CSCI567 Machine Learning (Fall 2017)

Prof. Fei Sha

U of Southern California

Lecture on Sept. 28, 2017

Outline

1. Administration
2. Review of last lecture
3. Clustering
4. Gaussian mixture models

Outline

1. Administration
2. Review of last lecture
3. Clustering
4. Gaussian mixture models

Q&A Post-Quiz 1

- We are grading your Quiz 1
- We do not know accurately how well the class is doing on Quiz 1
- We only have somme feedback
- At present time, we do not plan
  - to discuss curve grading
  - to down-weight or discard Quiz 1
  - to discuss individual circumstances

Outline

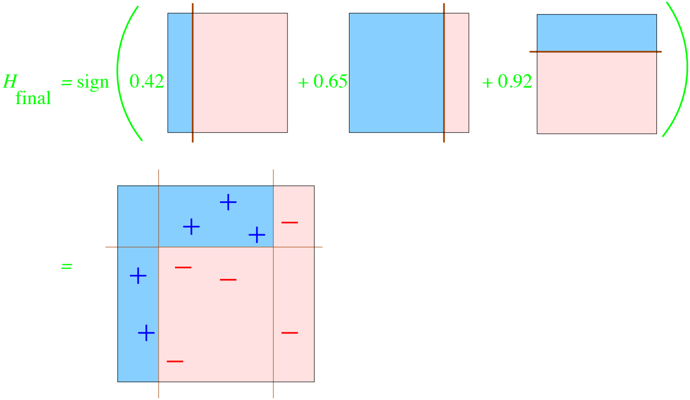
1. Administration
2. Review of last lecture
3. Clustering
4. Gaussian mixture models

How Boosting algorithm works?

- Given: N samples $\{\boldsymbol{x}_n, y_n\}$, where $y_n \in \{+1, -1\}$, and some ways of constructing weak (or base) classifiers
- Initialize weights $w_1(n) = \frac{1}{N}$ for every training sample.
- For t= 1 to T
  1. Train a weak classifier $h_t(\boldsymbol{x})$ based on the current weight $w_t(n)$, by minimizing the *weighted* classification error
  $$\epsilon_t = \sum_n w_t(n)\mathbb{I}[y_n \neq h_t(\boldsymbol{x}_n)]$$
  2. Calculate weights for combining classifiers $\beta_t = \frac{1}{2}\log\frac{1-\epsilon_t}{\epsilon_t}$
  3. Update weights
  $$w_{t+1}(n) \propto w_t(n)e^{-\beta_t y_n h_t(\boldsymbol{x}_n)}$$
  and normalize them such that $\sum_n w_{t+1}(n) = 1$.
- Output the final classifier

$$h[\boldsymbol{x}] = \text{sign}\left[\sum_{t=1}^{T}\beta_t h_t(\boldsymbol{x})\right]$$

Final classifier: combining 3 classifiers


$H_{\text{final}} = \text{sign}\left(0.42 \quad + 0.65 \quad + 0.92 \quad \right)$
=
+
+
+
+
+
–
–
–
–
–

- all data points are now classified correctly!

Outline

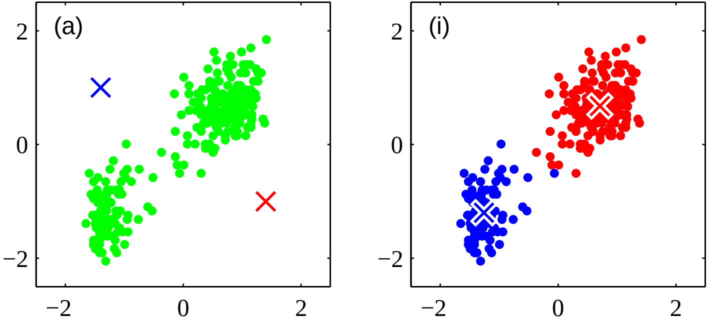
1. Administration
2. Review of last lecture
3. Clustering
4. Gaussian mixture models

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

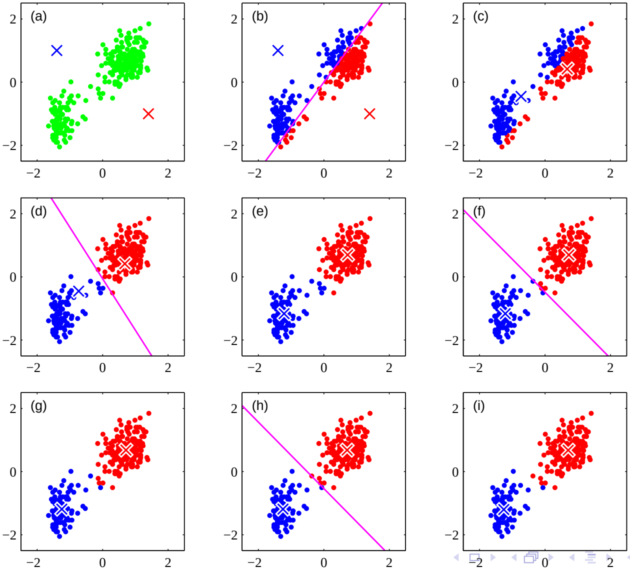
$$\boldsymbol{\mu}_k = \frac{\sum_n r_{nk}\boldsymbol{x}_n}{\sum_n r_{nk}}$$

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

Remarks

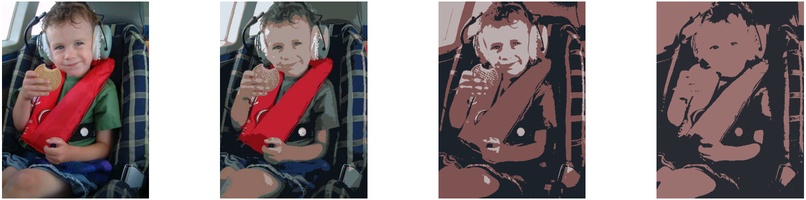
- The prototype $\boldsymbol{\mu}_k$ is *the means of data points assigned to the cluster k, hence the name K-means clustering*.
- The procedure terminates after a finite number of steps (in general, assuming that there is no tie in comparing distances in Step 1), as the procedure reduces J in both Step 1 and Step 2. Since J is lower bounded by 0, the procedure cannot be infinite.
- There is no guarantee the procedure terminates at the global optimum of J — in most cases, the algorithm stops at a local optimum, which depends on the initial values in Step 0.

Example of running K-means algorithm



Application: vector quantization

We can replace our data points with the prototypes $\boldsymbol{\mu}_k$ from the clusters they are assigned to. This is called *vector quantization*. In other words, we have compressed the data points into i) a codebook of all the prototypes; ii) a list of indices to the codebook for the data points. This compression is obviously lossy as certain information will be lost if we use a very small K.



Clustering the pixels in the image and vector quantizing them. From left to right: Original image, quantized one with a large K, a medium K, and a small K. Details are missing due to the higher compression (smaller K).

Outline

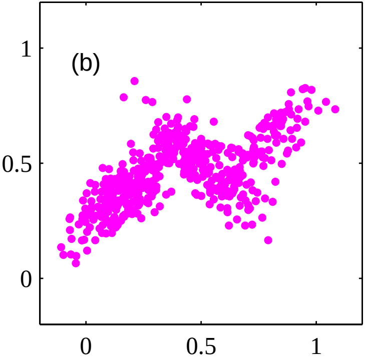
1. Administration
2. Review of last lecture
3. Clustering
4. Gaussian mixture models
    - EM Algorithm
    - Relation between K-means and GMMs
    - Demo of GMMs

Probabilistic interpretation of clustering?

We can impose a probabilistic interpretation of the intuition: data points stay close to the centers of their clusters. This is just a statement of how $p(\boldsymbol{x})$ looks like — we will see how to model this distribution.

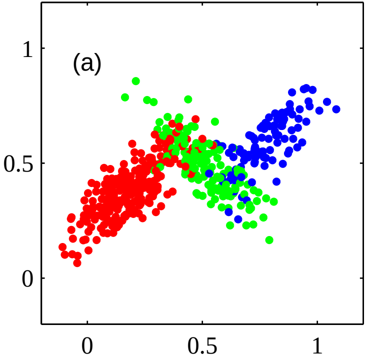
Probabilistic interpretation of clustering?

We can impose a probabilistic interpretation of the intuition: data points stay close to the centers of their clusters. This is just a statement of how $p(\boldsymbol{x})$ looks like — we will see how to model this distribution.



The data points seem to form 3 clusters. However, we cannot model $p(\boldsymbol{x})$ with simple and known distributions. For example, the data do not obey Gaussian distributions as there are seemingly 3 regions where data concentrate.

Gaussian mixture models: intuition



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

The problem we are now facing is that 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.

GMM as the marginal distribution of a joint distribution

Consider the following joint distribution

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

where z is a discrete random variable taking values between 1 and K. Denote

$$\omega_k = p(z = k)$$

and furthermore, assume the conditional distributions are Gaussian distributions

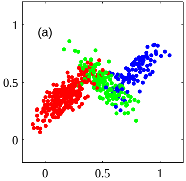
$$p(\boldsymbol{x}|z = k) = N(\boldsymbol{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

Then, the marginal distribution of $\boldsymbol{x}$ is

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

Namely, the Gaussian mixture models.

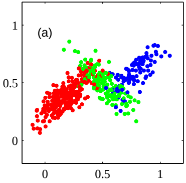
GMMs: example

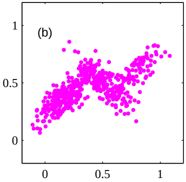


The conditional distribution between $\boldsymbol{x}$ and z (representing color) are

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

GMMs: example





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

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

The marginal distribution is thus

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

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

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

Parameter estimation for GMMs: complete data

The likelihood — which we will refer to *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n)p(\boldsymbol{x}_n|z_n) = \sum_k \sum_{n:z_n=k} \log p(z_n)p(\boldsymbol{x}_n|z_n)$$

where we have grouped data by its values z_n. Let us introduce a binary variable $\gamma_{nk} \in \{0, 1\}$ to indicate whether $z_n = k$. We can rewrite our decomposition as

Parameter estimation for GMMs: complete data

The likelihood — which we will refer to *complete* likelihood is decomposable

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_n \log p(z_n) p(\boldsymbol{x}_n | z_n) = \sum_k \sum_{n: z_n = k} \log p(z_n) p(\boldsymbol{x}_n | z_n)$$

where we have grouped data by its values z_n. Let us introduce a binary variable $\gamma_{nk} \in \{0, 1\}$ to indicate whether $z_n = k$. We can rewrite our decomposition as

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log p(z = k) p(\boldsymbol{x}_n | z = k)$$

Note that we have used a "dummy" variable z to denote all the possible values $\boldsymbol{x}_n$'s true z_n can take – but only one of the possible values is given in $\mathcal{D}'$.

Parameter estimation for GMMs: solution for complete data

Substituting our assumption about the conditional distributions, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Regrouping, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Parameter estimation for GMMs: solution for complete data

Substituting our assumption about the conditional distributions, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \left[\log \omega_k + \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)\right]$$

Regrouping, we have

$$\sum_n \log p(\boldsymbol{x}_n, z_n) = \sum_k \sum_n \gamma_{nk} \log \omega_k + \sum_k \left\{ \sum_n \gamma_{nk} \log N(\boldsymbol{x}_n | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\}$$

Note that, the term inside the braces depends on k-th component's parameters. It is now easy to show that (left as a homework exercise), the maximum likelihood estimation of the parameters are

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

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

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: incomplete data

When z_n is not given, we can guess which region/color $\boldsymbol{x}_n$ comes from by computing the posterior probability

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

Note that, to compute the posterior probability, we need to know the parameters $\boldsymbol{\theta}$. Let us for a second, we pretend we know the value of the parameters thus we can compute the posterior probability.

How is that going to help us?

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.

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 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

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

- $q(z = k) = 1/2$ 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:

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

Proof (not required to memorize)

$$
\begin{aligned}
Q(\boldsymbol{\theta}) &= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) \\
&= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \left[\log p(\boldsymbol{x}_n|\boldsymbol{\theta}) + \log p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta})\right] \\
&= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \\
&\qquad + \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \log p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) \\
&= \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}_n|\boldsymbol{x}_n;\boldsymbol{\theta}) - \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})] \\
&= \sum_n \log p(\boldsymbol{x}_n|\boldsymbol{\theta}) - \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})] \\
&= \ell(\boldsymbol{\theta}) - \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta})]
\end{aligned}
$$

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 $\boldsymbol{\theta}^{\mathrm{OLD}}$ to compute the expected likelihood

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

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

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

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

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

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

Proof (not required to memorize)

$$\begin{aligned}
\ell(\boldsymbol{\theta}) &= \sum_n \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}}) \frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}})} \\
&\geq \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}}) \log \frac{p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta})}{p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}})} \\
&= \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}}) \log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{\theta}) \\
&\qquad - \sum_n \sum_{\boldsymbol{z}_n} p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}}) \log p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}}) \\
&= Q(\boldsymbol{\theta}, \boldsymbol{\theta}^{\mathrm{OLD}}) + \sum_n \mathbb{H}[p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}})]
\end{aligned}$$

The inequality ($\geq$) is true because $\log$ is a concave function:

$$\log \sum_i w_i x_i \geq \sum_i w_i \log x_i, \quad \forall\ w_i \geq 0, \quad \sum_i w_i = 1$$

And in our case, the w_i is $p(\boldsymbol{z}|\boldsymbol{x}_n;\boldsymbol{\theta}^{\mathrm{OLD}})$.

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 $\boldsymbol{\theta}^{\mathrm{OLD}}$, then we maximize the *auxiliary function*

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

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 $\boldsymbol{\theta}^{\mathrm{OLD}}$, then we maximize the *auxiliary function*

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

With the new guess, we have

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

Repeating this process, we have

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

where

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

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 $\boldsymbol{\theta}$

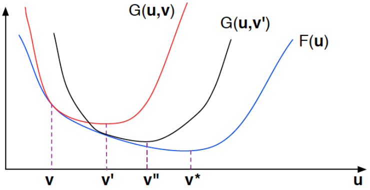
$$A(\boldsymbol{\theta}, \boldsymbol{\theta}^{(t)})$$

- 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

$$
\begin{aligned}
v' = \arg\min G(u,v) &\rightarrow v'' = \arg\min G(u,v') \\
\rightarrow v''' = \arg\min G(u,v'') \cdots \\
F(v) \geq F(v') \geq F(v'') \geq \cdots
\end{aligned}
$$

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 homework.)

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 $\sigma \rightarrow 0$
- 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.

Here we show how EM works