Chapter 4 Clustering. Mixture model and Expectation Maximization

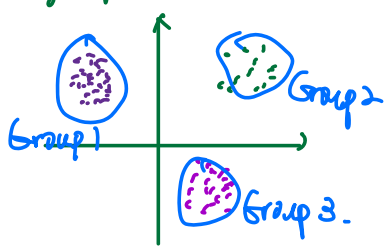
Lecture 1

1. What is clustering ?
2. K-means problem and cost function
3. K-means algorithm
4. Guarantees, convergence and local optima
5. How to pick K ?

• What is clustering ?

- Unsupervised. Often used in exploratory data analysis.
- Many approaches exists. { deterministic method (K-means)
  probabilitic method (Gaussian Mixtures)

Similarity



• K-means Problem statement + Cost function

Given:

$X_{1:N}$: a dataset $\begin{bmatrix} x_1^T \\ \vdots \\ x_N^T \end{bmatrix}_{N \times D}$ $\quad x = [\ ]_{D \times 1}$

K : designed number of clusters, must be ≥ 1

Goal:

Find locations of k cluster centers $\underline{\mu_{1:K}} \rightarrow [\ ]_{D \times 1}$ $\quad \mu = \begin{bmatrix} \mu_1^T \\ \vdots \\ \mu_K^T \end{bmatrix}_{K \times D}$

Such that if we compute the distance from each data point x_n to its nearest cluster center, and sum up the total distance. this total distance is minimized.

Cost function :

$$J(X_{1:N}, \mu_{1:K}) = \sum_{n=1}^{N} \min_{k \in \{1,2,\ldots,K\}} dist(x_n, \mu_k)$$

$$= \sum_{n=1}^{N} \min_{k \in \{1,2,\ldots,K\}} \sum_{d=1}^{D} (x_{nd} - \mu_{kd})^2$$

$$= \sum_{n=1}^{N} \min_{k \in \{1,2,\dots,k\}} (x_n - \mu_k)^T (x_n - \mu_k)$$

- Equivalent Formulation with assignment variables

Let's indicate which cluster (indexed by $k \in 1,2,\dots,k$) is closest to data point n with new variable

assignment variables

r_n is a one-hot indicator vector of size k $\quad r = [\quad]_{n \times k}$

Example: If $k=4$, the possible r_n values are:

x_n's closest cluster

4
3
2
1

$$r_n = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}$$

The cost function can be rewritten as:

$$J(x_{1:N}, \mu_{1:k}, r_{1:N}) = \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} \, \text{dist}(x_n, \mu_k)$$

$$= \sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} (x_n - \mu_k)^T (x_n - \mu_k)$$

previous: $= \sum_{n=1}^{N} \min_{k \in \{1,2,\dots,k\}} (x_n - \mu_k)^T (x_n - \mu_k)$

都了是最小的，写显式地写出来！

K Means Coordinate Descent Algorithm

- Goal: Find cluster locations $\mu_{1:k}$ and assignments $r_{1:N}$ that minimize the cost
- Opt problem: $\mu^*_{1:k}, r^*_{1:N} = \text{argmin}\, J(x_{1:N}, \mu_{1:k}, r_{1:N})$
- How to solve?
  - Gradient descent? No GD needs all continous variable
  - Coordinate descent? Yes!

    Iteratively update some vars while holding other fixed:

Each iteration has 2 steps:

Step 1. Fix $\mu = \mu^{t-1}$. Update r

$r^t = \arg\min_r J(X_{1:N}, \mu^{t-1}, r)$ → Assignment step

Step 2. Fix $r = r^t$ Update μ

$\mu^t = \arg\min_\mu J(X_{1:N}, \mu, r^t)$ → Cluster location step

- Assignment update step
  - aim: $\arg\min_{r_{1:N}} \sum_{n=1}^{N} \sum_{k=1}^{K} \underbrace{r_{nk}\, dist(X_n, \mu_k)}_{f(X_n, r_n)}$
  - Notice: No interaction between r_n terms. Each just in one term of sum.

    Can solve for each r_n value separately:

    $= \min_{r_1} f(X_1, r_1) + \min_{r_2} f(X_2, r_2) + \cdots + \min_{r_n} f(X_n, r_n)$

    $\Rightarrow r_n^* = \arg\min_{r_n \in \text{onehot}(K)} \sum_k r_{nk}\, dist(X_n, \mu_k)$

    = one hot indicator of k^*, $k^* = \arg\min_{k \in \{1,2,\cdots,K\}} dist(X_n, \mu_k)$

- Cluster Location update step:

Aim: $\sum_{n=1}^{N} \sum_{k=1}^{K} r_{nk} (X_n - \mu_k)^T (X_n - \mu_k)^T$

$\Rightarrow \sum_{k=1}^{K} \sum_{n=1}^{N} r_{nk} (X_n - \mu_k)^T (X_n - \mu_k)^T$

For each k, we want to solve:

$\arg\min_{\mu_k \in R^D} \underbrace{\sum_n r_{nk} (X_n - \mu_k)^T (X_n - \mu_k)}_{L(\mu_k)}$

Take the second derivative:

$\nabla_{\mu_k} L(\mu_k) = 0$

$\vdots$

$\mu_k^* = \frac{\sum_n r_{nk} X_n}{\sum_n r_{nk}}$ ← set μ_k to mean location of all data assigned to cluster k

K-Means Algorithm

Input: $x_{1:N}$: dataset

K : num clusters

$\mu^0_{1:K}$: initial locations — can be randomly drawn, can be picked manually, just need to be distinct

Procedure:

for iteration t in $1, 2, \cdots$ [until converged]

step 1 update $r_{1:N}$:

for n in $1, 2, \cdots N$:

Update $r_n^t \leftarrow$ to_one_hot(k^*), $k^* = \underset{k \in 1:K}{\text{argmin}}\ \text{dist}(x_n, \mu_k^{t-1})$

step 2 update $\mu_{1:K}$:

for k in $1, 2, \cdots K$:

Update $\mu_k^t \leftarrow \frac{\sum_n r_{nk}^t x_n}{\sum_n r_{nk}^t}$

Return: $\mu^*_{1:K}$ optimal cluster locations

$r^*_{1:N}$ optimal assignment indicators

Guarantees + Assessing Convergence

key idea of coordinate descent: Each step is guaranteed to improve the cost function (or at convergence, will stay the same)

$$J(x_{1:N}, \mu^0, r^1) \geq J(x_{1:N}, \mu^1, r^1) \geq J(x_{1:N}, \mu^1, r^2)$$

after iter 1 step 1 — after iter 1 step 2 — after iter 2 step 1

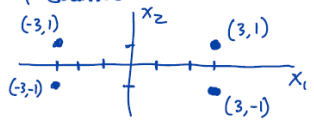
$$\cdots \geq J(x_{1:N}, \mu^T, r^T)$$

after iter T step 2

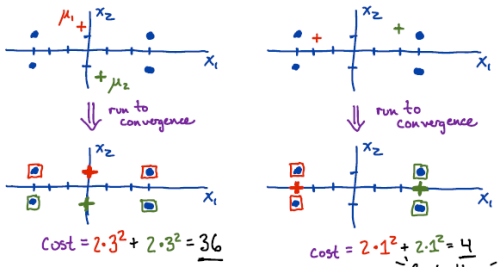
Issue: local optima (try kMeans++)

Issue: Local Optima

Consider N=4 dataset



Two possible initializations



For a given K, if you run K-means twice, might get arbitrarily worse solution from one vs. other.

To be robust, try many initializations and pick best solution

How to select number of cluster K?

How to select number of clusters K?

Not easy to do with training set alone.

Can always improve cost function by adding more clusters up to limit $K=N$, where each data point assigned to own cluster. Will have cost = 0!

Two options

(1) Pick K that minimizes heldout set cost

$$K^* = \underset{K \in \{2,3,\dots\}}{\operatorname{argmin}} \; J'(x^{val}_{1:M}, \mu^*_{1:K})$$

Might favor very large K values

$\mu^*_{1:K} = \underset{\mu_{1:K}}{\operatorname{argmin}} J'(x^{tr}_{1:N}, \mu_{1:K})$

(2) Add a "complexity penalty"

$$K^* = \underset{K \in \{2,3,\dots\}}{\operatorname{argmin}} \; J'(x_{1:N}, \mu_{1:K}) + \lambda \, \text{penalty}(K)$$

now need to pick $\lambda > 0$ and penalty function

Lecture 2. Gaussian Mixture Models

(1) Why mixture models

(2) Gaussian mixture model two view: without / with hidden assignments

(3) Estimating posterior over assignment indicators for one example

(4) Estimating parameters of GMM given dataset (LogSumExp trick)

(5) Problems with ML estimation for GMMs

Why mixture models?

Why Mixture Models?

Common data analysis problem:

We observe many examples of a diverse population

Examples: many patients in hospital have vital signs taken

many DNA sequences obtained from cells found near a tumor

We have good domain knowledge that examples come from some "types" or "groups" or clusters"

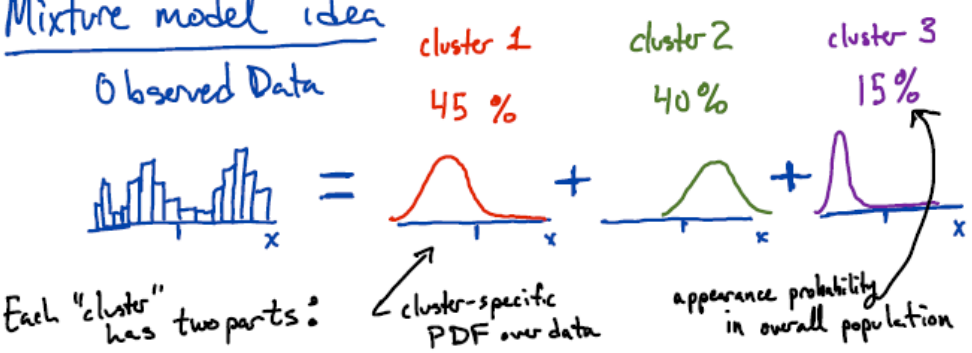
Examples: patients: disease A vs. disease B vs ... disease Z

cells: healthy vs. cancer vs. cancer w/ extra mutation

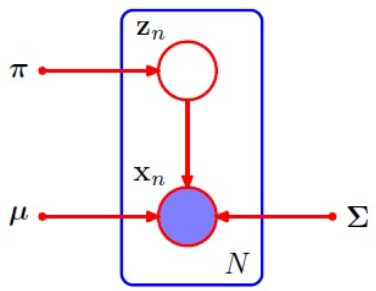
But, we just observe raw data, not cluster indicators.

Goal: Represent many clusters to get flexible model that fits data well.

Mixture model idea



Gaussian mixture models (GMM) Graphical model



Parameters:

$z = [z_1, \cdots, z_K]$

$\pi = [\pi_1, \pi_2, \cdots, \pi_K]$ $\quad \pi_k \in \Delta^K$

$\mu = [\mu_1, \mu_2, \cdots, \mu_K]$ $\quad \mu_k \in \mathbb{R}^D$

$\Sigma = [\Sigma_1, \cdots, \Sigma_K]$. $\quad \Sigma_k \in \mathbb{R}^{D \times D}$

Aim: Define a distribution over real valued vectors via a mixture of K separate Gaussian distributions.

$z : [z_1, z_2, \cdots, z_K]$ → one-hot vector size K

$z_k = \begin{cases} 1 & \text{if we assign cluster } k \text{ to generate our data } x \\ 0 & \text{otherwise} \end{cases}$

$z \sim \text{Cat}(\pi_{1:K})$: assign to cluster k with prob π_k

$p(z) = \prod_{k=1}^{K} \pi_k^{z_k}$ $\quad \pi_k \in \Delta^K$

The GMM model can be defined as:

$$p(x, z) = p(z) \cdot p(x \mid z)$$

$$= \text{Cat PMF}(z \mid \pi_{1:K}) \cdot \prod_{k=1}^{K} \text{NUMPDF}(x \mid \mu_k, \Sigma_k)^{z_k}$$

$$= \prod_{k=1}^{K} \pi_k^{z_k} \cdot N(x \mid \mu_k, \Sigma_k)^{z_k}$$

Computing posterior over assignments

Aim: Given a GMM with known parameters π, μ, Σ and a single data example $x \in \mathbb{R}^D$, which of the K clusters generated this x?

Can write this as a posterior:

$$p(z_k = 1 \mid x) = \frac{p(x \mid z_k = 1) \cdot p(z_k = 1)}{p(x)} = \frac{p(x \mid z_k = 1) \cdot p(z_k = 1)}{\sum_z p(z)\, p(x \mid z)}$$

$$= \frac{N(x|\mu_k, \Sigma_k) \cdot \pi_k}{\boxed{\sum_{\ell=1}^{K} \pi_\ell N(x|\mu_\ell, \Sigma_\ell)}} \quad \text{GMM PDF}$$

Thus, can write posterior as a categorical distribution:

$p(z|x) = \text{Cat PMF}[r_1, r_2, \dots, r_K]$

where $r_k = \frac{1}{\text{GMM PDF}(x)} \pi_k \cdot N(x|\mu_k, \Sigma_k)$

ML Estimation of Parameters

Given: N data examples $X_{1:N}$ s.t. $x_n \in \mathbb{R}^D$

K: number of assumed cluster

Goal: Estimate value of GMM parameters:

K π

$K \times D$ $\mu_{1:K}$

$K \times D \times D$ $\Sigma_{1:K}$

method: Maximize likelihood of N observed examples $X_{1:N}$

☆ logsumExp trick to avoid over/underflow

$$\pi^*, \mu^*, \Sigma^* = \arg\max_{\pi, \mu, \Sigma} \sum_{i=1}^{N} \log \text{GMM PDF}(x_n|\pi, \mu, \Sigma)$$

$$= \sum_{i=1}^{N} \log \sum_{k=1}^{K} \pi_k N(x_n|\mu_k, \Sigma_k)$$

$$= \text{logsumExp}([\log \pi_1, \dots, \log \pi_K] + [\log N(x_n|\mu_1, \Sigma_1) + \dots \log N(x_n|\mu_k, \Sigma_k)]$$

More:

scipy.special.logsumexp

$$\log \sum_k e^{a_k}$$

$$m = \max_k a_k$$

$$\log\left[e^m \left(\sum_k e^{a_k - m}\right)\right]$$

$$m + \log \sum_k e^{a_k - m}$$

Example:

log sum exp

$$\log\left(e^{1000} + e^0 + e^0\right)$$

$$\approx 1000 + \varepsilon$$

np.log(np.exp(1000) + np.exp(0)...)

+inf

$$\log\left(e^{1000}\left(e^0 + e^{-1000} + e^{-1000}\right)\right)$$

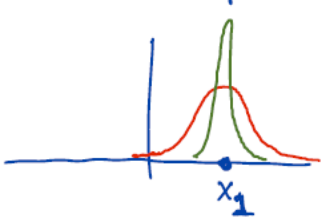
$$\log\left(e^{1000}\right) + \log\left(e^0 + e^{-1000} + e^{-1000}\right)$$

Problem with ML estimation.

Problems with ML Estimation

As we've seen before, ML estimation can have problems with poor generalization performance especially when data set size is small.

Consider $K=1$ GMM (basically, a single Gaussian) for $N=1$ example with $D=1$ dimension. $\Sigma_k = \sigma_k^2$ scalar variance



ML goal:

$$\max_{\substack{\mu_1 \in \mathbb{R} \\ \sigma_1 > 0}} \log \text{NormPDF}(x_1 | \mu_1, \sigma_1^2)$$

Two solutions shown

- $\mu_1 = x_1, \sigma_1 = 1$
- $\mu_1 = x_1, \sigma_1 = \frac{1}{100}$

Problem: We can make this point's PDF value $\to +\infty$ as $\sigma_1 \to 0^+$

Consider more general GMMs:

Problems arise when $N >> 1$ and $K \geq 2$, when one cluster with $\sigma_k \to 0$ lands on top of one point, and other clusters explain other examples

Problem summary: ML can yield — zero variance — +∞ PDF

- Want all clusters to have non-zero variance to have hope of generalizing well.
- Want finite PDF values even on training set for sensible model selection

Ways to Avoid this Problem

(1) Constrain variances away from zero

$\sigma^2 > \varepsilon$ not $\sigma^2 > 0$

for some $\varepsilon >> 0$, say 0.01

(2) Penalized ML

$\max \log p(x_{1:N}) - \lambda \, \text{penalty}(\sigma)$

Add penalty term to objective with weight $\lambda > 0$

Penalty term gives high "cost" to problematic σ values

(3) Do MAP instead of ML

Pick a prior distribution $p(\sigma)$ that favors variances far from zero

$\max \log p(x_{1:N}) + \log p(\sigma)$

Figure 9.7 Illustration of how singularities in the likelihood function arise with mixtures of Gaussians. This should be compared with the case of a single Gaussian shown in Figure 1.14 for which no singularities arise.

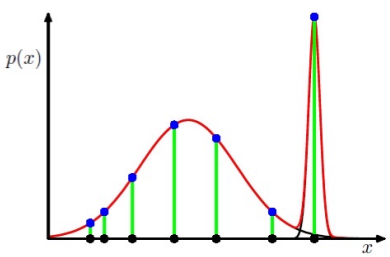
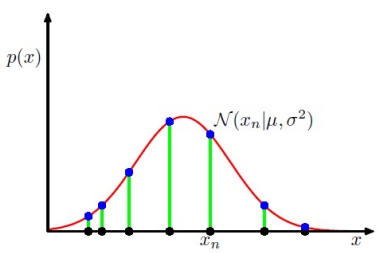


Figure 1.14 Illustration of the likelihood function for a Gaussian distribution, shown by the red curve. Here the black points denote a data set of values $\{x_n\}$, and the likelihood function given by (1.53) corresponds to the product of the blue values. Maximizing the likelihood involves adjusting the mean and variance of the Gaussian so as to maximize this product.



Lecture 3. EM algorithm and Gradient Descent

1. Penalized ML optimization problem for GMMs
2. Gradient Descent for GMMs
3. Derivation of Coordinate Descent for GMMs
4. The EM Coordinate Descent Algorithm for GMMs

Penalized ML optimization problem for GMMs

- Parameters:

  K π: K-length vector with non-negative entries that sum to one.

  KxD μ: μ_k is D-length vector

  KxDxD Σ: Σ_k is DxD symmetric, PD matrix

- object:

$$\pi^*, \mu^*, \Sigma^* = \underset{\pi,\mu,\Sigma}{\arg\max} \sum_{i=1}^{N} \log \text{GMMPDF}(x_n | \pi, \mu, \Sigma)$$

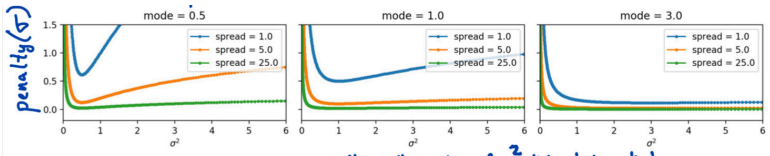
$$= \underset{\pi,\mu,\Sigma}{\arg\min} - \sum_{i=1}^{N} \log \text{GMMPDF}(x_n | \pi, \mu, \Sigma)$$

- object with penalty:
  - Goal: Do ML, but avoid pathology where one cluster has variance shrink to 0 to get "infinite" likelihood.
  - Instead of "full" DxD covariance, lets assume $\Sigma_k = \begin{bmatrix} \sigma_{k1}^2 & & \\ & \sigma_{k2}^2 & \\ & & \ddots \\ & & & \sigma_{kD}^2 \end{bmatrix}$

    Then we can define penalty over each dimension's Std. deviation Scaler:

$$\text{penalty}(\sigma) = \frac{1}{m^2 s} \log \sigma + \frac{1}{2} \frac{1}{m \cdot s} \frac{1}{\sigma^2}$$

  - Visualization and interpretation



$M > 0$ is the "mode", value of σ^2 that is least penalized

$S > 0$ is the "spread", higher values → larger region around mode is favored

Note: σ^2 near zero is always highly penalized

Can view is penalty as closely related to a Gamma distribution prior on the variance. Connects penalized ML to MAP estimation

We emphasize this is one possible penalty.
The diagonal covariance assumption is for simplicity.

- object after penalty:

$$\pi^*, \mu^*, \Sigma^* = \underset{\pi, \mu, \Sigma}{\operatorname{argmin}} -\sum_{i=1}^{n} \log \text{GMMPDF}(x_i \mid \pi, \mu, \Sigma) + \lambda \sum_{k=1}^{K} \sum_{d=1}^{D} \text{penalty}(\sigma_{kd}^2)$$

- use gradient descent:

Sketch: Input: initial values π^0, μ^0, Σ^0

Procedure: for iter t in 1, 2, ... until converged:

$$\pi^t \leftarrow \pi^{t-1} - \varepsilon \nabla_\pi \alpha(\pi^{t-1}, \mu^{t-1}, \Sigma^{t-1})$$
$$\mu^t \leftarrow \mu^{t-1} - \varepsilon \nabla_\mu \alpha(\pi^{t-1}, \mu^{t-1}, \Sigma^{t-1})$$
$$\Sigma^t \leftarrow \Sigma^{t-1} - \varepsilon \nabla_\Sigma \alpha(\pi^{t-1}, \mu^{t-1}, \Sigma^{t-1})$$

Δ • Problem:

Both π and Σ are constrained

π must sum to one and be non-negative

Σ_k must be symmetric and PD

The GD update may produce values that violate constraints

$\pi^t \leftarrow \pi^{t-1} - \varepsilon \cdot g^{t-1}$ will not keep $\pi^t \in \Delta^K$

- Solution: Reparameterize! Find unconstrained parameterization that maps to our constraint version

For π:

Let $\pi = [\pi_1, \pi_2, \cdots, \pi_K]$ be a "probability vector". That means π defines a discrete PMF over K choices. Mathematically, π must have non-negative entries that sum to one.

Define transform function $\pi(\cdot) = \mathbb{R}^K \to \Delta^K$ (softmax)

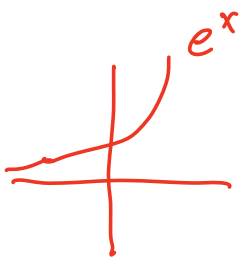
$$\pi([s_1, \cdots, s_K]) = \left[\frac{e^{s_1}}{Z}, \frac{e^{s_2}}{Z}, \cdots, \frac{e^{s_K}}{Z}\right]$$

where $Z = \sum_{e=1}^{K} e^{s_e}$

For σ

Let σ_K be a vector that must be positive

$$\sigma_K = [\sigma_{K1}, \cdots, \sigma_{KD}]$$



Option 1:

Define $t_K \in \mathbb{R}^D$, and use transformation $\sigma(\cdot)$

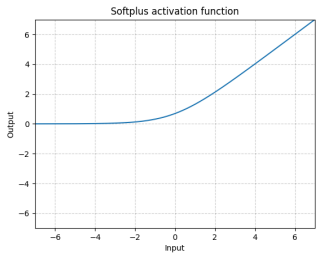
$$\sigma(t_K) \triangleq [e^{t_{K1}}, e^{t_{K2}}, \cdots, e^{t_{KD}}]$$

by definition. for any input t, $\sigma(t)$ should have positive entries.

Option 2:

Softplus function

$$\text{softplus}(x) = \log(1 + e^x)$$



$$\sigma(t_K) \triangleq [\text{softplus}(t_{K1}), \text{softplus}(t_{K2}), \cdots \text{softplus}(t_{KD})]$$

Option 2 might be better in GD, because less sensitive to small changes in input t.

- Summary for GD for GMMs

Gradient Descent for GMMs.

Parameters: $s : s_{1:K}$, $s_k \in \mathbb{R}$; $\mu : \mu_{1:K}$, $\mu_k \in \mathbb{R}^D$; $t : t_{1:K}$, $t_k \in \mathbb{R}^D$ — all unconstrained real values

Goal: $s^*, \mu^*, t^* = \operatorname{argmin} \alpha^{TPML}(s, \mu, t)$

then set $\pi^* \leftarrow \pi(s^*)$, $\sigma^* \leftarrow \sigma(t^*)$. Return π^*, μ^*, σ^*

R for re-parameterized

Loss: $\alpha^{RPML}(s, \mu, t) = \alpha^{PML}(\pi(s), \mu, \sigma(t))$

Grad: $\nabla_s \alpha^{RPML} = \nabla_s \pi(s) \cdot \nabla_\pi \alpha^{PML}(\pi(s), \cdots)$

$\nabla_t \alpha^{RPML} = \nabla_t \sigma(t) \cdot \nabla_\sigma \alpha^{PML}(\cdots, \sigma(t))$

chain rule!

GD is now possible!
Can do gradients by hand or via automatic differentiation

Keep in Mind:
- still need to select step size $\varepsilon > 0$ (as in any GD)
- loss has many local optima. May need many runs of GD to find a decent solution

Coordinate descent method Intuition + derivation

- objective:

$$\pi^*, \mu^*, \Sigma^* = \arg\min - \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \, \text{MVNPDF}(x_n | \mu_k, \Sigma_k)$$

- Idea:

Analytically consider an optimal set of parameters π^*, μ^*, Σ^*.

To be optimal, they must satisfy these blue equations.

$$\nabla_{\pi_k} \mathcal{L}^{ML}(\pi^*, \mu^*, \Sigma^*) = \vec{0}_K$$

$$\nabla_{\mu_k} \mathcal{L}^{ML}(\pi^*, \mu^*, \Sigma^*) = \vec{0}_D$$

$$\nabla_{\Sigma_k} \mathcal{L}^{ML}(\pi^*, \mu^*, \Sigma^*) = \vec{0}_{D \times D}$$

expand out

$$\nabla_{\mu_k} \mathcal{L}^{ML} = \sum_{n=1}^{N} \nabla_{\mu_k} \left[\log \text{GMMPDF}(x_n | \pi, \mu, \Sigma) \right]$$

$$= \sum_{n=1}^{N} \frac{1}{\text{GMMPDF}(x_n)} \cdot \nabla_{\mu_k} \text{GMMPDF}(x_n | \pi, \mu, \Sigma)$$

$$= \sum_{n=1}^{N} \frac{1}{\text{GMMPDF}(x_n)} \cdot \nabla_{\mu_k} \left[\sum_{l=1}^{K} \pi_l \, c(\Sigma_l) \cdot e^{-\frac{1}{2}(x_n - \mu_l)^T \Sigma_l^{-1} (x_n - \mu_l)} \right]$$

$$= \sum_{n=1}^{N} \frac{1}{\text{GMMPDF}(x_n)} \pi_k \, c(\Sigma_k) \cdot \nabla_{\mu_k} e^{-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)}$$

$$= \sum_{n=1}^{N} \frac{1}{\text{GMMPDF}(x_n)} \underbrace{\pi_k \, c(\Sigma_k) \cdot e^{-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k)}}_{p(z_{nk}=1, x_n)} \cdot \nabla_{\mu_k} \left[-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) \right]$$

$$= \sum_{n=1}^{N} \underbrace{\frac{1}{\text{GMMPDF}(x_n)} \cdot p(z_{nk}=1, x_n)}_{p(x_n)} \cdot \nabla_{\mu_k} \left[-\frac{1}{2}(x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) \right]$$

Define $\gamma_{nk} = \frac{p(z_{nk}=1, x_n)}{p(x_n)} = p(z_{nk}=1 | x_n)$ → posterior

$$= \sum_{n=1}^{N} \gamma_{nk} \cdot \Sigma_k^{-1} (x_n - \mu_k)$$

Insight: Can simplify use soft "responsibilities" $\gamma_n \in \Delta^K$

fix these, and become easy to solve

$$\sum_{n=1}^{N} \gamma_{nk} \Sigma_k^{-1} (x_n - \mu_k) = 0$$

$$\Sigma_k^{-1} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k) = 0$$

$$\mu_k^* = \frac{\sum_n \gamma_{nk} x_n}{\sum_n \gamma_{nk}}$$

Interpretation: weighted empirical mean of data assigned to cluster k

Similarly:

$$\pi_k^* = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk} \quad \text{(PRML 9.22)}$$

fraction of all data points assigned to cluster k

$$\Sigma_k^* = \frac{1}{\sum_n \gamma_{nk}} \sum_{n=1}^{N} \gamma_{nk} (x_n - \mu_k^*)(x_n - \mu_k^*)^T \quad (9.19)$$

weighted empirical covariance of data assigned to cluster k

EM algorithm for GMM

- Algorithm for estimating the parameters of a GMM using (penalized) maximum likelihood

Coordinate Descent algorithm for GMMs

Similar to KMeans

Minimizes the negative ML loss function $\mathcal{L}^{ML}$

Known as Expectation-Maximization or "E-M"
for reasons discussed in next class

Input: $X_{1:N}$ dataset

$\pi^0, \mu^0_{1:K}, \Sigma^0_{1:K}$ initial parameters

Procedure: for iter $t \in 1, 2, \cdots$ [until converged]:

E step — update responsibilities

for example $n \in 1, 2, \cdots N$:
for cluster $k \in 1, 2, \cdots K$:

$$\tilde{\gamma}_{nk} \leftarrow \pi_k^{t-1} \text{MVNormPDF}(x_n | \mu_k^{t-1}, \Sigma_k^{t-1})$$

$$\gamma_n \leftarrow \left[\frac{\tilde{\gamma}_{n1}}{\sum_k \tilde{\gamma}_{nk}}, \frac{\tilde{\gamma}_{n2}}{\sum_k \tilde{\gamma}_{nk}}, \cdots \frac{\tilde{\gamma}_{nK}}{\sum_k \tilde{\gamma}_{nk}} \right]$$

M step — update parameters

for cluster $k \in 1, 2, \cdots K$:

$$\pi_k^t \leftarrow \frac{1}{N} \sum_n \gamma_{nk}$$

$$\mu_k^t \leftarrow \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} x_n$$

$$\Sigma_k^t \leftarrow \frac{1}{\sum_n \gamma_{nk}} \sum_n \gamma_{nk} (x_n - \mu_k^t)(x_n - \mu_k^t)^T$$

could do with diagonal parameterization $\Sigma_k = [\sigma_1^2 \cdots \sigma_D^2]$ can adjust to penalty term

Will converge to a fixed point where π^*, μ^*, Σ^* is local optima of $\mathcal{L}^{ML}$

- Lecture 4 A new view of EM
  (as a principle, general purpose optimization algorithm)

Topics: Recap: EM for GMMs
GMM as latent variable model

Idea 1: Complete likelihood easier than incomplete
Idea 2: Expectation of complete likelihood also easy
Idea 3: Can formulate objective function that:
- Principled: Is a lower bound of incomplete likelihood
- Tractable: Uses expectations of complete likelihood

Idea 4: EM is coordinate ascent optimization applied to this objective

• Recap EM for GMMs

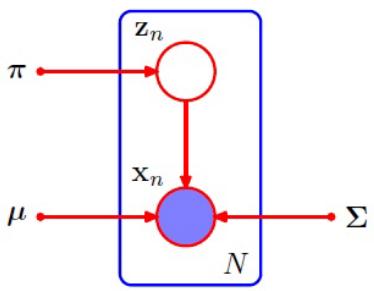
Big idea:

Coordinate ascent algorithm: each substep updates subset of variables (coordinates)

Open question:

- What principles let us use EM?
- What objective are we optimizing?

• Step back: Latent Variable Models



z_n is an one-hot indicates, cluster assigned to example n $\quad z_n \in \text{onehot}(k)$

z_n is latent or "hidden", we cannot observe it directly.

$$p(z_n) = \text{Cat}(\pi_1, \pi_2, \ldots, \pi_k) = \prod_k \pi_k^{z_{nk}}$$

$$p(x_n | z_n) = \prod_{k=1}^{K} \text{NormPDF}(x_n | \mu_k, \sigma_k^2)^{z_{nk}}$$

(z_n is a one-hot, so only one term in each product will be used)

Joint: $p(x_n, z_n) = p(z_n) \cdot p(x_n | z_n)$

Posterior: $p(z_n | x_n) = \frac{p(x_n, z_n)}{p(x_n)} = \frac{p(x_n, z_n)}{\sum_{z_n} p(x_n, z_n)}$

Assume $z_{nk} = 1$

$$\gamma_{nk} = p(z_{nk}=1 | x_n) = \frac{\pi_k \, \text{Norm}(x_n | \mu_k, \Sigma_k)}{\sum_{\ell}^{K} \pi_\ell \, \text{Norm}(x_n | \mu_\ell, \Sigma_\ell)}$$

$$p(z_n | x_n) = \text{Cat}(\gamma_{n1}, \gamma_{n2}, \cdots, \gamma_{nK}) \qquad \gamma_{nk} \geq 0, \quad \sum_k \gamma_{nk} = 1$$

Idea 1: Complete likelihood is easier to optimize

Incomplete likelihood: (marginal)

$$p(x_n) = \sum_k \pi_k \, N(x_n | \mu_k, \sigma_k^2)$$

$$\text{ML: } \pi^*, \mu^*, \sigma^* = \underset{\pi, \mu, \sigma}{\text{argmax}} \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \, N(x_n | \mu_k, \sigma_k^2)$$

Take derivative of μ_k:

$\log(f(\mu_1) + f(\mu_2) \cdots + f(\mu_K))$

Very complicated! $\mu_1, \mu_2, \cdots \mu_K$ is interactive

Complete likelihood (joint)

$$\max_{\pi, \mu, \sigma^2} \sum_n \log p(x_n, z_n)$$

$$\Rightarrow \sum_n \log \prod_{k=1}^{K} \pi_k^{z_{nk}} \cdot N(x_n | \mu_k, \sigma_k^2)^{z_{nk}}$$

$$\Rightarrow \sum_n \sum_k z_{nk} \log \pi_k + z_{nk} \log N(x_n | \mu_k, \sigma_k^2)$$

not interactive. easier to optimize.

Idea 2. Can optimize expectation of complete likelihood

Suppose we have a distribution over z that we thought was accurate:

$$q(z_n) = \text{Cat}(r_{n1}, r_{n2}, \dots, r_{nK})$$

$E_q[z_{nk}] = r_{nk}$

$p(z_n = 1) = r_{nk}$

$p(z_n = 0) = 1 - r_{nk}$

mean: $1 \times r_{nk} + 0 \times (1 - r_{nk}) = r_{nk}$

Suppose we wanted to compute

$$E_{q(z)}[\log p(x, z)]$$

$$= E_{q(z)}\left[\sum_n \sum_k z_{nk} \log \pi_k + z_{nk} \log N(x_n \mid \mu_k, \sigma_k^2)\right]$$

$$= \sum_n \sum_k \underbrace{E_q[z_{nk}]}_{r_{nk}} \log \pi_k + E_q[z_{nk}] \log N(x_n \mid \mu_k, \sigma_k)$$

Punchline: Expectations of complete log likelihood are easy to evaluate.
easy to optimize for π, μ, σ^2

Idea 3. We can develop an objective

(a) a lower bound on $\log p(x)$, so can be interpreted in principled way as optimizing likelihood

(b) uses expectations of complete likelihood, so it is tractable

$$\log p(x_n) = \log p(x_n) \cdot \sum_{z_n} q(z_n)$$

$$= \sum_{z_n} \log p(x_n) \cdot q(z_n)$$

$$= E_{q(z_n)} \log p(x_n)$$

$$= E_{q(z_n)} \log p\left(\frac{p(x_n, z_n)}{p(z_n \mid x_n)}\right)$$

$$= E_{q(z_n)} \log p\left(\frac{p(x_n, z_n)}{p(z_n \mid x_n)} \cdot \frac{q(z_n)}{q(z_n)}\right)$$

$$= E_{q(z_n)} \log p\left(\frac{p(x_n, z_n)}{q(z_n)} \cdot \frac{q(z_n)}{p(z_n \mid x_n)}\right)$$

$$= E_{q(z_n)} \log p\left(\frac{p(x_n, z_n)}{q(z_n)}\right) + \log\left(\frac{q(z_n)}{p(z_n \mid x_n)}\right)$$

Notation

$\sum_{z_n} q(z_n) = 1$

$E(x) = \sum x_i p_i$

Bayes: $p(z_n \mid x_n) = \frac{p(x_n, z_n)}{p(x_n)}$

$$= E_{q(z_n)} \log p\left(\frac{p(x_n, z_n)}{q(z_n)}\right) + E_{q(z_n)}\left[\log \frac{q(z_n)}{p(z_n|x_n)}\right]$$

$$= E_{q(z_n)} \log p\left(\frac{p(x_n, z_n)}{q(z_n)}\right) + \sum q(z_n) \cdot \log \frac{q(z_n)}{p(z_n|x_n)}$$

$$\log p(x_n) = \underbrace{E_{q(z_n)} \log p[x_n, z_n) - E_{q(z_n)} \log q(z_n)}_{\text{expected complete likelihood}} + \underbrace{KL(q(z_n) \| p(z_n|x_n))}_{\text{entropy of } q(z_n)}$$

KL divergence:

$KL(p\|q) = \sum p(x) \log \frac{p(x)}{q(x)}$

$KL \geq 0$

entropy:

$H(x) = -\sum_{i=1}^{n} p(x_i) \log p(x_i)$

We have a principled lower bound of the incomplete likelihood!

$$\log p(x_n) \geq \boxed{E_{q(z_n)} \log \frac{p(x_n, z_n)}{q(z_n)}}$$

Evidence Lower Bound (ELBO)

$$\mathcal{L}(x_n, r_n, \pi, \mu, \sigma) = E_{q(z_n)} \log p[x_n, z_n) - E_{q(z_n)} \log q(z_n)$$

we can easily calculate $\mathcal{L}$ given

- x data
- r_n prob vector
- π weights
- μ means
- σ variances

} GMM parameters

最后可得:

$$\Rightarrow \sum_K \left(r_{nk} \log \pi_k + r_{nk} \log \text{NormPDF}(x_n | \mu_k, \sigma_k^2) - r_{nk} \log r_{nk}\right)$$

Idea 4. EM is coordinate ascent optimization applied to the objective.

New View of EM:

Optimizing an objective $\mathcal{L}$ that is lower bound of incomplete likelihood

$$\log p(x \mid \pi, \mu, \tau) \geq \sum_n \mathcal{L}(x_n, r_n, \pi, \mu, \tau)$$

E-step M-step

E step: Visit every example n, find $q(z_n \mid r_n)$ that maximizes $\mathcal{L}$ given current params π, μ, τ

$$r_n = \underset{r_n \in \Delta^K}{\operatorname{argmax}} \mathcal{L}(x_n, r_n, \pi, \mu, \tau^2)$$

Visit each k:

M step: Find point estimate of each param π, μ, τ that maximizes whole dataset objective

$$\pi, \mu, \tau = \underset{\pi, \mu, \tau}{\operatorname{argmax}} \sum_n \mathcal{L}(x_n, r_n, \pi, \mu, \tau)$$

EM (coordinate descent) guarantee improvement!

Question: How good is $\mathcal{L}$ as a bound?

Earlier, we saw

$$\log p(x_n) = \mathcal{L}(x_n, r_n, \pi, \mu, \tau) + KL(q(z) \| p(z|x))$$

KL always ≥ 0

Recall KL equals 0 ONLY when $q(z) = p(z|x)$

Earlier, we said

$$p(z_n | x_n) = \text{Cat}(\gamma_{n1}, \gamma_{n2}, \ldots \gamma_{nK})$$

where $\gamma_{nk} = \frac{\pi_k \text{Norm}(x_n | \mu_k, \tau_k^2)}{\sum_l \pi_l \text{Norm}(x_n | \mu_l, \tau_l^2)}$

△

So if we set $r_n = \gamma_n$, then KL term $= 0.0$ and bound is tight

Turns out, this is optimal E step update. $\log p(x_n) = \mathcal{L}(x_n, \gamma_n, \pi, \mu, \tau)$ with equality

EM for Gaussian Mixtures

Given a Gaussian mixture model, the goal is to maximize the likelihood function with respect to the parameters (comprising the means and covariances of the components and the mixing coefficients).

1. Initialize the means $\boldsymbol{\mu}_k$, covariances $\boldsymbol{\Sigma}_k$ and mixing coefficients π_k, and evaluate the initial value of the log likelihood.
2. **E step**. Evaluate the responsibilities using the current parameter values

$$\gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}. \tag{9.23}$$

3. **M step**. Re-estimate the parameters using the current responsibilities

$$\boldsymbol{\mu}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n \tag{9.24}$$

$$\boldsymbol{\Sigma}_k^{\text{new}} = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right) \left(\mathbf{x}_n - \boldsymbol{\mu}_k^{\text{new}}\right)^{\mathrm{T}} \tag{9.25}$$

$$\pi_k^{\text{new}} = \frac{N_k}{N} \tag{9.26}$$

Page 15

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}). \tag{9.27}$$

4. Evaluate the log likelihood

$$\ln p(\mathbf{X}|\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \right\} \tag{9.28}$$

and check for convergence of either the parameters or the log likelihood. If the convergence criterion is not satisfied return to step 2.

The General EM Algorithm

Given a joint distribution $p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta})$ over observed variables $\mathbf{X}$ and latent variables $\mathbf{Z}$, governed by parameters $\boldsymbol{\theta}$, the goal is to maximize the likelihood function $p(\mathbf{X}|\boldsymbol{\theta})$ with respect to $\boldsymbol{\theta}$.

1. Choose an initial setting for the parameters $\boldsymbol{\theta}^{\text{old}}$.

2. **E step** Evaluate $p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}})$.

3. **M step** Evaluate $\boldsymbol{\theta}^{\text{new}}$ given by

$$\boldsymbol{\theta}^{\text{new}} = \arg\max_{\boldsymbol{\theta}} \mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) \tag{9.32}$$

   where

$$\mathcal{Q}(\boldsymbol{\theta}, \boldsymbol{\theta}^{\text{old}}) = \sum_{\mathbf{Z}} p(\mathbf{Z}|\mathbf{X}, \boldsymbol{\theta}^{\text{old}}) \ln p(\mathbf{X}, \mathbf{Z}|\boldsymbol{\theta}). \tag{9.33}$$

4. Check for convergence of either the log likelihood or the parameter values. If the convergence criterion is not satisfied, then let

$$\boldsymbol{\theta}^{\text{old}} \leftarrow \boldsymbol{\theta}^{\text{new}} \tag{9.34}$$

   and return to step 2.