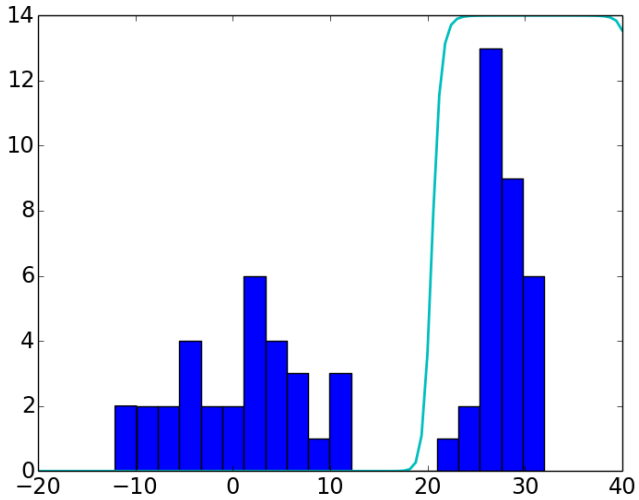
M-step



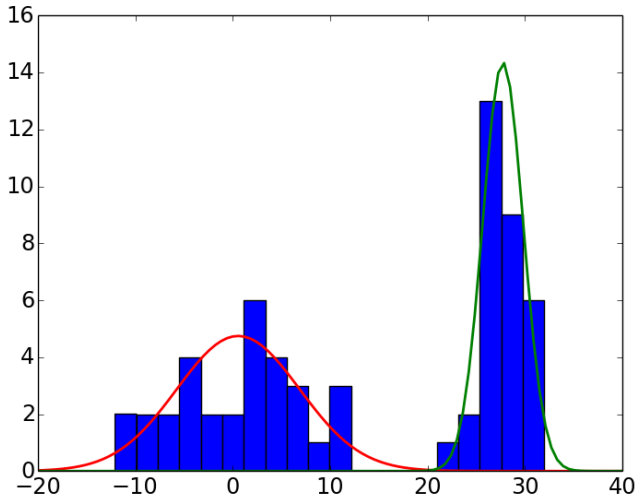
Example

Step 10:

E-step

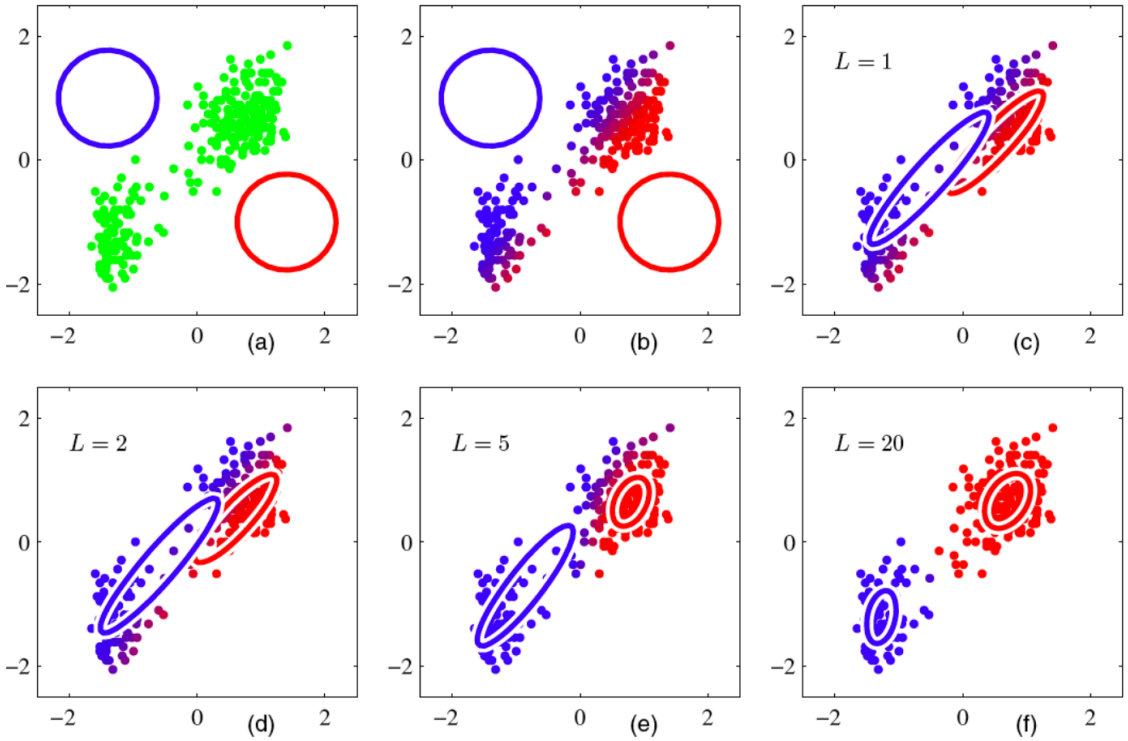


M-step



Expectation-Maximization

- We used univariate Gaussian components for simplicity, but other distributions are possible.
- Multivariate Gaussians:



- In Programming Assignment 4, you will fit a mixture of Bernoullis model.

Odds and Ends

- E-M can get stuck in local optima.
  - Initialize from k-means, which can be more robust in practice
  - Use multiple random restarts and pick the one with the best objective function
- Mixture models are a localist representation: the latent variables take values in a small discrete set.
  - We can use more complex distributions over latent variables to get a distributed representation.
  - The difficulty is posterior inference: while this is easy to do exactly for mixture models, it's intractable in general, and we'll need to make approximations.