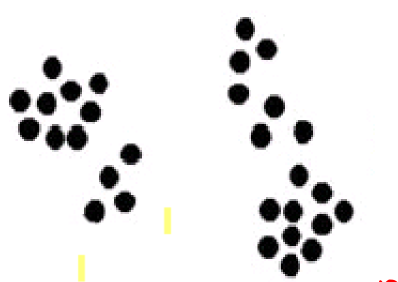
Machine Learning

10-701, Fall 2016

Expectation Maximization:
Mixture model ~~and HMM~~

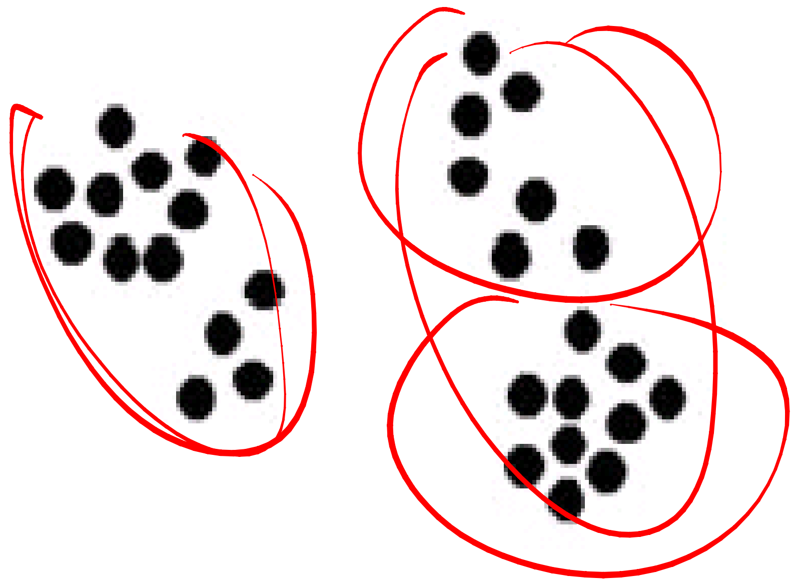
Eric Xing



Lecture 16, October 31, 2016

Reading: Chap. 9, 13, C.B book

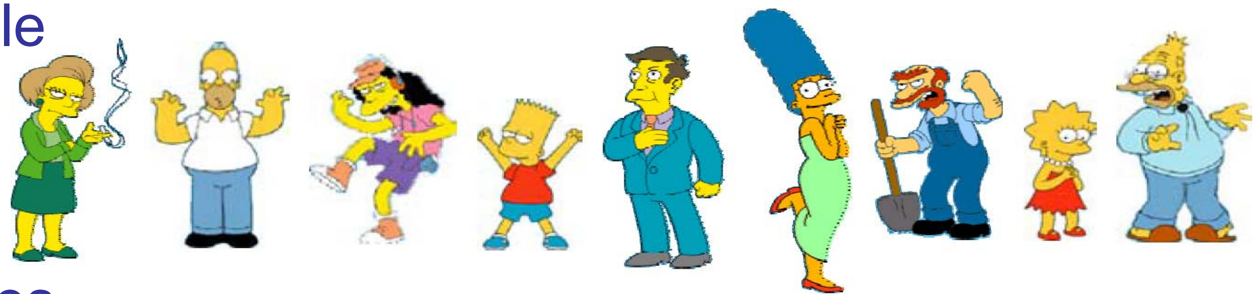
What is clustering?



- Are there any “grouping” them ?
- What is each group ?
- How many ?
- How to identify them?

Examples

- People



- Images

- Language

Piotr Pyotr Petros Pietro Pedro Pierre Piero Peter Peder Peka Peadar

- species

Issues for clustering

- What is a natural grouping among these objects?
  - Definition of "groupness"
- What makes objects “related”?
  - Definition of "similarity/distance"
- Representation for objects
  - Vector space? Normalization?
- How many clusters?
  - Fixed a priori?
  - Completely data driven?
    - Avoid “trivial” clusters - too large or small
- Clustering Algorithms
  - Partitional algorithms
  - Hierarchical algorithms
- Formal foundation and convergence

Minkowski metric

$$d(x,y) = \sqrt[r]{\sum_{i=1}^{p} |x_i - y_i|^r}$$

$r=2$

Partitioning Algorithms

- Partitioning method: Construct a partition of *n* objects into a set of *K* clusters
- Given: a set of objects and the number *K*
- Find: a partition of *K* clusters that optimizes the chosen partitioning criterion
  - Globally optimal: exhaustively enumerate all partitions
  - Effective heuristic methods: K-means and K-medoids algorithms

K-Means

Algorithm

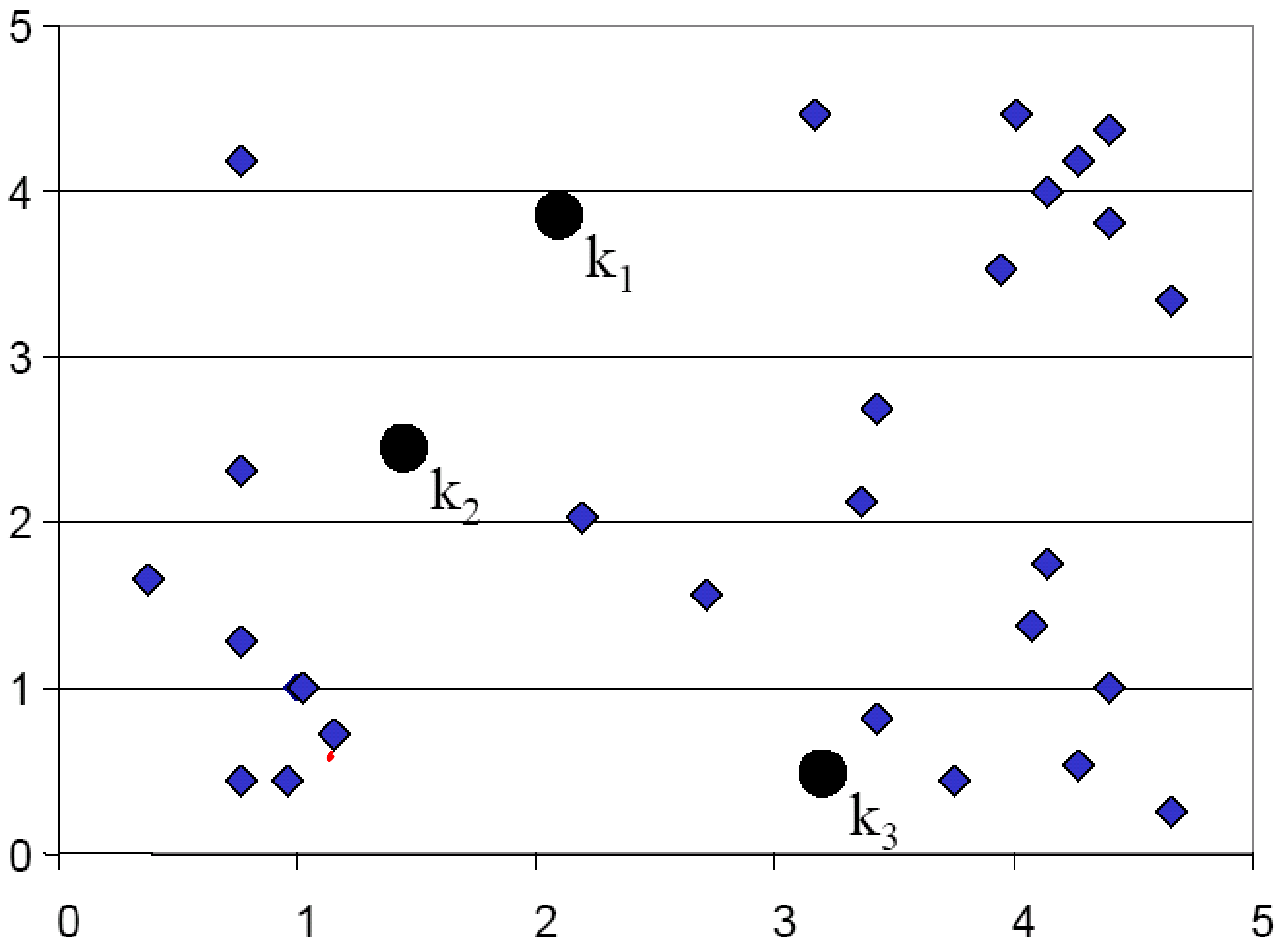
1. Decide on a value for *k*.
2. Initialize the *k* cluster centers randomly if necessary.
3. Decide the class memberships of the *N* objects by assigning them to the nearest cluster centroids (aka the center of gravity or mean)

$$\vec{\mu}_k = \frac{1}{\mathcal{C}_k} \sum_{i \in \mathcal{C}_k} \vec{x}_i$$

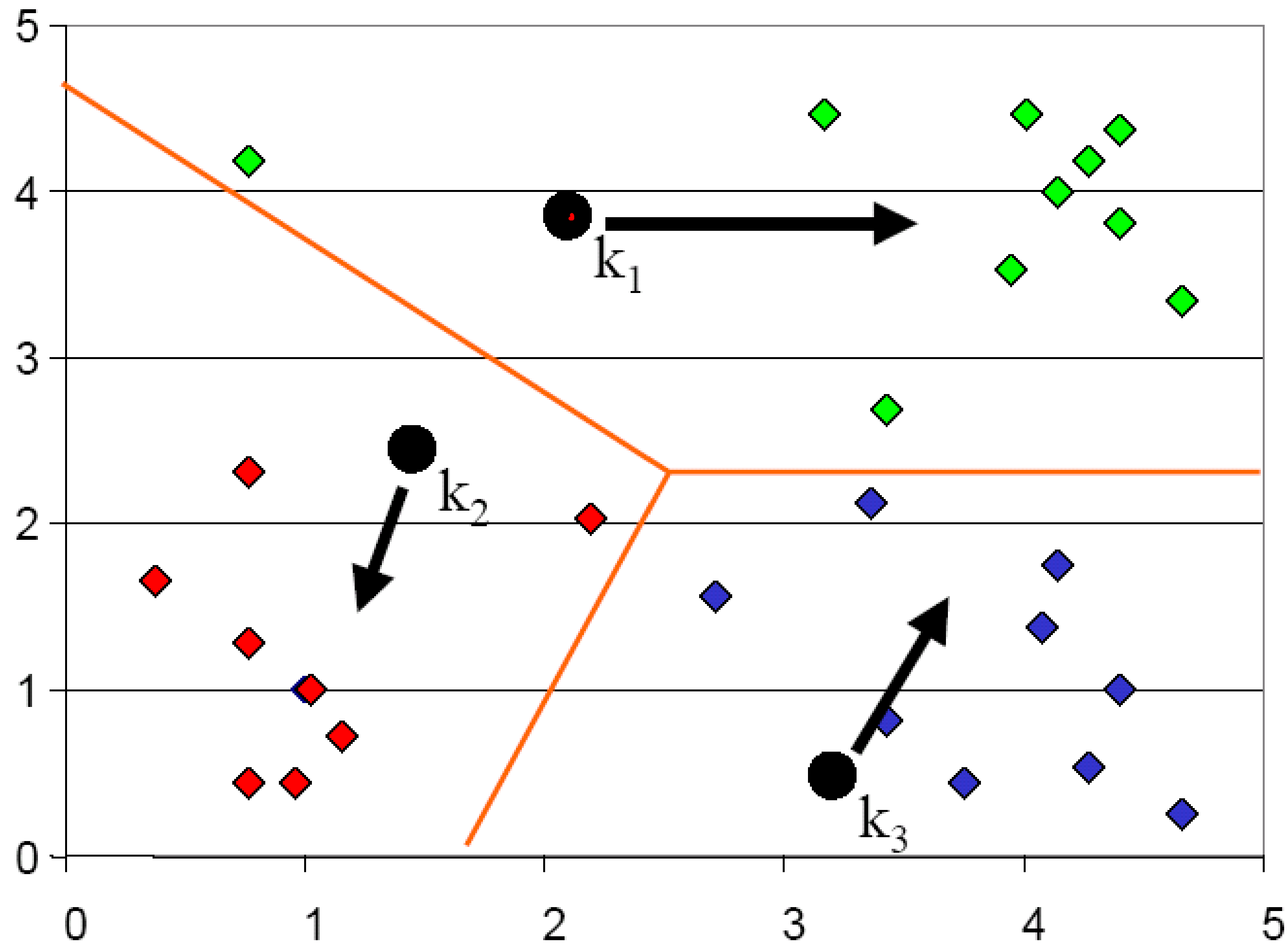
4. Re-estimate the *k* cluster centers, by assuming the memberships found above are correct.
5. If none of the *N* objects changed membership in the last iteration, exit. Otherwise go to 3.

K-means Clustering: Step 1

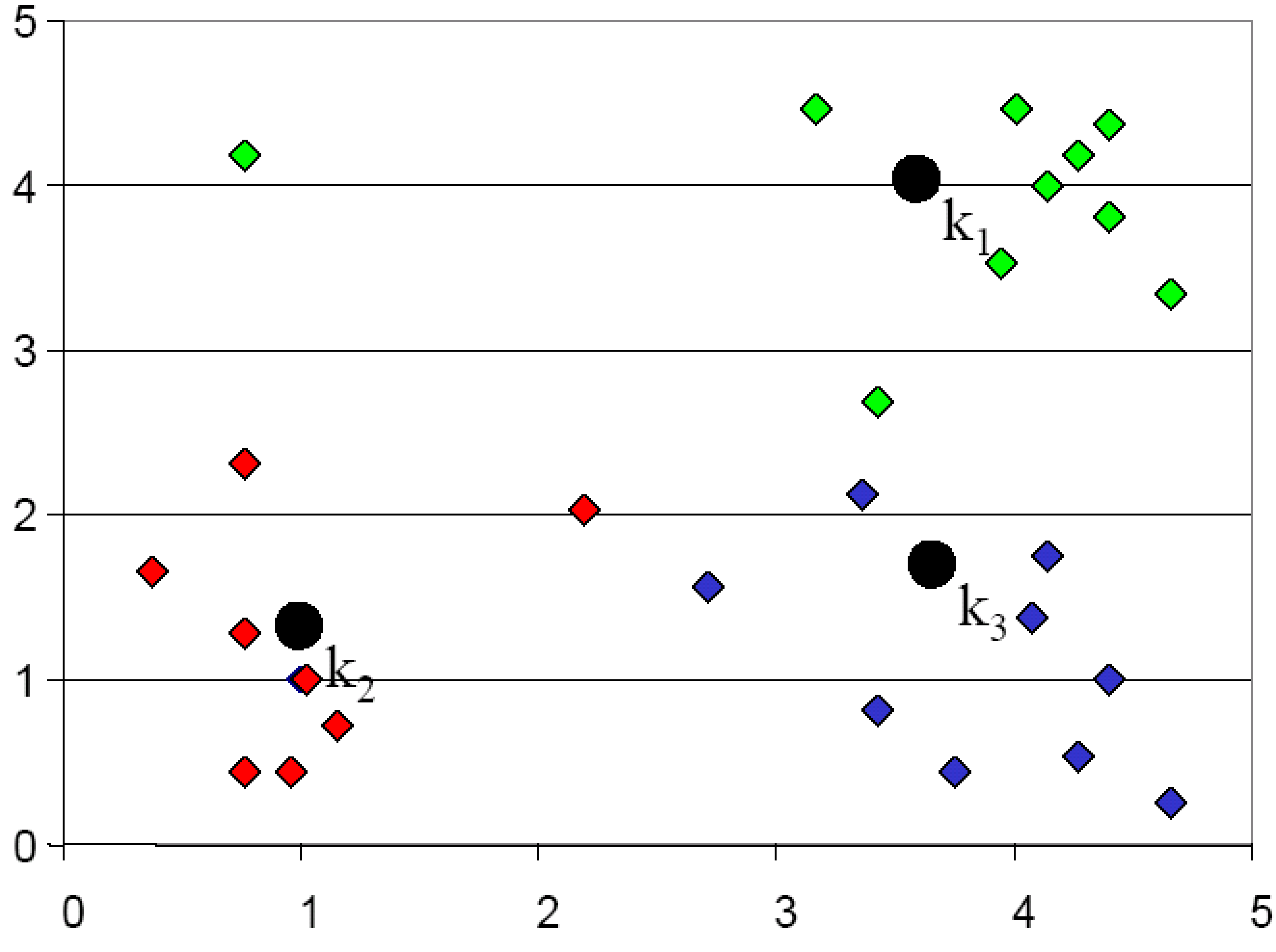
K=3



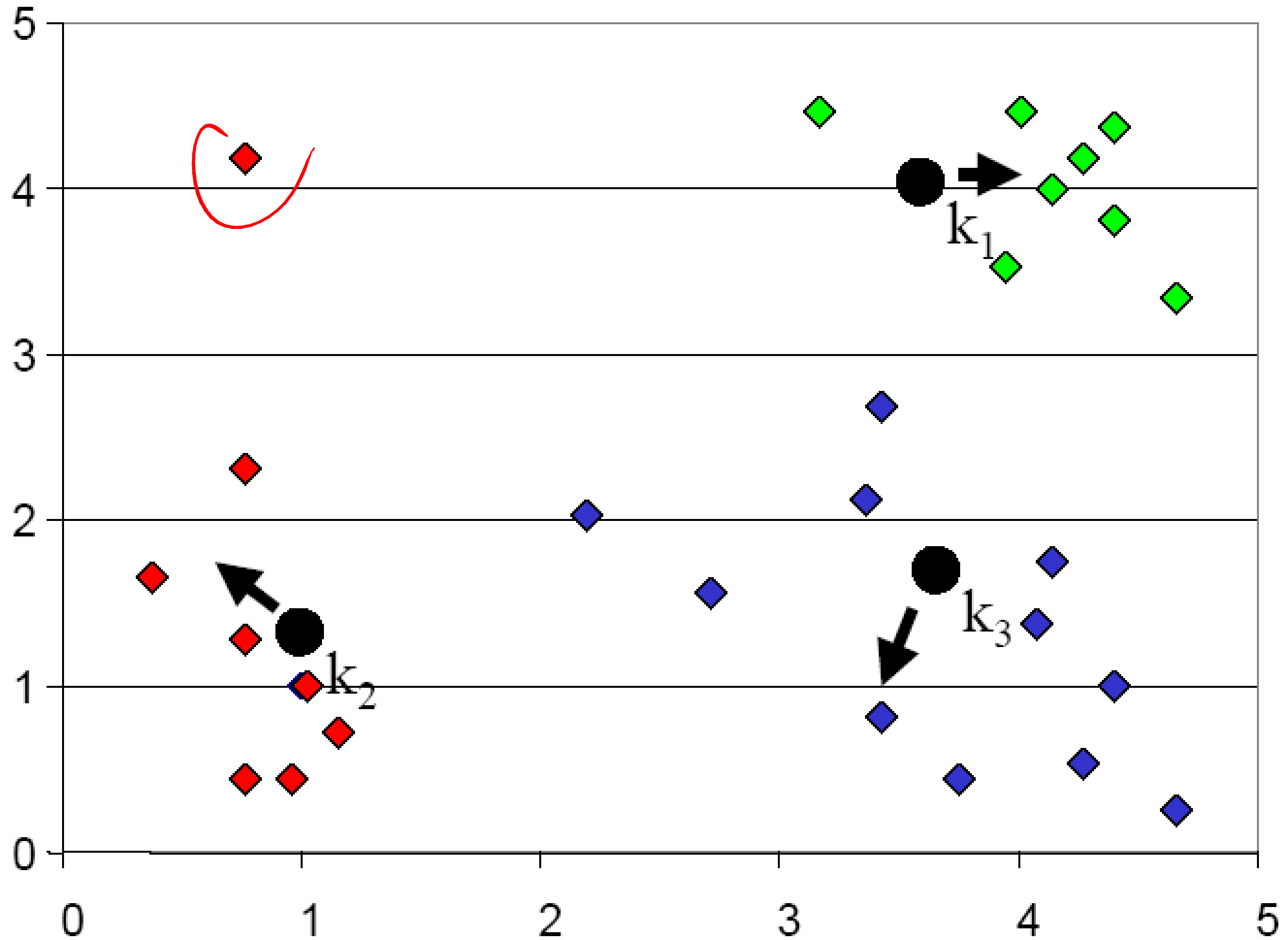
K-means Clustering: Step 2



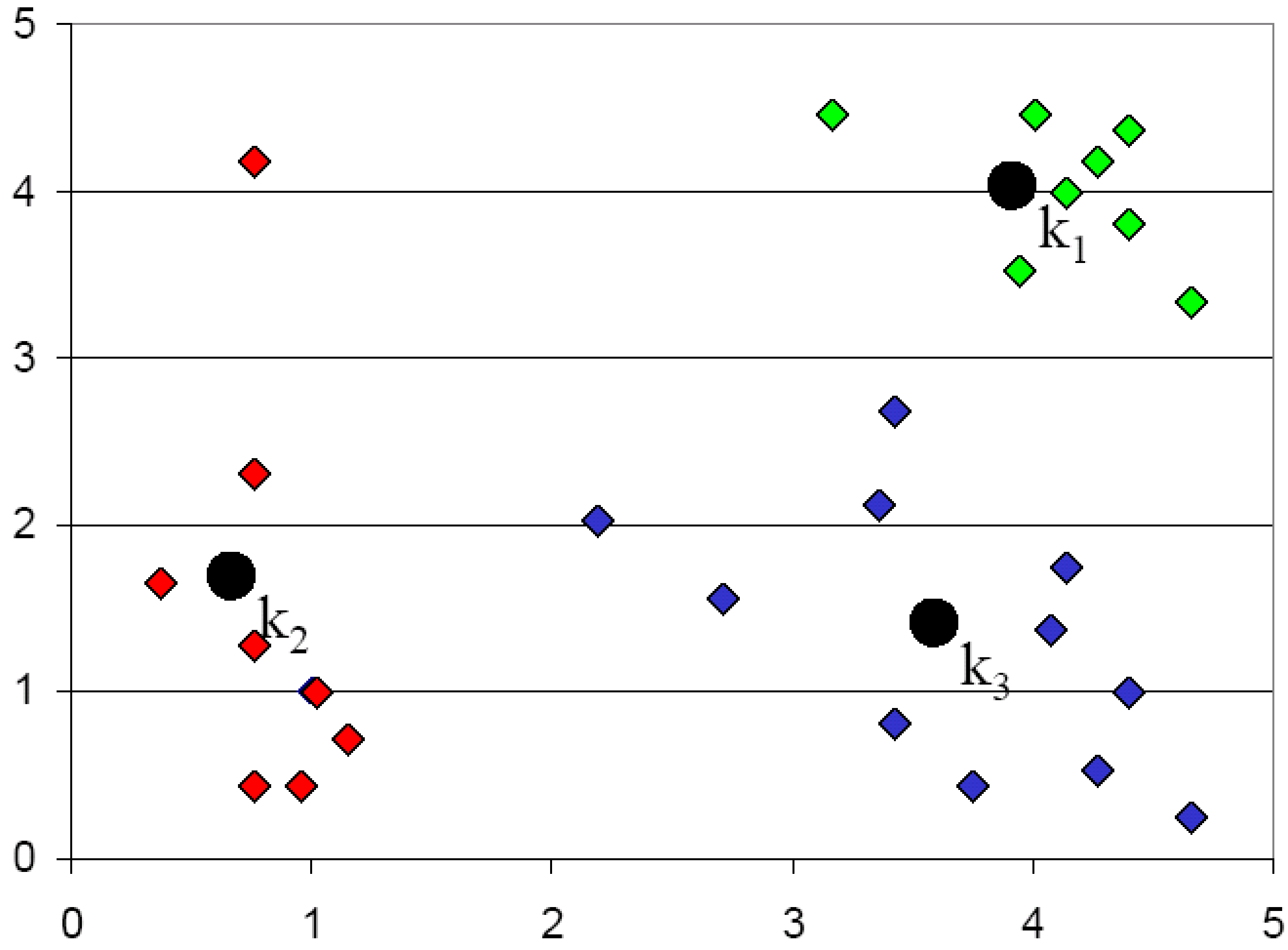
K-means Clustering: Step 3



K-means Clustering: Step 4



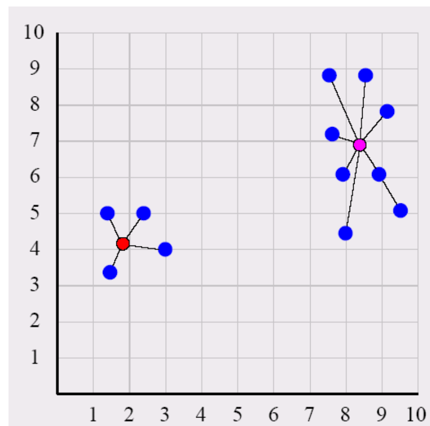
K-means Clustering: Step 5



Convergence

- Why should the K-means algorithm ever reach a fixed point?
  - -- A state in which clusters don't change.
- K-means is a special case of a general procedure known as the Expectation Maximization (EM) algorithm.
  - EM is known to converge.
  - Number of iterations could be large.
- Goodness measure
  - sum of squared distances from cluster centroid:

$$SD_{K_i} = \sum_{j=1}^{m_k} ||x_{ij} - \mu_i||^2 \qquad SD_K = \sum_{i=1}^{k} SD_{K_i}$$



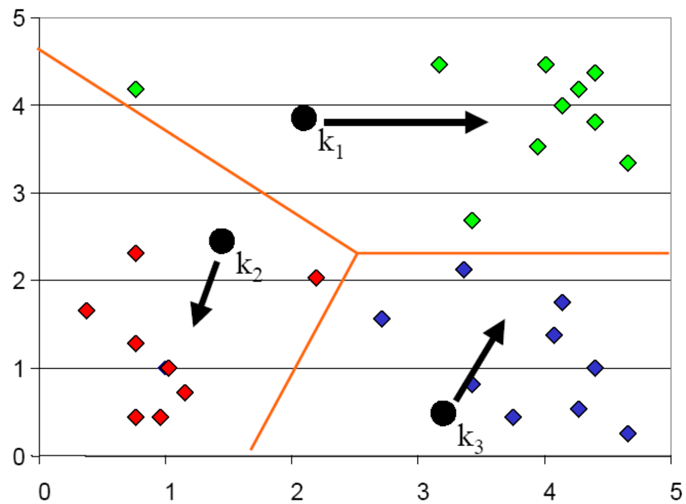
- Reassignment monotonically decreases SD since each vector is assigned to the closest centroid.

Time Complexity

- Computing distance between two objs is O(m) where m is the dimensionality of the vectors.
- Reassigning clusters: O(Kn) distance computations, or O(Knm).
- Computing centroids: Each doc gets added once to some centroid: O(nm).
- Assume these two steps are each done once for l iterations: O($lKnm$).

Seed Choice

- Results can vary based on random seed selection.

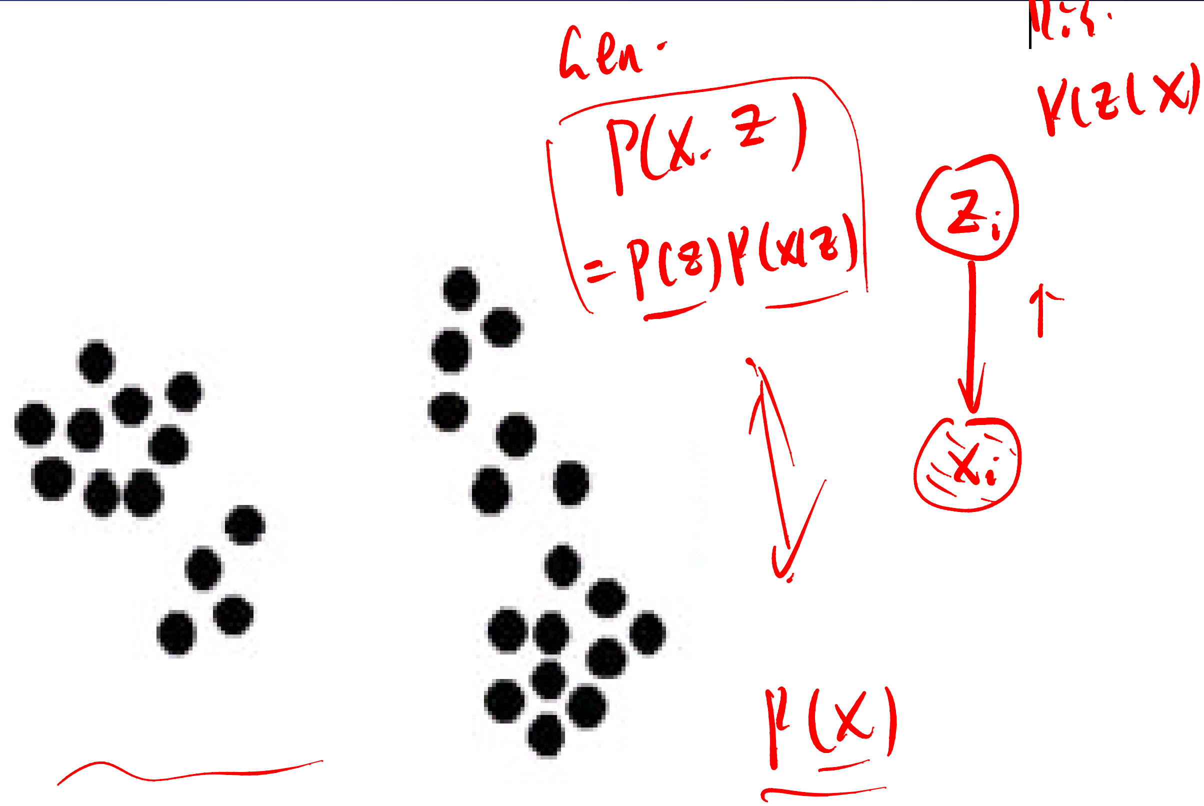


- Some seeds can result in poor convergence rate, or convergence to sub-optimal clusterings.
  - Select good seeds using a heuristic (e.g., doc least similar to any existing mean)
  - Try out multiple starting points (very important!!!)
  - Initialize with the results of another method.

How Many Clusters?

- Number of clusters K is given
  - Partition n docs into predetermined number of clusters
- Finding the "right" number of clusters is part of the problem
  - Given objs, partition into an "appropriate" number of subsets.
  - E.g., for query results - ideal value of K not known up front - though UI may impose limits.
- Solve an optimization problem: penalize having lots of clusters
  - application dependent, e.g., compressed summary of search results list.
  - Information theoretic approaches: model-based approach
- Tradeoff between having more clusters (better focus within each cluster) and having too many clusters
- Nonparametric Bayesian Inference

Clustering and partially observable probabilistic models



Unobserved Variables

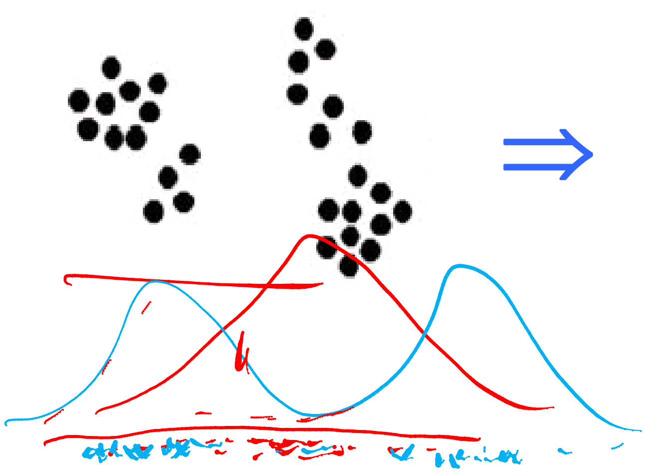
- A variable can be unobserved (latent) because:
  - it is an imaginary quantity meant to provide some simplified and abstractive view of the data generation process
    - e.g., speech recognition models, mixture models …
  - it is a real-world object and/or phenomena, but difficult or impossible to measure
    - e.g., the temperature of a star, causes of a disease, evolutionary ancestors …
  - it is a real-world object and/or phenomena, but sometimes wasn't measured, because of faulty sensors; or was measure with a noisy channel, etc.
    - e.g., traffic radio, aircraft signal on a radar screen,

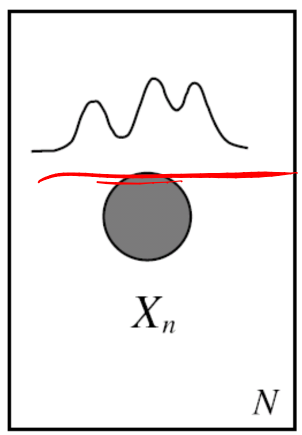
- Discrete latent variables can be used to partition/cluster data into sub-groups (mixture models, forthcoming).

- Continuous latent variables (factors) can be used for dimensionality reduction (factor analysis, etc., later lectures).

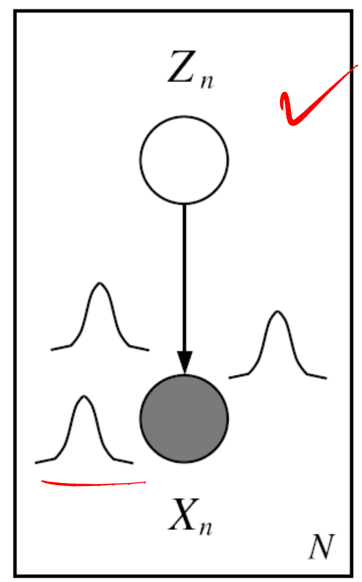
Mixture Models

- A density model $p(x)$ may be multi-modal.
- We may be able to model it as a mixture of uni-modal distributions (e.g., Gaussians).
- Each mode may correspond to a different sub-population (e.g., male and female).





(a)



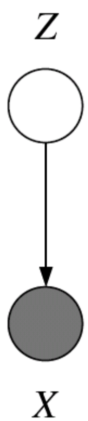
(b)

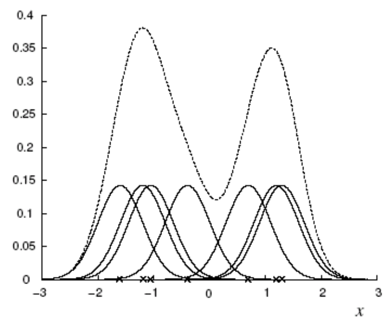
Gaussian Mixture Models (GMMs)

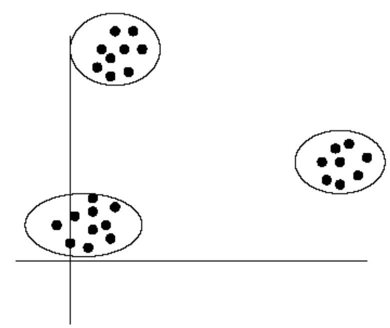
- Consider a mixture of *K* Gaussian components:

$$p(x_n|\mu,\Sigma)=\sum_k \pi_k N(x_n|\mu_k,\Sigma_k)$$

mixture proportion   mixture component





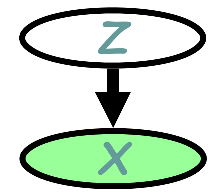


- This model can be used for unsupervised clustering.
  - This model (fit by AutoClass) has been used to discover new kinds of stars in astronomical data, etc.

GGM derivations

p(x)

- Consider a mixture of *K* Gaussian components:
  - *Z* is a latent class indicator vector:



$$p(z_n) = \text{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - *X* is a conditional Gaussian variable with a class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)\right\}$$

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

  - The likelihood of a sample:

mixture component

mixture proportion

$$p(x_n | \mu, \Sigma) = \sum_k p(z^k = 1 | \pi)\, p(x, | z^k = 1, \mu, \Sigma)$$
$$= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k} \right) = \sum_k \pi_k N(x, | \mu_k, \Sigma_k)$$

Learning mixture models

$x_i \rightarrow z_i \quad P(z_i | x_i)$

$\begin{cases} \mu_k \\ \Sigma_k \quad \forall k. \\ \pi_k \end{cases}$

obj: $\checkmark P(x)$

$$P(z|x) = \frac{P(x, z)}{P(x)} \begin{matrix} \checkmark \\ \checkmark \end{matrix}$$

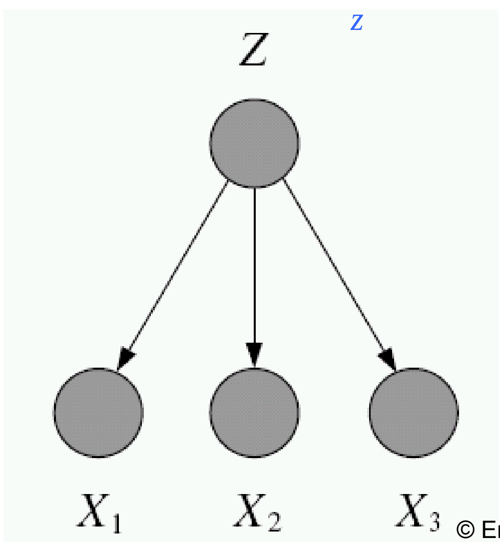
Why is Learning Harder?

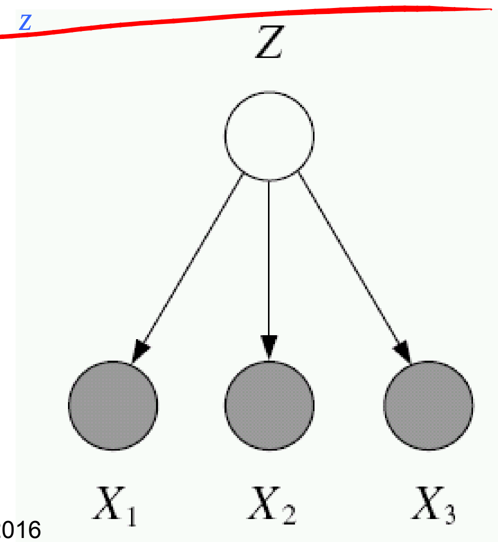
- In fully observed iid settings, the log likelihood decomposes into a sum of local terms.

$$\ell_c(\theta; D) = \log p(x, z \mid \theta) = \log p(z \mid \theta_z) + \log p(x \mid z, \theta_x)$$

- With latent variables, all the parameters become coupled together via *marginalization*

$$\ell_c(\theta; D) = \log \sum_z p(x, z \mid \theta) = \log \sum_z p(z \mid \theta_z) p(x \mid z, \theta_x)$$





Gradient Learning for mixture models

- We can learn mixture densities using gradient descent on the log likelihood. The gradients are quite interesting:

$$\ell(\theta) = \log p(\mathrm{x} \mid \theta) = \log \sum_k \pi_k p_k(\mathrm{x} | \theta_k)$$

$$\frac{\partial \ell}{\partial \theta} = \frac{1}{p(\mathrm{x} \mid \theta)} \sum_k \pi_k \frac{\partial p_k(\mathrm{x} | \theta_k)}{\partial \theta}$$

$$= \sum_k \frac{\pi_k}{p(\mathrm{x} \mid \theta)} p_k(\mathrm{x} | \theta_k) \frac{\partial \log p_k(\mathrm{x} | \theta_k)}{\partial \theta}$$

$$= \sum_k \pi_k \frac{p_k(\mathrm{x} | \theta_k)}{p(\mathrm{x} \mid \theta)} \frac{\partial \log p_k(\mathrm{x} | \theta_k)}{\partial \theta_k} = \sum_k r_k \frac{\partial \ell_k}{\partial \theta_k}$$

- In other words, the gradient is the responsibility weighted sum of the individual log likelihood gradients.
- Can pass this to a conjugate gradient routine.

Parameter Constraints

- Often we have constraints on the parameters, e.g. $\Sigma_k \pi_k = 1$, Σ being symmetric positive definite (hence $\Sigma_{ii} > 0$).
- We can use constrained optimization, or we can reparameterize in terms of unconstrained values.
  - For normalized weights, use the softmax transform:
  - For covariance matrices, use the Cholesky decomposition:

$$\Sigma^{-1} = \mathbf{A}^T \mathbf{A}$$

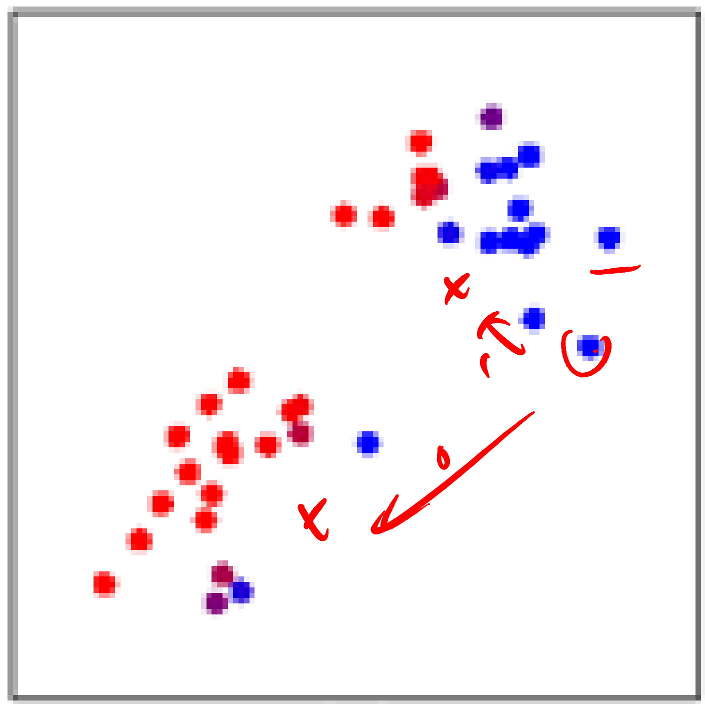
where A is upper diagonal with positive diagonal:

$$\mathbf{A}_{ii} = \exp(\lambda_i) > 0 \quad \mathbf{A}_{ij} = \eta_{ij} \ \ (j > i) \quad \mathbf{A}_{ij} = 0 \ \ (j < i)$$

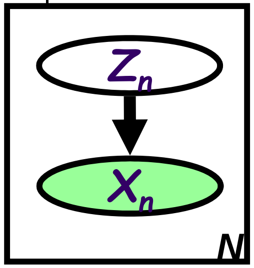
the parameters $\gamma_i, \lambda_i, \eta_{ij} \in \mathbb{R}$ are unconstrained.

  - Use chain rule to compute $\frac{\partial \ell}{\partial \pi}, \frac{\partial \ell}{\partial \mathbf{A}}$.

The Expectation-Maximization (EM) Algorithm



EM algorithm for GMM



- E.g., A mixture of K Gaussians:
  - Z is a latent class indicator vector

$$p(z_n) = \text{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

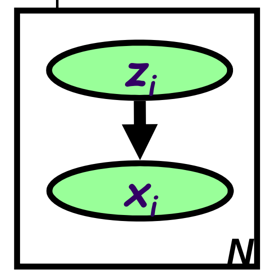
  - X is a conditional Gaussian variable with a class-specific mean/covariance

$$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2} |\Sigma_k|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)\right\}$$

  - The likelihood of a sample:

$$\begin{aligned} p(x_n|\mu,\Sigma) &= \sum_k p(z^k = 1 \mid \pi)\, p(x, \mid z^k = 1, \mu, \Sigma) \\ &= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k}\right) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k) \end{aligned}$$

EM algorithm for GMM



- **Recall MLE for completely observed data**

- **Data log-likelihood**

$$\ell(\theta; D) = \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n \mid \pi) p(x_n \mid z_n, \mu, \sigma)$$

$$= \sum_n \log \prod_k \pi_k^{z_n^k} + \sum_n \log \prod_k N(x_n; \mu_k, \sigma)^{z_n^k}$$

$$= \sum_n \sum_k z_n^k \log \pi_k - \sum_n \sum_k z_n^k \frac{1}{2\sigma^2} (x_n - \mu_k)^2 + C$$

- **MLE**

$$\hat{\pi}_{k,MLE} = \arg\max_\pi \ell(\theta; D),$$

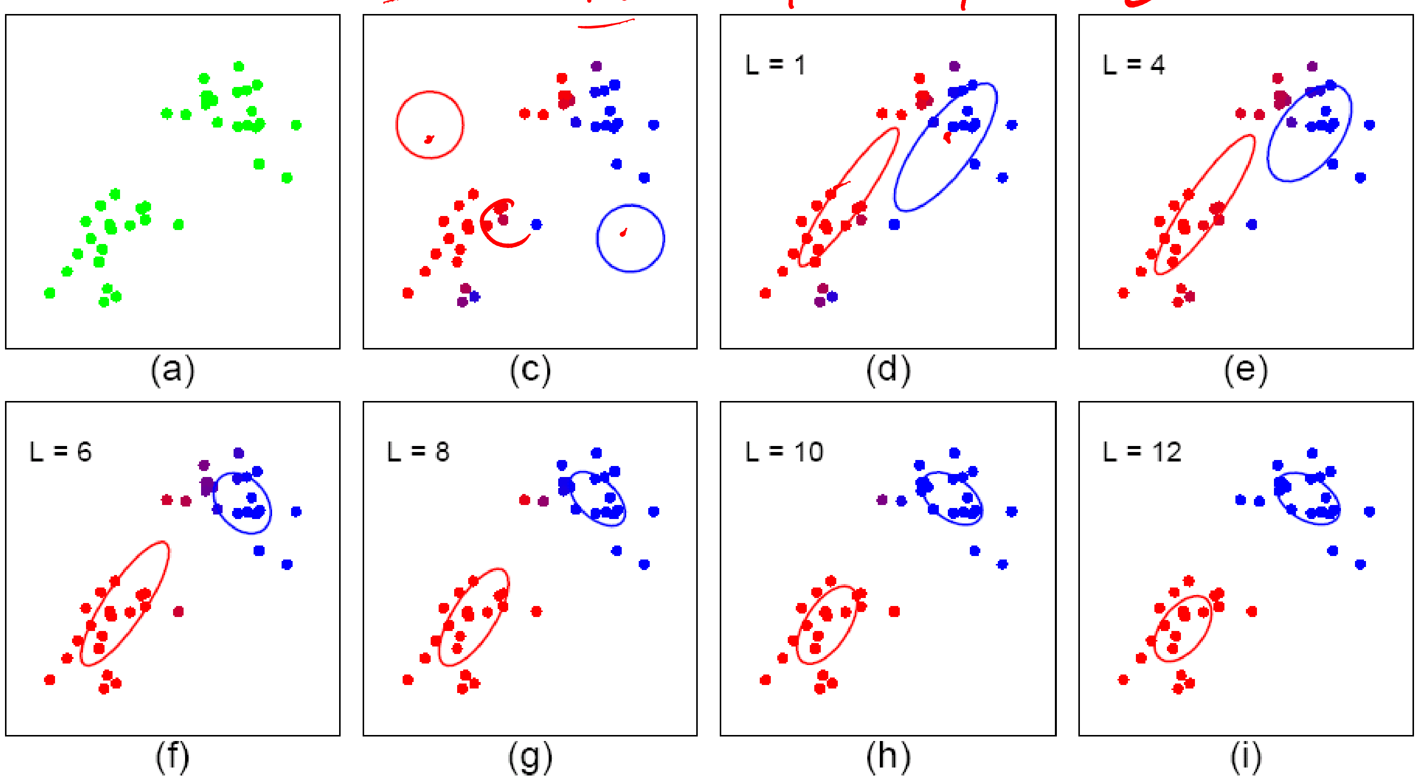
$$\hat{\mu}_{k,MLE} = \arg\max_\mu \ell(\theta; D) \quad \Rightarrow \quad \hat{\mu}_{k,MLE} = \frac{\sum_n z_n^k x_n}{\sum_n z_n^k}$$

$$\hat{\sigma}_{k,MLE} = \arg\max_\sigma \ell(\theta; D)$$

- **What if we do not know z_n?** $\quad z_n \rightarrow p(z_n^k = 1 \mid x, \mu^{(t)}, \Sigma^{(t)})$

EM algorithm for GMM

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop



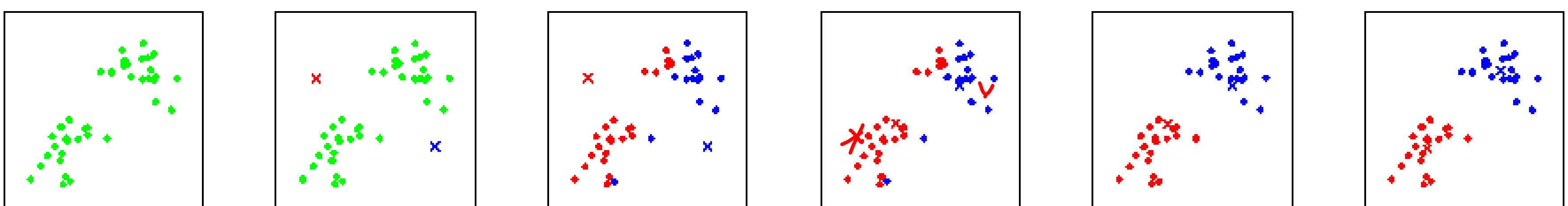
Comparing to K-means

- Start:
  - "Guess" the centroid μ_k and coveriance Σ_k of each of the K clusters
- Loop
  - For each point n=1 to N,
    compute its cluster label:

$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)})$$

  - For each cluster k=1:K

$$\mu_k^{(t+1)} = \frac{\sum_n \delta(z_n^{(t)}, k) x_n}{\sum_n \delta(z_n^{(t)}, k)} \qquad \Sigma_k^{(t+1)} = \ldots$$



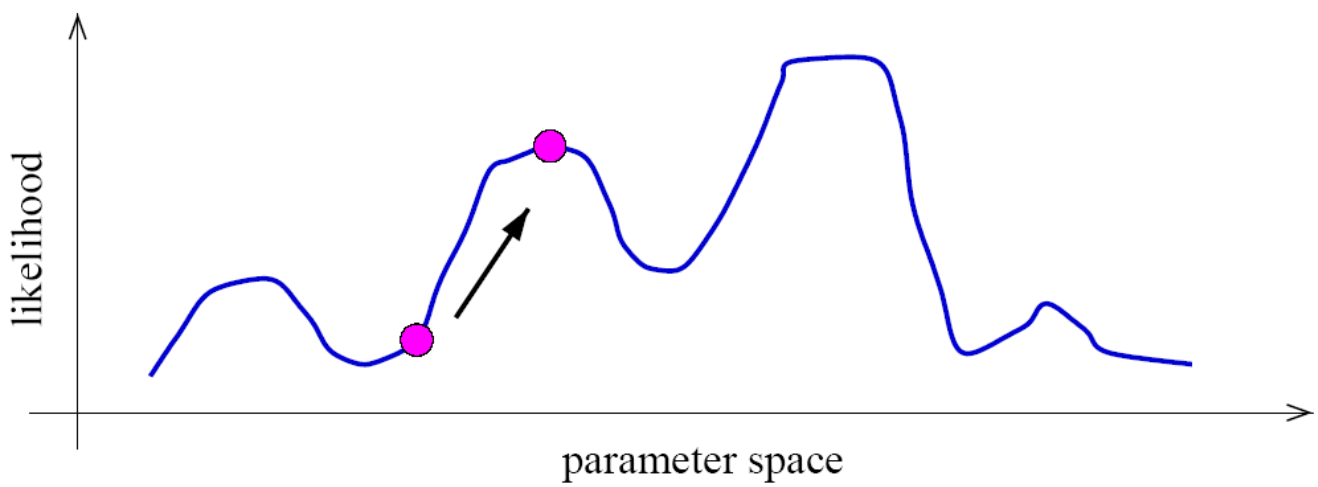
(a) (b) (c) (d) (e) (f)

Notes on EM Algorithm

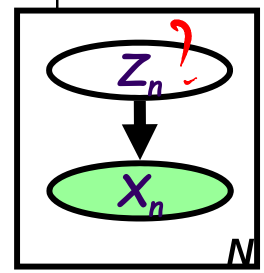
- EM is an optimization strategy for objective functions that can be interpreted as likelihoods in the presence of missing data.
- It is much simpler than gradient methods:
  - No need to choose step size.
  - Enforces constraints automatically.
  - Calls inference and fully observed learning as subroutines.
- EM is an Iterative algorithm with two linked steps:
  - E-step: fill-in hidden values using inference, $p(z|x, \theta^t)$.
  - M-step: update parameters t+1 using standard MLE/MAP method applied to completed data
- We will prove that this procedure monotonically improves (or leaves it unchanged). Thus it always converges to a local optimum of the likelihood.

Identifiability

- A mixture model induces a multi-modal likelihood.
- Hence gradient ascent can only find a local maximum.
- Mixture models are unidentifiable, since we can always switch the hidden labels without affecting the likelihood.
- Hence we should be careful in trying to interpret the "meaning" of latent variables.



How is EM derived?



- A mixture of K Gaussians:
  - Z is a latent class indicator vector

    $$p(z_n) = \text{multi}(z_n : \pi) = \prod_k (\pi_k)^{z_n^k}$$

  - X is a conditional Gaussian variable with a class-specific mean/covariance

    $$p(x_n \mid z_n^k = 1, \mu, \Sigma) = \frac{1}{(2\pi)^{m/2}|\Sigma_k|^{1/2}} \exp\left\{-\tfrac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1}(x_n - \mu_k)\right\}$$

  - The likelihood of a sample:

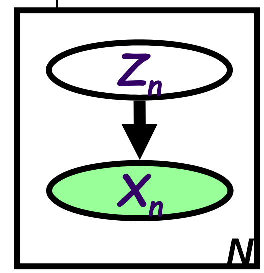
    $$p(x_n|\mu,\Sigma) = \sum_k p(z_n^k = 1 \mid \pi)\, p(x, \mid z_n^k = 1, \mu, \Sigma)$$
    $$= \sum_{z_n} \prod_k \left((\pi_k)^{z_n^k} N(x_n : \mu_k, \Sigma_k)^{z_n^k}\right) = \sum_k \pi_k N(x, \mid \mu_k, \Sigma_k)$$

- The "complete" likelihood

  $$p(x_n, z_n^k = 1|\mu,\Sigma) = p(z_n^k = 1 \mid \pi)\, p(x, \mid z_n^k = 1, \mu, \Sigma) = \pi_k N(x, \mid \mu_k, \Sigma_k)$$
  $$p(x_n, z_n|\mu,\Sigma) = \prod_k \left[\pi_k N(x, \mid \mu_k, \Sigma_k)\right]^{z_n^k}$$

But this is itself a random variable! Not good as objective function

How is EM derived?



- The complete log likelihood:

$$\ell(\theta; D) = \log \prod_n p(z_n, x_n) = \log \prod_n p(z_n \mid \pi) p(x_n \mid z_n, \mu, \sigma)$$

$$= \sum_n \log \prod_k \pi_k^{z_n^k} + \sum_n \log \prod_k N(x_n; \mu_k, \sigma)^{z_n^k}$$

$$= \sum_n \sum_k z_n^k \log \pi_k - \sum_n \sum_k z_n^k \frac{1}{2\sigma^2}(x_n - \mu_k)^2 + C$$

$q(z) = p(z|x)$

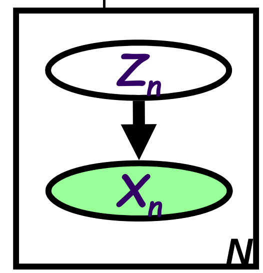
- The expected complete log likelihood

$$\langle \ell_c(\theta; x, z) \rangle = \sum_n \langle \log p(z_n \mid \pi) \rangle_{p(z|x)} + \sum_n \langle \log p(x_n \mid z_n, \mu, \Sigma) \rangle_{p(z|x)}$$

$$= \sum_n \sum_k \langle z_n^k \rangle \log \pi_k - \frac{1}{2} \sum_n \sum_k \langle z_n^k \rangle \left((x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) + \log |\Sigma_k| + C \right)$$

E-step

- We maximize $\langle l_c(\theta) \rangle$ iteratively using the following iterative procedure:

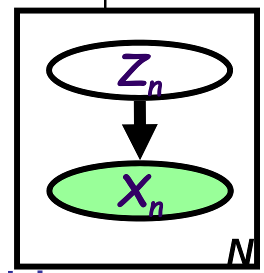


  – Expectation step: computing the expected value of the sufficient statistics of the hidden variables (i.e., z) given current est. of the parameters (i.e., π and μ).

$$\tau_n^{k(t)} = \langle z_n^k \rangle_{q^{(t)}} = p(z_n^k = 1 \mid x, \mu^{(t)}, \Sigma^{(t)}) = \frac{\pi_k^{(t)} N(x_n, \mid \mu_k^{(t)}, \Sigma_k^{(t)})}{\sum_i \pi_i^{(t)} N(x_n, \mid \mu_i^{(t)}, \Sigma_i^{(t)})}$$

    - Here we are essentially doing **inference**

M-step



- We maximize $\langle l_c(\boldsymbol{\theta}) \rangle$ iteratively using the following iterative procedure:

  – Maximization step: compute the parameters under current results of the expected value of the hidden variables

$$\pi_k^* = \arg\max \langle l_c(\boldsymbol{\theta}) \rangle, \qquad \Rightarrow \frac{\partial}{\partial \pi_k} \langle l_c(\boldsymbol{\theta}) \rangle = 0, \forall k, \quad \text{s.t. } \sum_k \pi_k = 1$$

$$\Rightarrow \pi_k^* = \sum_n \langle z_n^k \rangle_{q^{(t)}} \Big/ N = \sum_n \tau_n^{k(t)} \Big/ N = \langle n_k \rangle \Big/ N$$

$$\mu_k^* = \arg\max \langle l(\boldsymbol{\theta}) \rangle, \qquad \Rightarrow \mu_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} x_n}{\sum_n \tau_n^{k(t)}}$$

$$\Sigma_k^* = \arg\max \langle l(\boldsymbol{\theta}) \rangle, \qquad \Rightarrow \Sigma_k^{(t+1)} = \frac{\sum_n \tau_n^{k(t)} (x_n - \mu_k^{(t+1)})(x_n - \mu_k^{(t+1)})^T}{\sum_n \tau_n^{k(t)}}$$

Fact :

$$\frac{\partial \log |\mathrm{A}^{-1}|}{\partial \mathrm{A}^{-1}} = \mathrm{A}^T$$

$$\frac{\partial \mathbf{x}^T \mathrm{A} \mathbf{x}}{\partial \mathrm{A}} = \mathbf{x}\mathbf{x}^T$$

- This is isomorphic to **MLE** except that the variables that are hidden are replaced by their expectations (in general they will by replaced by their corresponding "**sufficient statistics**")

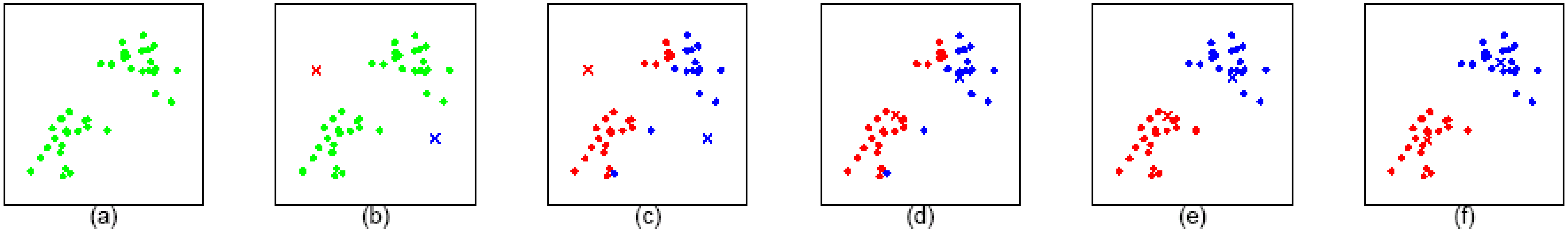
Compare: K-means

- The EM algorithm for mixtures of Gaussians is like a "soft version" of the K-means algorithm.
- In the K-means "E-step" we do hard assignment:

$$z_n^{(t)} = \arg\max_k (x_n - \mu_k^{(t)})^T \Sigma_k^{-1(t)} (x_n - \mu_k^{(t)})$$

- In the K-means "M-step" we update the means as the weighted sum of the data, but now the weights are 0 or 1:

$$\mu_k^{(t+1)} = \frac{\sum_n \delta(z_n^{(t)}, k) x_n}{\sum_n \delta(z_n^{(t)}, k)}$$



Theory underlying EM

- What are we doing?

- Recall that according to MLE, we intend to learn the model parameter that would have maximize the likelihood of the data.

- But we do not observe z, so computing

$$\ell_c(\theta; D) = \log \sum_z p(x, z \mid \theta) = \log \sum_z p(z \mid \theta_z) p(x \mid z, \theta_x)$$

is difficult!

- What shall we do?

Complete & Incomplete Log Likelihoods

- Complete log likelihood

  Let X denote the observable variable(s), and Z denote the latent variable(s). If Z could be observed, then

  $$\ell_c(\theta; x, z) \overset{\text{def}}{=} \log p(x, z \mid \theta)$$

  - Usually, optimizing $\ell_c()$ given both z and x is straightforward (c.f. MLE for fully observed models).
  - Recalled that in this case the objective for, e.g., MLE, decomposes into a sum of factors, the parameter for each factor can be estimated separately.
  - **But given that Z is not observed, $\ell_c()$ is a random quantity, cannot be maximized directly**.

- Incomplete log likelihood

  With z unobserved, our objective becomes the log of a marginal probability:

  $$\ell_c(\theta; x) = \log p(x \mid \theta) = \log \sum_z p(x, z \mid \theta)$$

  - **This objective won't decouple**

Expected Complete Log Likelihood

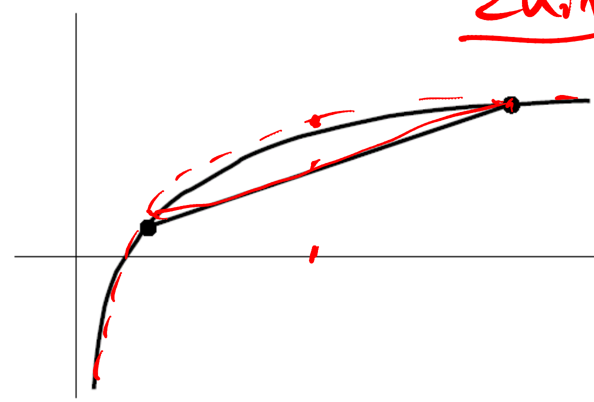
- For ***any*** distribution $q(z)$, define *expected complete log likelihood*:

$$\langle \ell_c(\theta; x, z)\rangle_q \overset{\text{def}}{=} \sum_z q(z \mid x, \theta) \log p(x, z \mid \theta)$$

  - A deterministic function of θ
  - Linear in $\ell_c()$ --- inherit its factorizabiility
  - Does maximizing this surrogate yield a maximizer of the likelihood?

- Jensen's inequality

$$\begin{aligned}
\ell(\theta; x) &= \log p(x \mid \theta) \\
&= \log \sum_z p(x, z \mid \theta) \\
&= \log \sum_z q(z \mid x) \frac{p(x, z \mid \theta)}{q(z \mid x)} \\
&\geq \sum_z q(z \mid x) \log \frac{p(x, z \mid \theta)}{q(z \mid x)}
\end{aligned}$$



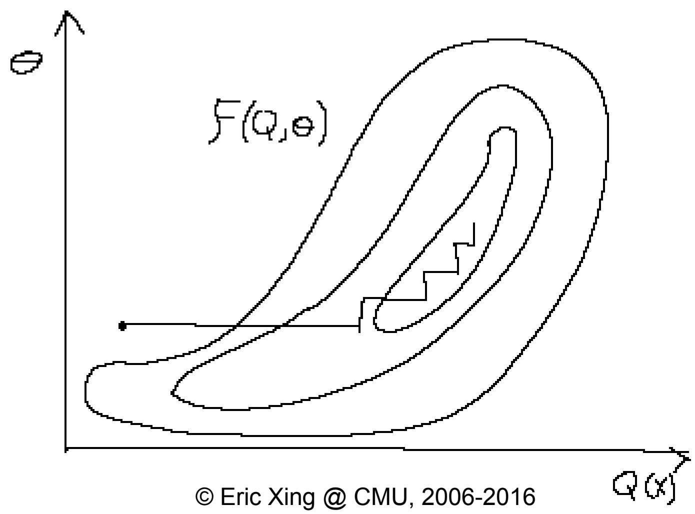
$$\Rightarrow \quad \ell(\theta; x) \geq \langle \ell_c(\theta; x, z)\rangle_q + H_q$$

Lower Bounds and Free Energy

- For fixed data x, define a functional called the free energy:

$$F(q,\theta) \stackrel{\text{def}}{=} \sum_{z} q(z\,|\,x) \log \frac{p(x,z\,|\,\theta)}{q(z\,|\,x)} \leq \ell(\theta;x)$$

- The EM algorithm is coordinate-ascent on F :
  - **E-step:** $q^{t+1} = \arg\max_{q} F(q,\theta^{t})$
  - **M-step:** $\theta^{t+1} = \arg\max_{\theta} F(q^{t+1},\theta^{t})$



E-step: maximization of expected ℓ_c w.r.t. q

- Claim: $$q^{t+1} = \arg\max_q F(q, \theta^t) = p(z \mid x, \theta^t)$$
  - This is the posterior distribution over the latent variables given the data and the parameters. Often we need this at test time anyway (e.g. to perform classification).
- Proof (easy): this setting attains the bound $\ell(\theta;x) \geq F(q,\theta)$

$$F(p(z|x,\theta^t),\theta^t) = \sum_z p(z|x,\theta^t)\log\frac{p(x,z\mid\theta^t)}{p(z|x,\theta^t)}$$
$$= \sum_z p(z|x,\theta^t)\log p(x\mid\theta^t)$$
$$= \log p(x\mid\theta^t) = \ell(\theta^t;x)$$

- Can also show this result using variational calculus or the fact that $\ell(\theta;x) - F(q,\theta) = \mathrm{KL}\big(q \,\|\, p(z \mid x,\theta)\big)$

E-step ≡ plug in posterior expectation of latent variables

- Without loss of generality: assume that $p(x,z|\theta)$ is a generalized exponential family distribution:

$$p(x,z|\theta)=\frac{1}{Z(\theta)}h(x,z)\exp\left\{\sum_i \theta_i f_i(x,z)\right\}$$

  - Special cases: if $p(X|Z)$ are GLIMs, then $f_i(x,z)=\eta_i^T(z)\xi_i(x)$

- The expected complete log likelihood under $q^{t+1}=p(z|x,\theta^t)$ is

$$\left\langle \ell_c(\theta^t;x,z)\right\rangle_{q^{t+1}}=\sum_z q(z|x,\theta^t)\log p(x,z|\theta^t)-A(\theta)$$

$$=\sum_i \theta_i^t \left\langle f_i(x,z)\right\rangle_{q(z|x,\theta^t)}-A(\theta)$$

$$\overset{p\sim\text{GLIM}}{=}\sum_i \theta_i^t \left\langle \eta_i(z)\right\rangle_{q(z|x,\theta^t)}\xi_i(x)-A(\theta)$$

M-step: maximization of expected ℓ_c w.r.t. θ

- Note that the free energy breaks into two terms:

$$
\begin{aligned}
F(q,\theta) &= \sum_z q(z \mid x) \log \frac{p(x,z \mid \theta)}{q(z \mid x)} \\
&= \sum_z q(z \mid x) \log p(x,z \mid \theta) - \sum_z q(z \mid x) \log q(z \mid x) \\
&= \langle \ell_c(\theta; x, z) \rangle_q + H_q
\end{aligned}
$$

  - The first term is the expected complete log likelihood (energy) and the second term, which does not depend on θ, is the entropy.

- Thus, in the M-step, maximizing with respect to θ for fixed q we only need to consider the first term:

$$
\theta^{t+1} = \arg\max_\theta \langle \ell_c(\theta; x, z) \rangle_{q^{t+1}} = \arg\max_\theta \sum_z q(z \mid x) \log p(x,z \mid \theta)
$$

  - Under optimal q^{t+1}, this is equivalent to solving a standard MLE of fully observed model $p(x,z|\theta)$, with the sufficient statistics involving z replaced by their expectations w.r.t. $p(z|x,\theta)$.

Summary: EM Algorithm

- A way of maximizing likelihood function for latent variable models. Finds MLE of parameters when the original (hard) problem can be broken up into two (easy) pieces:
  1. Estimate some "missing" or "unobserved" data from observed data and current parameters.
  2. Using this "complete" data, find the maximum likelihood parameter estimates.

- Alternate between filling in the latent variables using the best guess (posterior) and updating the parameters based on this guess:
  - E-step: $q^{t+1} = \arg\max_q F(q, \theta^t)$
  - M-step: $\theta^{t+1} = \arg\max_\theta F(q^{t+1}, \theta^t)$

- In the M-step we optimize a lower bound on the likelihood. In the E-step we close the gap, making bound=likelihood.

EM Variants

- Sparse EM:

  Do not re-compute exactly the posterior probability on each data point under all models, because it is almost zero. Instead keep an “active list” which you update every once in a while.

- Generalized (Incomplete) EM:

  It might be hard to find the ML parameters in the M-step, even given the completed data. We can still make progress by doing an M-step that improves the likelihood a bit (e.g. gradient step). Recall the IRLS step in the mixture of experts model.

A Report Card for EM

- Some good things about EM:
  - no learning rate (step-size) parameter
  - automatically enforces parameter constraints
  - very fast for low dimensions
  - each iteration guaranteed to improve likelihood

- Some bad things about EM:
  - can get stuck in local minima
  - can be slower than conjugate gradient (especially near convergence)
  - requires expensive inference step
  - is a maximum likelihood/MAP method