

```
    for object to mirror  
    mirror_mod.mirror_object =  
    operation == "MIRROR_X":  
    mirror_mod.use_x = True  
    mirror_mod.use_y = False  
    mirror_mod.use_z = False  
    operation == "MIRROR_Y":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = True  
    mirror_mod.use_z = False  
    operation == "MIRROR_Z":  
    mirror_mod.use_x = False  
    mirror_mod.use_y = False  
    mirror_mod.use_z = True
```

```
    selection at the end -add  
    mirror_ob.select= 1  
    mirror_ob.select= 1  
    context.scene.objects.active  
    ("Selected" + str(modifier))  
    mirror_ob.select = 0  
    = bpy.context.selected_objects  
    data.objects[one.name].select  
    print("please select exactly
```

```
-- OPERATOR CLASSES ----
```

```
types.Operator):  
    X mirror to the selected  
    object.mirror_mirror_x"  
    mirror X"
```

EM Algorithm

Jiatong

Unsupervised Learning

No labels

- Can't do classification anymore!

We still can have a notion of groups

Task: divide things into piles of similar things

Classification found patterns that explained a label

- We can find patterns that separate the data

Clustering

Sort the data into clusters (groups)

Examples that are in the same group are similar

- Should be more similar to one another than to items not in the cluster
- Ideally clusters correspond to (unknown) class labels

We don't know what we will get (unsupervised!)

- What does it mean for two examples to be similar?

Solving Clustering

How do we group examples into clusters?

Same as before!

- Design a model
- Define a model objective to represent learning goal
- Write procedure for maximizing objective
- Compute model parameters using procedure

Defining Clusters

- A cluster is a group of similar examples
- Define cluster k by a prototype μ_k
- value of 1 means example n in cluster k, $r_{nk} \in \{0,1\}$
- Mean of the examples in cluster k
- $$\mu_k = \frac{1}{\sum_{n=1}^N r_{nk}} \sum_{n=1}^N r_{nk} x_n$$

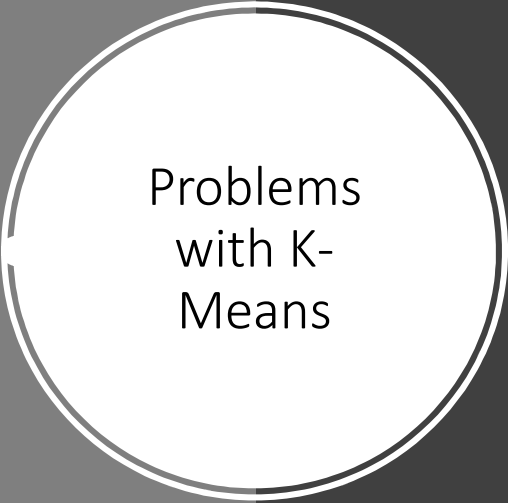


Algorithm: K-Means

- Given Data
- Initialize μ_k
- Iteratively update until convergence:

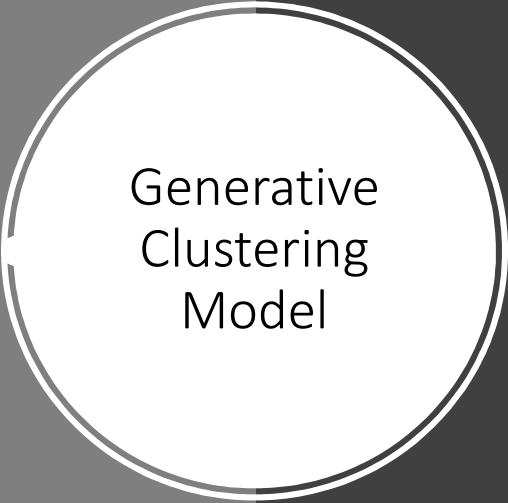
$$r_{nk} = \begin{cases} 1 & \text{if } k = \operatorname{argmin}_j \|x_n - \mu_j\|^2 \\ 0 & \text{otherwise} \end{cases}$$

$$\mu_k = \frac{\sum_n r_{nk} x_n}{\sum_n r_{nk}}$$



Problems with K- Means

- Hard assignment
 - Examples go from one cluster to the other right away
- A smooth transition might be better
- How do we do smooth transitions?
 - Probabilities



Generative Clustering Model

- Assume we have K clusters
- Each cluster represented by a multi-variate Gaussian
- Generative process:
 - Select a cluster (a Gaussian distribution)
 - Generate an example by sampling from the Gaussian

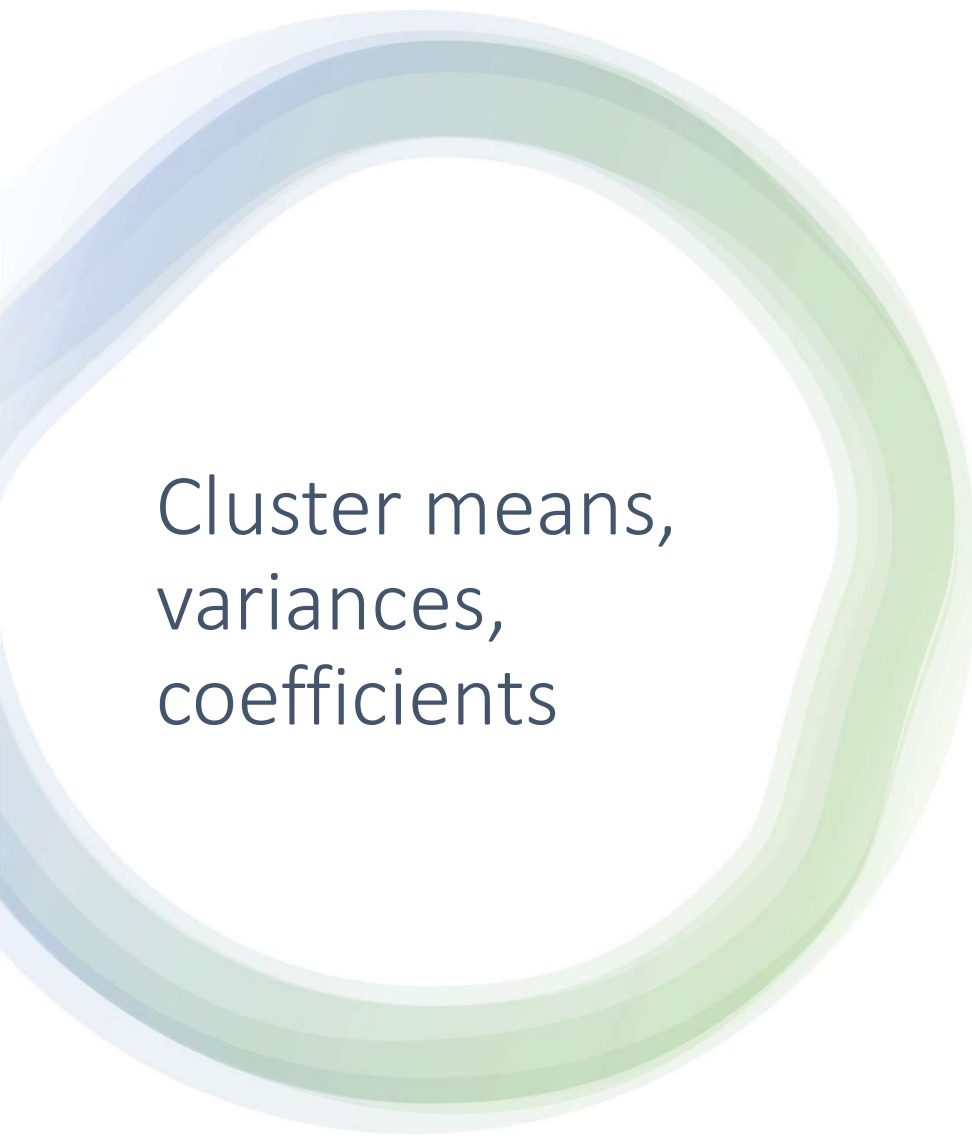


Gaussian Mixtures

- Since we have multiple Gaussians generating points, we call the model Gaussian Mixture Model
- Why Gaussians?
 - Captures intuition about clusters
 - ● Examples are more likely to be near center of cluster

Gaussian Mixture Model

Derivation



Cluster means,
variances,
coefficients

- $N_k = \sum \gamma(z_{nk})$
- $\pi_k = \frac{N_k}{N} = \frac{N_k}{\sum N_{k'}}$
- $\mu_k = \frac{1}{N_k} \sum \gamma(z_{nk}) x_n$
- $\Sigma_k = \frac{1}{N_k} \sum \gamma(z_{nk}) (x_n - \mu_k)(x_n - \mu_k)^T$



Cluster Responsibilities

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum^K \pi_k N(x_n | \mu_k, \Sigma_k)}$$

Algorithm

- Given Data
- Initialize μ σ and π
- Iteratively update until convergence
 - $\gamma(z_{nk})$
 - $\mu_k \Sigma_k \pi_k$



Problems with GMMs

- Mode collapse
 - When a single data point is isolated and becomes its own Gaussian
 - Watch for this and reset the Gaussian
- Very non-convex likelihood
 - $K!$ equivalent best solutions
 - Can find any one but may end up in a local minimum
- Requires more iterations to converge than K-Means
 - Plus, each iteration is more computationally expensive



Similarities

	KMEANS	GAUSSIAN MIXTURES
Assign an examples to clusters	r_{nk}	$\gamma(z_{nk})$
Compute new model	μ_k	$\mu_k \sum_k \pi_k$

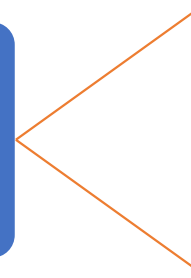
Same Algorithm

The maximization
algorithm for both
models is the same

Iterate two steps

Compute the
expected cluster
assignments

Maximize the
model parameter

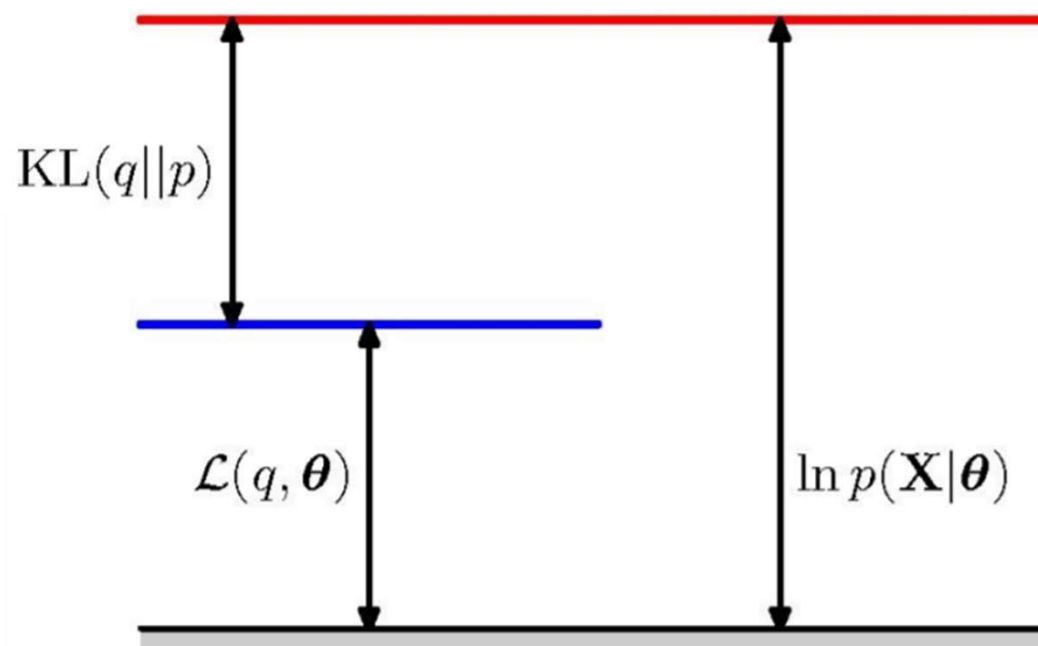




General EM ALgorithm

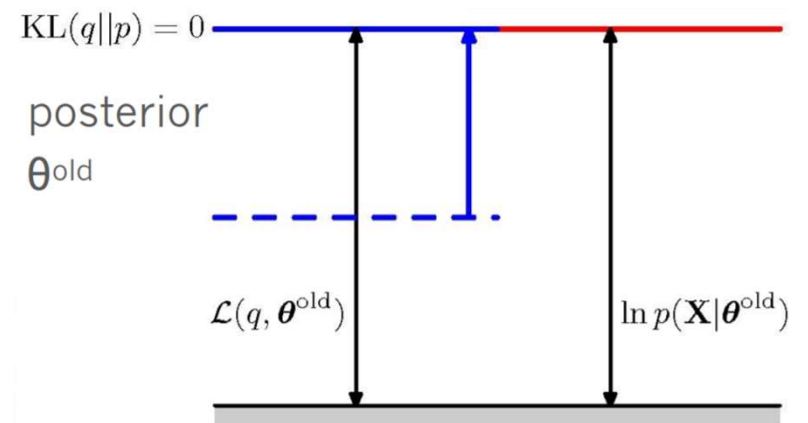
Pictorial View

- We decompose the likelihood as
- $\log p(\mathbf{X}|\theta) = \mathcal{L}(q, \theta) + \text{KL}(q||p)$



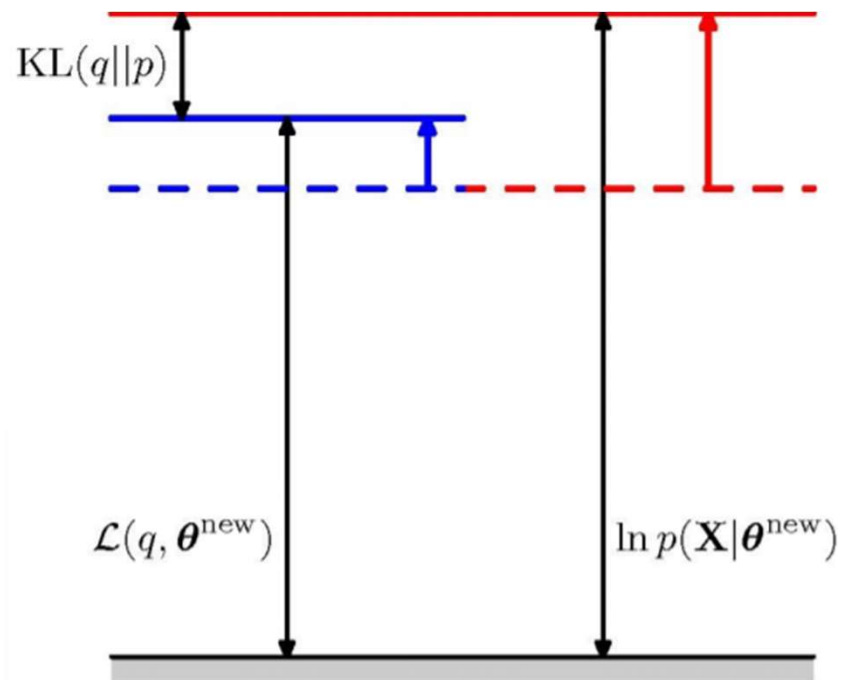
Pictorial View

- E step:
- $\log p(X|\theta) = L(q, \theta) + KL(q||p)$
goes to 0
- $\log p(X|\theta) = L(q, \theta) + 0$
- The observed variable Z match what we expect given parameters



Pictorial View

- M-Step
- Maximize $L(q, \theta)$ by finding new θ for fixed $q(Z)$
- $\log p(X|\theta) = L(q, \theta) + KL(q||p)$
- increase can only increase
- $\log p(X|\theta) \geq \log p(X|\theta^{old})$
- The New parameters best explain the “observed” variable Z





Convergence

- When will $\log p(X|\theta) = \log p(X|\theta^{old})$
- When we can no longer increase the likelihood
- Since the likelihood always increases, this must be a maximum (possibly local)



Convergence

- We now see why EM converges in general
 - We are always increasing the likelihood function
 - At some point we won't be able to increase it any more
- Very powerful result
 - For any problem with latent variables, if you can write the complete data likelihood, you can use EM
 - The algorithm will always converge!



Further Examine EM



The General EM Algorithm

- Goal: maximize a likelihood function $p(X|\theta)$
 - Write a joint distribution over the complete data $p(X, Z|\theta)$
- Choose an initial setting for θ^{old}
- E Step: Compute the $q(Z)$ as $p(Z|X, \theta^{old})$
- M step: Compute θ^{new} given by $\theta^{new} = \operatorname{argmax}_{\theta}(Q(\theta, \theta^{old}))$
 - $Q(\theta, \theta^{old}) = \sum_z p(Z|X, \theta^{old}) \log p(X, Z|\theta)$
- Let $\theta^{old} = \theta^{new}$
- Repeat until convergence



GMM with EM

EM is Everywhere



Remember the similar forms of GMM and K-means?

K-means is an application of EM in the limit
Force hard cluster assignments



See, EM really is everywhere

Google scholar: Dempster, et al. Maximum
likelihood from incomplete data via the EM
algorithm.

22083 citations

General EM



The EM form is the same, but each step can be more complicated



E step

Finding the values for the hidden variables may not be easy

We may need to approximate the values



Maximization may require multiple steps, optional constraints

Latent Variables



EM is useful for latent variables

Variables that you do not observe



What is the structure of these latent variables?

How do they influence the observed variables?

Can you have multiple latent variables in a complex structure?



We need some way to talk about these variables formally



Q&A