CSCI 567 Machine Learning (Spring 2018)

Michael Shindler

Lecture 16: March 19

Outline

Outline

Administrative Stuff

- It's now week 10.
- Friday April 6 is coming up. Quiz 2.
- Quiz 2:
  - Bring a pencil
  - Bring your USC ID.
  - Be sure to fill out the ID section on Scantron
  - Be sure to STOP when time called
    - Stop writing immediately.
    - Look up, not at your exam or desk.

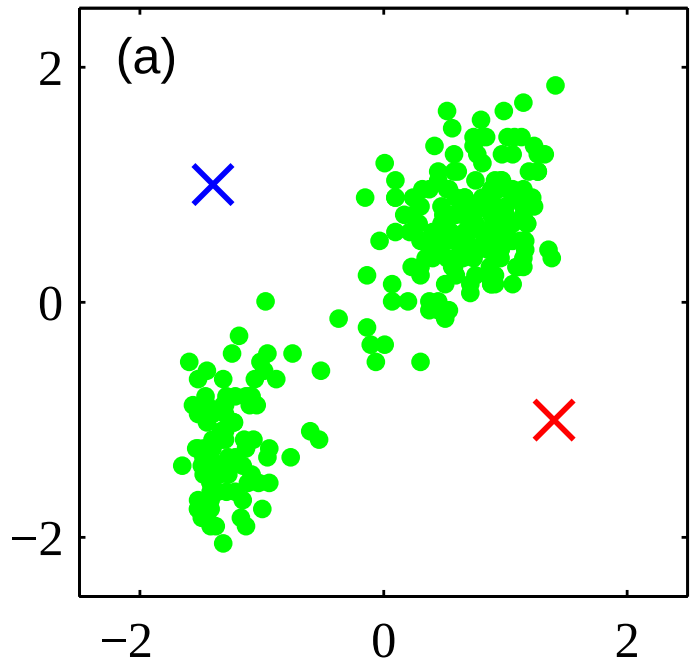
Outline

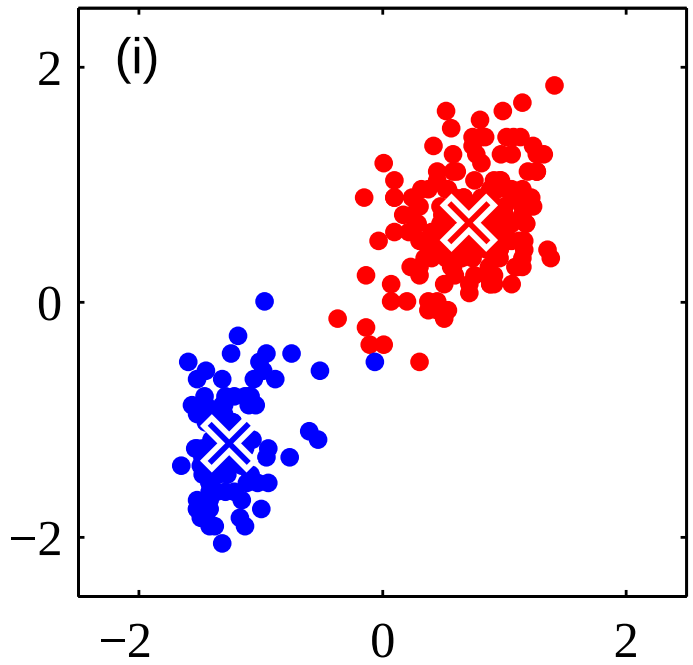
Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $\boldsymbol{x}_n$

Example Cluster data into two clusters.





Clustering

Setup Given $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$ and K, we want to output

- $\{\boldsymbol{\mu}_k\}_{k=1}^K$: prototypes of clusters
- $A(\boldsymbol{x}_n) \in \{1, 2, \ldots, K\}$: the cluster membership, i.e., the cluster ID assigned to $\boldsymbol{x}_n$

Key difference from *supervised learning problems*

Nobody tells us what the ground-truth is for any $\boldsymbol{x}_n$!

Algorithm: K-means clustering

Intuition Data points assigned to cluster k should be close to $\boldsymbol{\mu}_k$, the prototype.

Algorithm: K-means clustering

Intuition Data points assigned to cluster k should be close to $\boldsymbol{\mu}_k$, the prototype.

Distortion measure (clustering objective function, cost function)

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \|\boldsymbol{x}_n - \boldsymbol{\mu}_k\|_2^2$$

where $r_{nk} \in \{0, 1\}$ is an indicator variable

$$r_{nk} = 1 \quad \text{if and only if} \quad A(\boldsymbol{x}_n) = k$$

Algorithm

Minimize distortion measure alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values

Algorithm

Minimize distortion measure alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, minimize J over $\{r_{nk}\}$, which leads to the following cluster assignment rule

$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

Algorithm

Minimize distortion measure alternative optimization between $\{r_{nk}\}$ and $\{\boldsymbol{\mu}_k\}$

- **Step 0** Initialize $\{\boldsymbol{\mu}_k\}$ to some values
- **Step 1** Assume the current value of $\{\boldsymbol{\mu}_k\}$ fixed, minimize J over $\{r_{nk}\}$, which leads to the following cluster assignment rule

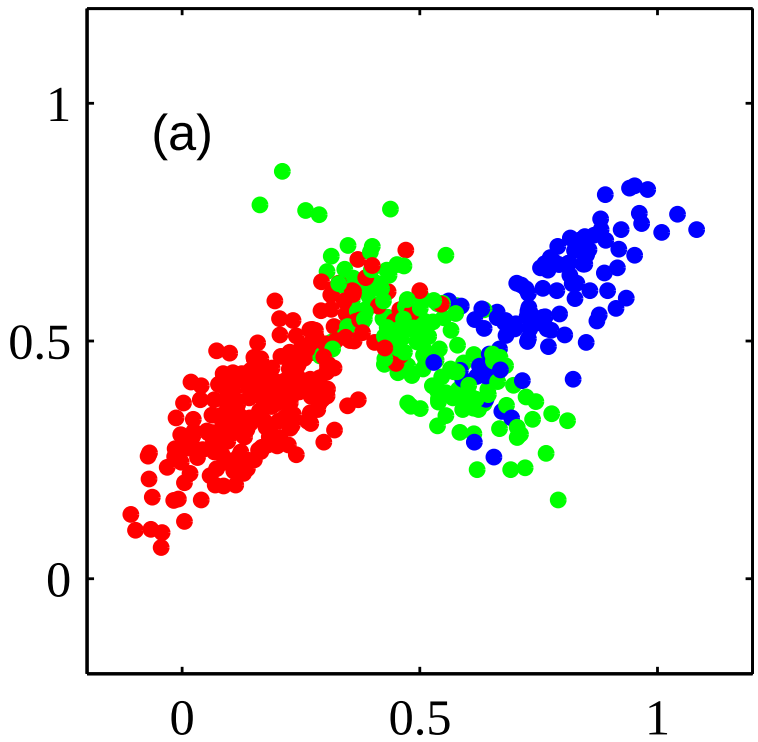
$$r_{nk} = \begin{cases} 1 & \text{if } k = \arg\min_j \|\boldsymbol{x}_n - \boldsymbol{\mu}_j\|_2^2 \\ 0 & \text{otherwise} \end{cases}$$

- **Step 2** Assume the current value of $\{r_{nk}\}$ fixed, minimize J over $\{\boldsymbol{\mu}_k\}$, which leads to the following rule to update the prototypes of the clusters

$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

- **Step 3** Determine whether to stop or return to Step 1

Motivation: Gaussian mixture models



We will model each region with a Gaussian distribution. This leads to the idea of Gaussian mixture models (GMMs) or mixture of Gaussians (MoGs).

challenge: i) we do not know which (color) region a data point comes from; ii) the parameters of Gaussian distributions in each region. We need to find all of them from *unsupervised* data $\mathcal{D} = \{\boldsymbol{x}_n\}_{n=1}^N$.

Gaussian mixture models: formal definition

A Gaussian mixture model has the following density function for $\boldsymbol{x}$

$$p(\boldsymbol{x}) = \sum_{k=}^{K} \omega_k N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

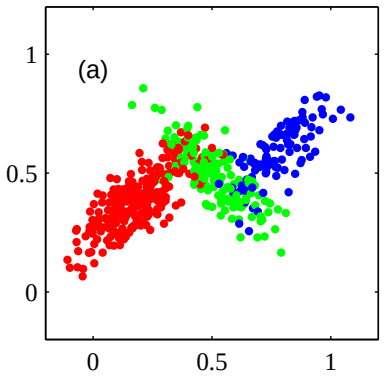
where

- K: the number of Gaussians — they are called (mixture) components
- $\boldsymbol{\mu}_k$ and $\boldsymbol{\Sigma}_k$: mean and covariance matrix of the k-th component
- ω_k: mixture weights – they represent how much each component contributes to the final distribution. It satisfies two properties:

$$\forall\ k,\ \omega_k > 0, \quad \text{and} \quad \sum_k \omega_k = 1$$

The properties ensure $p(\boldsymbol{x})$ is a properly normalized probability density function.

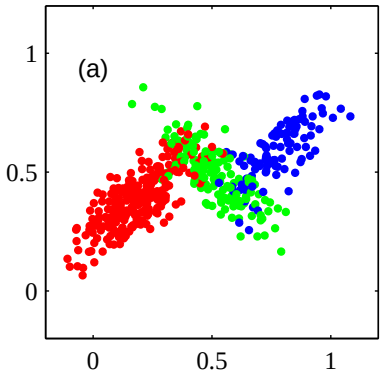
GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

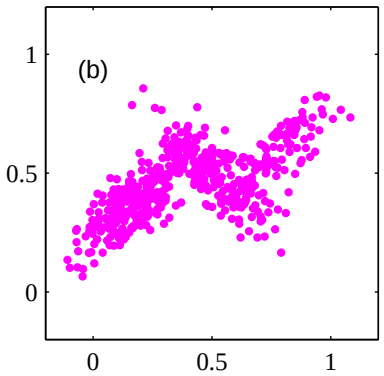
$$p(\boldsymbol{x}|z =' red') = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z =' blue') = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z =' green') = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

GMMs: example



The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

$$p(\boldsymbol{x}|z =' red') = N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$$
$$p(\boldsymbol{x}|z =' blue') = N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$$
$$p(\boldsymbol{x}|z =' green') = N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$



The marginal distribution is thus

$$p(\boldsymbol{x}) = p('red')N(\boldsymbol{x}|\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1) + p('blue')N(\boldsymbol{x}|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2) + p('green')N(\boldsymbol{x}|\boldsymbol{\mu}_3, \boldsymbol{\Sigma}_3)$$

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. To estimate, consider the simple case first.

z is given If we assume z is observed for every $\boldsymbol{x}$, then our estimation problem is easier to solve. Particularly, our training data is *augmented*

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$$

Note that, for every $\boldsymbol{x}_n$, we have a z_n to denote the region/color where the specific $\boldsymbol{x}_n$ comes from. We call $\mathcal{D}'$ the *complete* data and $\mathcal{D}$ the *incomplete* data.

Parameter estimation for Gaussian mixture models

The parameters in GMMs are $\boldsymbol{\theta} = \{\omega_k, \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k\}_{k=1}^K$. To estimate, consider the simple case first.

z is given If we assume z is observed for every $\boldsymbol{x}$, then our estimation problem is easier to solve. Particularly, our training data is *augmented*

$$\mathcal{D}' = \{\boldsymbol{x}_n, z_n\}_{n=1}^N$$

Note that, for every $\boldsymbol{x}_n$, we have a z_n to denote the region/color where the specific $\boldsymbol{x}_n$ comes from. We call $\mathcal{D}'$ the *complete* data and $\mathcal{D}$ the *incomplete* data.

Given $\mathcal{D}'$, the maximum likelihood estimation of the $\boldsymbol{\theta}$ is given by

$$\boldsymbol{\theta} = \arg\max \log \mathcal{D}' = \sum_n \log p(\boldsymbol{x}_n, z_n)$$

Intuition

Since γ_{nk} is binary, the previous solution is nothing but

- For ω_k: count the number of data points whose z_n is k and divide by the total number of data points (note that $\sum_k \sum_n \gamma_{nk} = N$)
- For $\boldsymbol{\mu}_k$: get all the data points whose z_n is k, compute their mean
- For $\boldsymbol{\Sigma}_k$: get all the data points whose z_n is k, compute their covariance matrix

This intuition is going to help us to develop an algorithm for estimating $\boldsymbol{\theta}$ when we do not know z_n.

Parameter estimation for GMMs: complete vs. incomplete data

Complete Data

γ_{nk} is binary as z_n is given

$$\gamma_{nk} = \mathbb{I}[z_n = k]$$

Incomplete Data

γ_{nk} is "guessed" as z_n is not given

$$p(z_n = k|\boldsymbol{x}_n) = \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{p(\boldsymbol{x}_n)} \quad (1)$$

$$= \frac{p(\boldsymbol{x}_n|z_n = k)p(z_n = k)}{\sum_{k'=1}^{K} p(\boldsymbol{x}_n|z_n = k')p(z_n = k')} \quad (2)$$

Same estimation formula

$$\omega_k = \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}}, \quad \boldsymbol{\mu}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k = \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}$$

Estimation with soft γ_{nk}

We are going to pretend $p(z_n = k|\boldsymbol{x}_n)$ as γ_{nk} which should be binary – but now is regarded as "soft" assigning $\boldsymbol{x}_n$ to k-th component. With that in mind, we have

$$\begin{aligned}
\gamma_{nk} &= p(z_n = k|\boldsymbol{x}_n) \\
\omega_k &= \frac{\sum_n \gamma_{nk}}{\sum_k \sum_n \gamma_{nk}} \\
\boldsymbol{\mu}_k &= \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} \boldsymbol{x}_n \\
\boldsymbol{\Sigma}_k &= \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (\boldsymbol{x}_n - \boldsymbol{\mu}_k)(\boldsymbol{x}_n - \boldsymbol{\mu}_k)^{\mathrm{T}}
\end{aligned}$$

In other words, every data point $\boldsymbol{x}_n$ is assigned to a component fractionally according to $p(z_n = k|\boldsymbol{x}_n)$ — sometimes, this quantity is also called "responsibility".

Iterative procedure

Since we do not know $\boldsymbol{\theta}$ to begin with, we cannot compute the soft γ_{nk}. However, we can invoke an iterative procedure and alternate between estimating γ_{nk} and using the estimated γ_{nk} to compute the parameters

- Step 0: guess $\boldsymbol{\theta}$ with initial values
- Step 1: compute γ_{nk} using the current $\boldsymbol{\theta}$
- Step 2: update $\boldsymbol{\theta}$ using the just computed γ_{nk}
- Step 3: go back to Step 1

Questions: i) is this procedure correct, for example, optimizing a sensible criteria? ii) practically, will this procedure ever stop instead of iterating forever?

The answer lies in the EM algorithm — a powerful procedure for model estimation with unknown data.

Demo of EM Algorithm

EM algorithm: motivation and setup

As a general procedure, EM is used to estimate parameters for probabilistic models with hidden/latent variables. Suppose the model is given by a joint distribution

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \sum_{\boldsymbol{z}} p(\boldsymbol{x}, \boldsymbol{z}|\boldsymbol{\theta})$$

where $\boldsymbol{x}$ is the observed random variable and $\boldsymbol{z}$ is hidden.

We are given data containing only the observed variable $\mathcal{D} = \{\boldsymbol{x}_n\}$ where the corresponding hidden variable values $\boldsymbol{z}$ is not included. Our goal is to obtain the maximum likelihood estimate of $\boldsymbol{\theta}$. Namely, we choose

$$\begin{aligned}\boldsymbol{\theta} &= \arg\max \log \mathcal{D} = \arg\max \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \\ &= \arg\max \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})\end{aligned}$$

The objective function $\ell(\boldsymbol{\theta})$ is called *incomplete* log-likelihood.

Expected (complete) log-likelihood

The difficulty with incomplete log-likelihood is that it needs to sum over all possible values that $\boldsymbol{z}_n$ can take, then take a logarithm. This log-sum format makes computation intractable. Instead, the EM algorithm uses a clever trick to change this into sum-log form.

To this end, we define the following

$$\begin{aligned} Q_q(\boldsymbol{\theta}) &= \sum_n \mathbb{E}_{\boldsymbol{z}_n \sim q(\boldsymbol{z}_n)} \log p(\boldsymbol{x}_n, \boldsymbol{z}_n | \boldsymbol{\theta}) \\ &= \sum_n \sum_{\boldsymbol{z}_n} q(\boldsymbol{z}_n) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n | \boldsymbol{\theta}) \end{aligned}$$

which is called *expected (complete) log-likelihood* (with respect to $q(\boldsymbol{z})$. $q(\boldsymbol{z})$ is a distribution over z. Note that $Q_q(\boldsymbol{\theta})$ takes the form of sum-log, which turns out to be tractable.

Examples

Consider the previous model where $\boldsymbol{x}$ could be from 3 regions. We can choose $q(\boldsymbol{z})$ any valid distribution. This will lead to different $Q_q(\boldsymbol{\theta})$. Note that z here represents different colors.

- $q(z = k) = 1/3$ for any of 3 colors. This gives rise to

$$Q_q(\boldsymbol{\theta}) = \sum_n \frac{1}{3} \left[\log p(\boldsymbol{x}_n, ' red' | \boldsymbol{\theta}) + \log p(\boldsymbol{x}_n, ' blue' | \boldsymbol{\theta}) + \log p(\boldsymbol{x}_n, ' green' | \boldsymbol{\theta}) \right]$$

- $q(z = k) = 1/2$ for 'red' and 'blue', 0 for 'green'. This gives rise to

$$Q_q(\boldsymbol{\theta}) = \sum_n \frac{1}{2} \left[\log p(\boldsymbol{x}_n, ' red' | \boldsymbol{\theta}) + \log p(\boldsymbol{x}_n, ' blue' | \boldsymbol{\theta}) \right]$$

Which $q(\boldsymbol{z})$ to choose?

We will choose a special $q(\boldsymbol{z}) = p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$, i.e., the posterior probability of $\boldsymbol{z}$. We define

$$Q(\boldsymbol{\theta}) = Q_{\boldsymbol{z}\sim p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})}(\boldsymbol{\theta})$$

and we will show

$$\ell(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})]$$

where $\mathbb{H}[p]$ is the entropy of the probabilistic distribution p:

$$\mathbb{H}[p(\boldsymbol{x})] = -\int p(\boldsymbol{x}) \log p(\boldsymbol{x}) d\boldsymbol{x}$$

A computable $Q(\boldsymbol{\theta})$

As before, $Q(\boldsymbol{\theta})$ cannot be computed, as it depends on the unknown parameter values $\boldsymbol{\theta}$ to compute the posterior probability $p(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta})$. Instead, we will use a known value $\boldsymbol{\theta}^{\text{OLD}}$ to compute the expected likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) = \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})$$

Note that, in the above, the variable is $\boldsymbol{\theta}$. $\boldsymbol{\theta}^{\text{OLD}}$ is assumed to be known. By its definition, the following is true

$$Q(\boldsymbol{\theta}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta})$$

However, how does $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}})$ relates to $\ell(\boldsymbol{\theta})$? We will show that

$$\ell(\boldsymbol{\theta}) \geq Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\text{OLD}})]$$

Thus, in a way, $Q(\boldsymbol{\theta})$ is better than $Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}})$ (because we have equality there) except that we cannot compute the former.

Putting things together: auxiliary function

So far we have shown a lower bound on the log-likelihood

$$\ell(\boldsymbol{\theta}) \geq A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n; \boldsymbol{\theta}^{\text{OLD}})]$$

We will call the right-hand-side an *auxiliary function*.

This auxiliary function has an important property. When $\boldsymbol{\theta} = \boldsymbol{\theta}^{\text{OLD}}$,

$$A(\boldsymbol{\theta}, \boldsymbol{\theta}) = \ell(\boldsymbol{\theta})$$

Use auxiliary function to increase log-likelihood

Suppose we have an initial guess $\boldsymbol{\theta}^{\text{OLD}}$, then we maximize the *auxiliary function*

$$\boldsymbol{\theta}^{\text{NEW}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}})$$

Use auxiliary function to increase log-likelihood

Suppose we have an initial guess $\boldsymbol{\theta}^{\text{OLD}}$, then we maximize the *auxiliary function*

$$\boldsymbol{\theta}^{\text{NEW}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}})$$

With the new guess, we have

$$\ell(\boldsymbol{\theta}^{\text{NEW}}) \geq A(\boldsymbol{\theta}^{\text{NEW}}, \boldsymbol{\theta}^{\text{OLD}}) \geq A(\boldsymbol{\theta}^{\text{OLD}}, \boldsymbol{\theta}^{\text{OLD}}) = \ell(\boldsymbol{\theta}^{\text{OLD}})$$

Use auxiliary function to increase log-likelihood

Suppose we have an initial guess $\boldsymbol{\theta}^{\text{OLD}}$, then we maximize the *auxiliary function*

$$\boldsymbol{\theta}^{\text{NEW}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{OLD}})$$

With the new guess, we have

$$\ell(\boldsymbol{\theta}^{\text{NEW}}) \geq A(\boldsymbol{\theta}^{\text{NEW}}, \boldsymbol{\theta}^{\text{OLD}}) \geq A(\boldsymbol{\theta}^{\text{OLD}}, \boldsymbol{\theta}^{\text{OLD}}) = \ell(\boldsymbol{\theta}^{\text{OLD}})$$

Repeating this process, we have

$$\ell(\boldsymbol{\theta}^{\text{EVEN NEWER}}) \geq \ell(\boldsymbol{\theta}^{\text{NEW}}) \geq \ell(\boldsymbol{\theta}^{\text{OLD}})$$

where

$$\boldsymbol{\theta}^{\text{EVEN NEWER}} = \arg\max_{\boldsymbol{\theta}} A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{NEW}})$$

Iterative and monotonic improvement

Thus, by maximizing the auxiliary function, we obtain a sequence of guesses

$$\boldsymbol{\theta}^{\text{OLD}}, \boldsymbol{\theta}^{\text{NEW}}, \boldsymbol{\theta}^{\text{EVEN NEWER}}, \cdots,$$

that will keep increasing the likelihood. This process will eventually stops if the likelihood is bounded from above (i.e.. less than $+\infty$). This is the core of the EM algorithm.

Expectation-Maximization (EM)

- Step 0: Initialize $\boldsymbol{\theta}$ with $\boldsymbol{\theta}^{(0)}$
- Step 1 (E-step): Compute the auxiliary function using the current value of $\boldsymbol{\theta}$

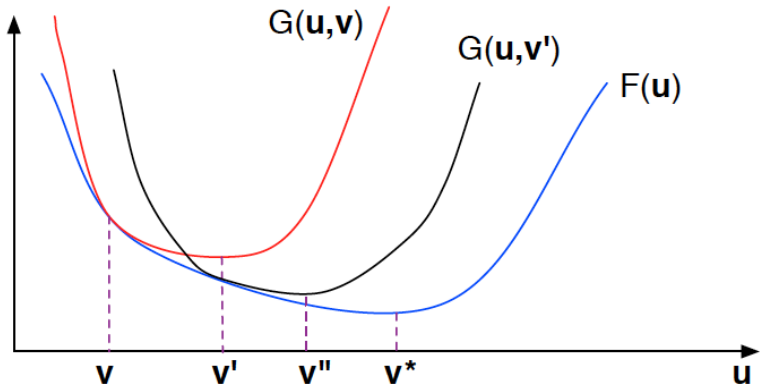
$$A(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

- Step 2 (M-step): Maximize the auxiliary function

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \arg\max A(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

- Step 3: Increase t to $t+1$ and go back to Step 1; or stop if $\ell(\boldsymbol{\theta}^{(t+1)})$ does not improve $\ell(\boldsymbol{\theta}^{(t)})$ much.

Auxiliary function (for minimizing)



- Target: minimize $F(u)$
- Auxiliary: $G(u, v) \geq F(u)$ and $G(u, u) = F(u)$
- Sequence of improvement

$$v' = \arg\min G(u, v) \to v'' = \arg\min G(u, v')$$
$$\to v''' = \arg\min G(u, v'') \cdots$$
$$F(v) \geq F(v') \geq F(v'') \geq \cdots$$

Auxiliary function used in EM

For the incomplete likelihood $\ell(\theta)$,

$$\ell(\boldsymbol{\theta}) \geq A(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) = Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n; \boldsymbol{\theta}^{\mathrm{OLD}})]$$

where the expected likelihood

$$Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) = \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n; \boldsymbol{\theta}^{\mathrm{OLD}}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})$$

Remarks

- The EM procedure converges but only converges to a local optimum. Global optimum is not guaranteed to be found.
- The E-step depends on computing the posterior probability

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}^{(t)})$$

- The M-step does not depend on the entropy term, so we need only to do the following

$$\boldsymbol{\theta}^{(t+1)} \leftarrow \arg\max A(\boldsymbol{\theta},\boldsymbol{\theta}^{(t)}) = \arg\max Q(\boldsymbol{\theta},\boldsymbol{\theta}^{(t)})$$

We often call the last term Q-function.

Example: applying EM to GMMs

What is the E-step in GMM? We compute the responsibility

$$\gamma_{nk} = p(z = k|\boldsymbol{x}_n; \boldsymbol{\theta}^{(t)})$$

What is the M-step in GMM? The Q-function is

$$\begin{aligned} Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)}) &= \sum_n \sum_k p(z = k|\boldsymbol{x}_n; \boldsymbol{\theta}^{(t)}) \log p(\boldsymbol{x}_n, z = k|\boldsymbol{\theta}) \\ &= \sum_n \sum_k \gamma_{nk} \log p(\boldsymbol{x}_n, z = k|\boldsymbol{\theta}) \\ &= \sum_k \sum_n \gamma_{nk} \log p(z = k) p(\boldsymbol{x}_n|z = k) \\ &= \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right] \end{aligned}$$

Hence, we have recovered the parameter estimation algorithm for GMMs, seen previously. (We still need to do the maximization to get $\boldsymbol{\theta}^{(t+1)}$ — left as exercise.)

GMMs and K-means

GMMs provide probabilistic interpretation for K-means. We have the following observation:

- Assume all Gaussian components have $\sigma^2 \boldsymbol{I}$ as their covariance matrices
- Further assume $\sigma \to 0$
- Thus, we only need to estimate $\boldsymbol{\mu}_k$, i.e., means
- Then, the EM for GMM parameter estimation simplifies to K-means.

For this reason, K-means is often called "hard" GMM or GMMs is called "soft" K-means. The soft posterior γ_{nk} provides a probabilistic assignment for $\boldsymbol{x}_n$ to cluster k represented by the corresponding Gaussian distribution.