

# *K*-Means and Gaussian Mixture Models

David Rosenberg, Brett Bernstein

New York University

April 25, 2017

## Intro Question

# Intro Question

Consider the following probability model for generating data.

- 1 Roll a weighted  $k$ -sided die to choose a label  $z \in \{1, \dots, k\}$ . Let  $\pi$  denote the PMF for the die.
- 2 Draw  $x \in \mathbf{R}^d$  randomly from the multivariate normal distribution  $\mathcal{N}(\mu_z, \Sigma_z)$ .

Solve the following questions.

- 1 What is the joint distribution of  $x, z$  given  $\pi$  and the  $\mu_z, \Sigma_z$  values?
- 2 Suppose you were given the dataset  $\mathcal{D} = \{(x_1, z_1), \dots, (x_n, z_n)\}$ . How would you estimate the die weightings, and the  $\mu_z, \Sigma_z$  values?
- 3 How would you determine the label for a new datapoint  $x$ ?

## Intro Solution

- 1 The joint PDF/PMF is given by

$$p(x, z) = \pi(z) f(x; \mu_z, \Sigma_z)$$

where

$$f(x; \mu_z, \Sigma_z) = \frac{1}{\sqrt{|2\pi\Sigma_z|}} \exp\left(-\frac{1}{2}(x - \mu)^T \Sigma^{-1}(x - \mu)\right).$$

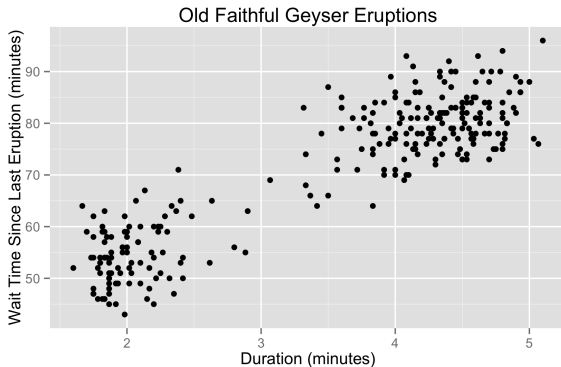
- 2 We could use maximum likelihood estimation. Our estimates are

$$\begin{aligned} n_z &= \sum_{i=1}^n \mathbf{1}(z_i = z) \\ \hat{\pi}(z) &= \frac{n_z}{n} \\ \hat{\mu}_z &= \frac{1}{n_z} \sum_{i: z_i = z} x_i \\ \hat{\Sigma}_z &= \frac{1}{n_z} \sum_{i: z_i = z} (x_i - \hat{\mu}_z)(x_i - \hat{\mu}_z)^T. \end{aligned}$$

- 3  $\arg \max_z p(x, z)$

# $K$ -Means Clustering

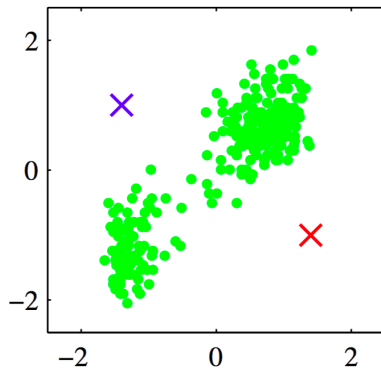
## Example: Old Faithful Geyser



- Looks like two clusters.
- How to find these clusters algorithmically?

## k-Means: By Example

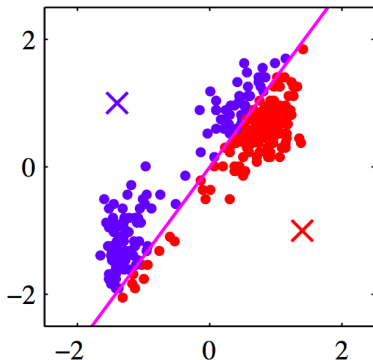
- Standardize the data.
- Choose two cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(a).

## k-means: by example

- Assign each point to closest center.

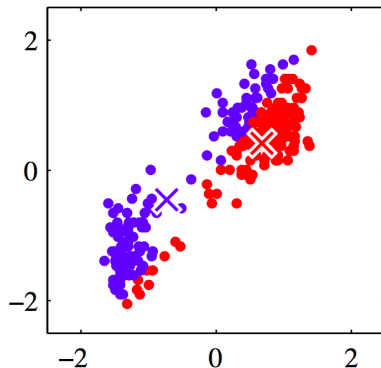


From Bishop's *Pattern recognition and machine learning*, Figure 9.1(b).



## k-means: by example

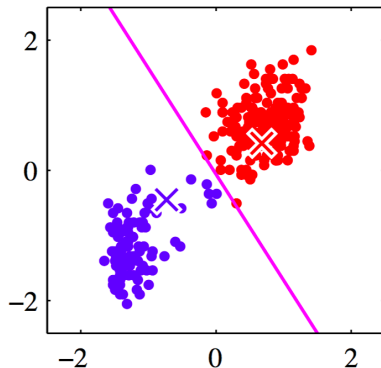
- Compute new class centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(c).

## k-means: by example

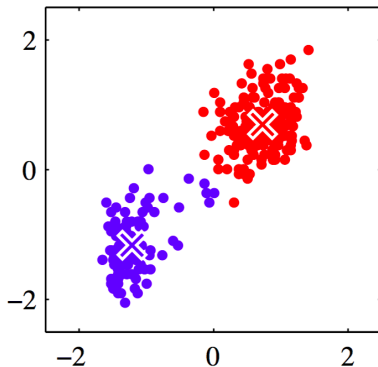
- Assign points to closest center.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(d).

## k-means: by example

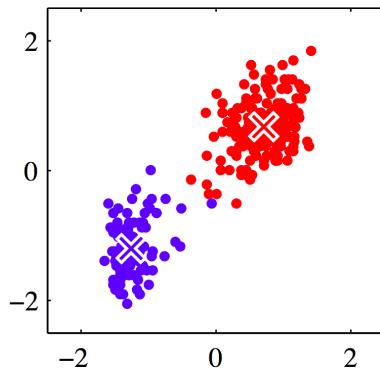
- Compute cluster centers.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(e).

## k-means: by example

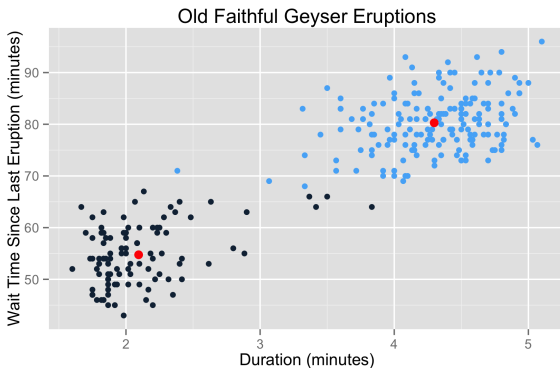
- Iterate until convergence.



From Bishop's *Pattern recognition and machine learning*, Figure 9.1(i).

# k-Means Algorithm: Standardizing the data

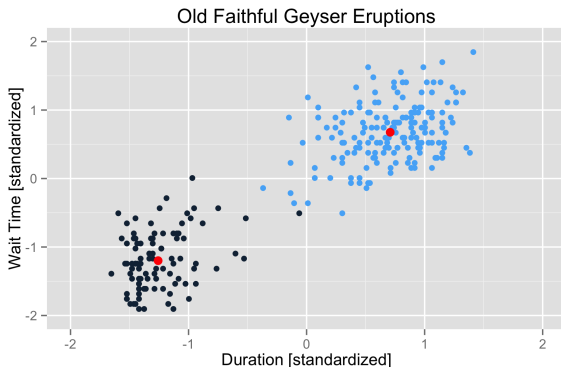
- Without standardizing:



- Blue and black show results of k-means clustering
- Wait time dominates the distance metric

# k-Means Algorithm: Standardizing the data

- With standardizing:



- Note several points have been reassigned from black to blue cluster.

## k-Means: Objective

- Let  $x_1, \dots, x_n$  denote the data points and  $\mu_1, \dots, \mu_k$  the cluster points.
- Define the objective  $\phi$  by

$$\phi(x, \mu) = \sum_{i=1}^n \|x_i - \mu_{c(x_i)}\|_2^2,$$

where  $\mu_{c(x_i)}$  is the cluster point associated to  $x_i$ .

- Then  $\phi$  decreases at every round of  $k$ -means. Why?

## k-Means: Objective

- Let  $x_1, \dots, x_n$  denote the data points and  $\mu_1, \dots, \mu_k$  the cluster points.
- Define the objective  $\phi$  by

$$\phi(x, \mu) = \sum_{i=1}^n \|x_i - \mu_{c(x_i)}\|_2^2,$$

where  $\mu_{c(x_i)}$  is the cluster point associated to  $x_i$ .

- Then  $\phi$  decreases at every round of  $k$ -means. Why?
- Selecting mean of all associated data points improves objective.
- Selecting closest cluster point for each data points improves objective.



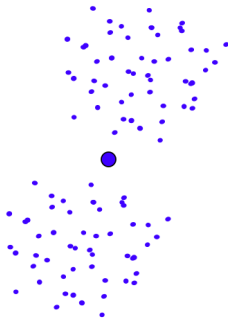
## $k$ -Means: Failure Cases

# $k$ -Means: Suboptimal Local Minimum

- The clustering for  $k = 3$  below is a local minimum, but suboptimal:



Would be better to have  
one cluster here



... and two clusters here

## $k$ -Means++

- Improvement on  $k$ -means by controlling the random initialization of the cluster centers.
- Randomly choose first center amongst the data points.
- For each of the remaining  $k - 1$  centers:
  - Compute the distance from each data point to the closest already chosen center.
  - Randomly choose a point as the new center with probability proportional to its computed distance squared.
- If we let  $\phi$  denote the total sum of squares distances from each point to the closest cluster, then  $k$ -means++ has

$$E[\phi] \leq 8(\log k + 2)\phi_{\text{OPT}},$$

where  $\phi_{\text{OPT}}$  is from the optimal  $k$ -cluster assignment.

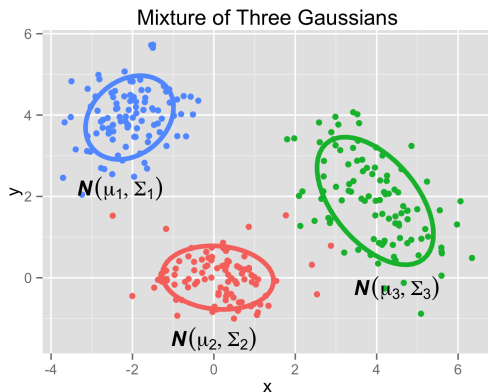
# Gaussian Mixture Models

# Probabilistic Model for Clustering

- Let's consider a **generative model** for the data.
- Suppose
  - 1 There are  $k$  clusters.
  - 2 We have a probability density for each cluster.
- Generate a point as follows
  - 1 Choose a random cluster  $z \in \{1, 2, \dots, k\}$ .
  - 2 Choose a point from the distribution for cluster  $Z$ .

# Gaussian Mixture Model ( $k = 3$ )

- 1 Choose  $z \in \{1, 2, 3\}$  with  $p(1) = p(2) = p(3) = \frac{1}{3}$ .
- 2 Choose  $x \mid z \sim \mathcal{N}(X \mid \mu_z, \Sigma_z)$ .

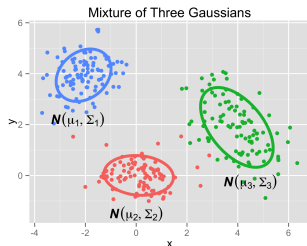


# Gaussian Mixture Model Parameters ( $k$ Components)

Cluster probabilities:  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means:  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$



For now, **suppose all these parameters are known.**  
 We'll discuss how to **learn** or **estimate** them later.

# Gaussian Mixture Model: Joint Distribution

- Factorize the joint distribution:

$$\begin{aligned} p(x, z) &= p(z)p(x | z) \\ &= \pi_z \mathcal{N}(x | \mu_z, \Sigma_z) \end{aligned}$$

- $\pi_z$  is probability of choosing cluster  $z$ .
  - $x | z$  has distribution  $\mathcal{N}(\mu_z, \Sigma_z)$ .
  - $z$  corresponding to  $x$  is the true cluster assignment.
- Suppose we know the model parameters  $\pi_z, \mu_z, \Sigma_z$ .
- Then we can easily compute the joint  $p(x, z)$ .



# Latent Variable Model

- We observe  $x$ .
- In the intro problem we had labeled data. Here we don't observe  $z$ , the cluster assignment.
- Cluster assignment  $z$  is called a **hidden variable** or **latent variable**.

## Definition

A **latent variable model** is a probability model for which certain variables are never observed.

e.g. The Gaussian mixture model is a latent variable model.

# The GMM “Inference” Problem

- We observe  $x$ . We want to know  $z$ .
- The conditional distribution of the cluster  $z$  given  $x$  is

$$p(z | x) = p(x, z) / p(x)$$

- The conditional distribution is a **soft assignment** to clusters.
- A **hard assignment** is

$$z^* = \arg \max_{z \in \{1, \dots, k\}} p(z | x).$$

- So if we have the model, clustering is trivial.

# Mixture Models

# Gaussian Mixture Model: Marginal Distribution

- The **marginal distribution** for a single observation  $x$  is

$$\begin{aligned} p(x) &= \sum_{z=1}^k p(x, z) \\ &= \sum_{z=1}^k \pi_z \mathcal{N}(x \mid \mu_z, \Sigma_z) \end{aligned}$$

- Note that  $p(x)$  is a convex combination of probability densities.
- This is a common form for a probability model...

# Mixture Distributions (or Mixture Models)

## Definition

A probability density  $p(x)$  represents a **mixture distribution** or **mixture model**, if we can write it as a **convex combination** of probability densities. That is,

$$p(x) = \sum_{i=1}^k w_i p_i(x),$$

where  $w_i \geq 0$ ,  $\sum_{i=1}^k w_i = 1$ , and each  $p_i$  is a probability density.

- In our Gaussian mixture model,  $x$  has a **mixture distribution**.
- More constructively, let  $S$  be a set of probability distributions:
  - 1 Choose a distribution randomly from  $S$ .
  - 2 Sample  $x$  from the chosen distribution.
- Then  $x$  has a mixture distribution.

# Learning in Gaussian Mixture Models

# The GMM “Learning” Problem

- Given data  $x_1, \dots, x_n$  drawn from a GMM,
- Estimate the parameters:

Cluster probabilities :  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means :  $\mu = (\mu_1, \dots, \mu_k)$

Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

- Once we have the parameters, we're done.
- Just do “inference” to get cluster assignments.

# Estimating/Learning the Gaussian Mixture Model

- One approach to learning is **maximum likelihood**
  - find parameter values that give **observed data** the **highest likelihood**.
- The model likelihood for  $\mathcal{D} = \{x_1, \dots, x_n\}$  is

$$\begin{aligned} L(\pi, \mu, \Sigma) &= \prod_{i=1}^n p(x_i) \\ &= \prod_{i=1}^n \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z). \end{aligned}$$

- As usual, we'll take our objective function to be the log of this:

$$J(\pi, \mu, \Sigma) = \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}$$



# Properties of the GMM Log-Likelihood

- GMM log-likelihood:

$$J(\pi, \mu, \Sigma) = \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i | \mu_z, \Sigma_z) \right\}$$

- Let's compare to the log-likelihood for a single Gaussian:

$$\begin{aligned} & \sum_{i=1}^n \log \mathcal{N}(x_i | \mu, \Sigma) \\ &= -\frac{nd}{2} \log(2\pi) - \frac{n}{2} \log |\Sigma| - \frac{1}{2} \sum_{i=1}^n (x_i - \mu)' \Sigma^{-1} (x_i - \mu) \end{aligned}$$

- For a single Gaussian, the log cancels the exp in the Gaussian density.
  - $\implies$  Things simplify a lot.
- For the GMM, the sum inside the log prevents this cancellation.
  - $\implies$  Expression more complicated. No closed form expression for MLE.

## Issues with MLE for GMM

# Identifiability Issues for GMM

- Suppose we have found parameters

Cluster probabilities :  $\pi = (\pi_1, \dots, \pi_k)$

Cluster means :  $\mu = (\mu_1, \dots, \mu_k)$

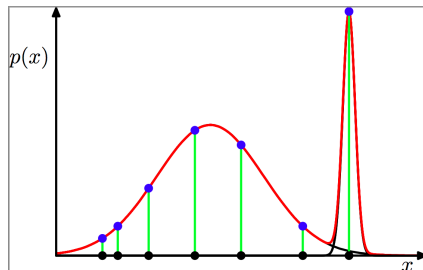
Cluster covariance matrices:  $\Sigma = (\Sigma_1, \dots, \Sigma_k)$

that are at a local minimum.

- What happens if we shuffle the clusters? e.g. Switch the labels for clusters 1 and 2.
- We'll get the same likelihood. How many such equivalent settings are there?
- Assuming all clusters are distinct, there are  $k!$  equivalent solutions.
- Not a problem per se, but something to be aware of.

# Singularities for GMM

- Consider the following GMM for 7 data points:



- Let  $\sigma^2$  be the variance of the skinny component.
- What happens to the likelihood as  $\sigma^2 \rightarrow 0$ ?
- In practice, we end up in local minima that do not have this problem.
  - Or keep restarting optimization until we do.
- Bayesian approach or regularization will also solve the problem.

From Bishop's *Pattern recognition and machine learning*, Figure 9.7.

# Gradient Descent / SGD for GMM

- What about running gradient descent or SGD on

$$J(\pi, \mu, \Sigma) = - \sum_{i=1}^n \log \left\{ \sum_{z=1}^k \pi_z \mathcal{N}(x_i \mid \mu_z, \Sigma_z) \right\}?$$

- Can be done – but need to be clever about it.
- Each matrix  $\Sigma_1, \dots, \Sigma_k$  has to be positive semidefinite.
- How to maintain that constraint?
  - Rewrite  $\Sigma_i = M_i M_i^T$ , where  $M_i$  is an unconstrained matrix.
  - Then  $\Sigma_i$  is positive semidefinite.

# The EM Algorithm for GMM

## MLE for GMM

- From the intro questions, we know that we can solve the MLE problem if the cluster assignments  $z_i$  are known

$$\begin{aligned}
 n_z &= \sum_{i=1}^n \mathbf{1}(z_i = z) \\
 \hat{\pi}(z) &= \frac{n_z}{n} \\
 \hat{\mu}_z &= \frac{1}{n_z} \sum_{i: z_i = z} x_i \\
 \hat{\Sigma}_z &= \frac{1}{n_z} \sum_{i: z_i = z} (x_i - \hat{\mu}_z)(x_i - \hat{\mu}_z)^T.
 \end{aligned}$$

- In the EM algorithm we will modify the equations to handle our evolving **soft assignments**, which we will call **responsibilities**.

# Cluster Responsibilities: Some New Notation

- Denote the probability that observed value  $x_i$  comes from cluster  $j$  by

$$\gamma_i^j = \mathbb{P}(Z = j \mid X = x_i).$$

- The **responsibility** that cluster  $j$  takes for observation  $x_i$ .
- Computationally,

$$\begin{aligned} \gamma_i^j &= \mathbb{P}(Z = j \mid X = x_i). \\ &= p(Z = j, X = x_i) / p(x) \\ &= \frac{\pi_j \mathcal{N}(x_i \mid \mu_j, \Sigma_j)}{\sum_{c=1}^k \pi_c \mathcal{N}(x_i \mid \mu_c, \Sigma_c)} \end{aligned}$$

- The vector  $(\gamma_i^1, \dots, \gamma_i^k)$  is exactly the **soft assignment** for  $x_i$ .
- Let  $n_c = \sum_{i=1}^n \gamma_i^c$  be the “number” of points “soft assigned” to cluster  $c$ .



# EM Algorithm for GMM: Overview

- If we know  $\pi$  and  $\mu_j, \Sigma_j$  for all  $j$  then we can easily find  $\gamma_i^j = \mathbb{P}(Z = j \mid X = x_i)$ .
- If we know the (soft) assignments, we can easily find estimates for  $\pi, \mu_j, \Sigma_j$  for all  $j$ .
- Repeatedly alternate the previous 2 steps.

# EM Algorithm for GMM: Overview

- 1 Initialize parameters  $\mu, \Sigma, \pi$ .
- 2 “E step”. Evaluate the responsibilities using current parameters:

$$\gamma_i^j = \frac{\pi_j \mathcal{N}(x_i | \mu_j, \Sigma_j)}{\sum_{c=1}^k \pi_c \mathcal{N}(x_i | \mu_c, \Sigma_c)}, \quad \text{for } i = 1, \dots, n \text{ and } j = 1, \dots, k.$$

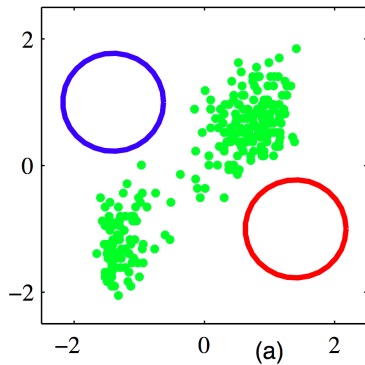
- 3 “M step”. Re-estimate the parameters using responsibilities. [Compare with intro question.]

$$\begin{aligned} \mu_c^{\text{new}} &= \frac{1}{n_c} \sum_{i=1}^n \gamma_i^c x_i \\ \Sigma_c^{\text{new}} &= \frac{1}{n_c} \sum_{i=1}^n \gamma_i^c (x_i - \mu_{\text{MLE}}) (x_i - \mu_{\text{MLE}})^T \\ \pi_c^{\text{new}} &= \frac{n_c}{n}, \end{aligned}$$

- 4 Repeat from Step 2, until log-likelihood converges.

# EM for GMM

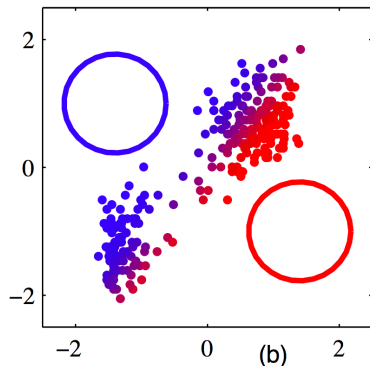
- Initialization



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

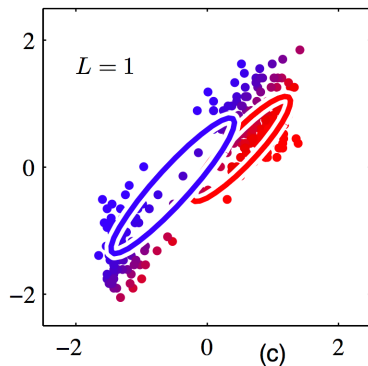
- First soft assignment:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

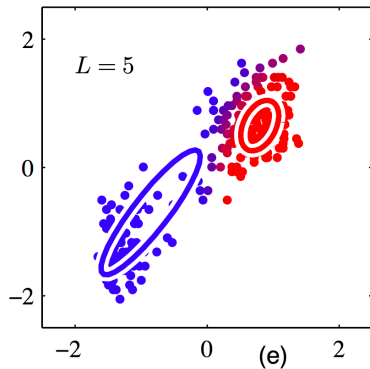
- First soft assignment:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

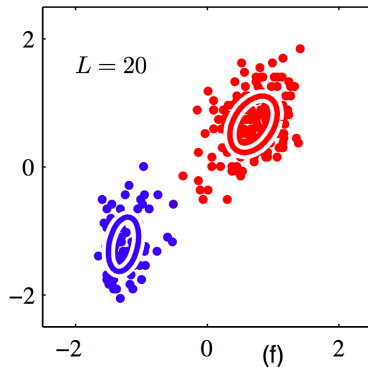
- After 5 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

# EM for GMM

- After 20 rounds of EM:



From Bishop's *Pattern recognition and machine learning*, Figure 9.8.

## Relation to $K$ -Means

- EM for GMM seems a little like  $k$ -means.
- In fact, there is a precise correspondence.
- First, fix each cluster covariance matrix to be  $\sigma^2 I$ .
- As we take  $\sigma^2 \rightarrow 0$ , the update equations converge to doing  $k$ -means.
- If you do a quick experiment yourself, you'll find
  - Soft assignments converge to hard assignments.
  - Has to do with the tail behavior (exponential decay) of Gaussian.
- Can use  $k$ -means++ to initialize parameters of EM algorithm.