SPR Day 19

day 17 in 2023s

Algorithms for estimating the parameters of a Gaussian mixture model using (penalized) maximum likelihood

Readings: Bishop PRML Sec. 9.2.1 (ML for GMMs)

Sec. 9.2.2 (EM for GMMs)

Outline:

(1) Maximum Likelihood (ML) and Penalized ML Optimization Problems

not in Bishop PRML → (2) Gradient descent methods
intuition + derivation

(3) Coordinate descent methods
intuition + derivation

(4) Expectation-Maximization algorithm

ML Estimation: Problem Statement

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

K : number of assumed clusters

Goal: Estimate values of all GMM parameters

total size		
K	$\pi = \pi_{1:K}$	: K-length vector with non-negative entries that sum to one
$K \times D$	$\mu = \mu_{1:K}$	: μ_k is D-length vector of reals
$K \times D \times D$	$\Sigma = \Sigma_{1:K}$	: Σ_k is $D \times D$ sym, pos. definite matrix

Method: Maximize likelihood of N observed examples $x_{1:N}$ under model where each $x_n \overset{iid}{\sim} \mathrm{GMM}(\pi, \mu, \Sigma)$

Objective:

$$\pi^*, \mu^*, \Sigma^* = \underset{\substack{\pi \in \Delta^K \\ \mu : \mu_k \in \mathbb{R}^D \\ \Sigma : \Sigma_k \in \text{D×D sym pos def}}}{\operatorname{argmax}} \sum_{n=1}^{N} \log \mathrm{GMMPDF}(x_n | \pi, \mu, \Sigma)$$

equivalently as a min problem:

$$\pi^*, \mu^*, \Sigma^* = \underset{\substack{\pi \in \Delta^K \\ \mu : \mu_k \in \mathbb{R}^D \\ \Sigma : \Sigma_k \in D \times D}}{\operatorname{argmin}} \underbrace{-1 \cdot \sum_{n=1}^{N} \log \mathrm{GMMPDF}(x_n | \pi, \mu, \Sigma)}_{\mathcal{L}^{ML}(\pi, \mu, \Sigma)}$$

Loss function to minimize: $\mathcal{L}^{ML}$

Penalty on Variance Parameter

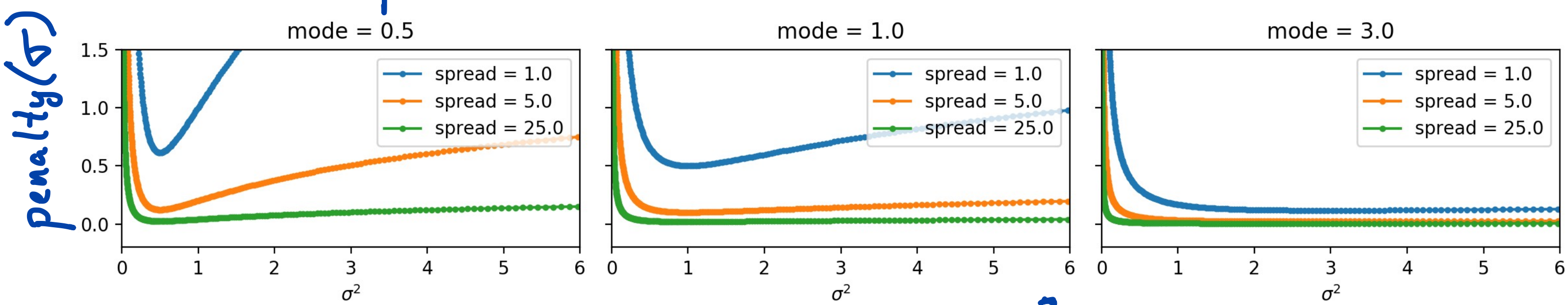
Goal: Do ML, but avoid pathology where one cluster has variance shrink to 0 to get "infinite" likelihood

Instead of "full" D×D covariance, let's assume $\Sigma_k = \begin{bmatrix} \sigma_{k1}^2 & & 0 \\ & \sigma_{k2}^2 & \\ 0 & & \ddots \; \sigma_{kD}^2 \end{bmatrix}$ call this a "diagonal" covariance.

all zero

Then we can define penalty over each dimension's std. deviation scalar

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



$m > 0$ 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.

Penalized ML Estimation: Problem Statement

$$\pi^*, \mu^*, \Sigma^* = \underset{\substack{\pi \in \Delta^K \\ \mu:\ \mu_k \in \mathbb{R}^D \\ \sigma:\ \sigma_k \in \mathbb{R}^D_+}}{\operatorname{argmin}} \underbrace{-1 \cdot \sum_{n=1}^{N} \log \mathrm{GMMPDF}(x_n \mid \pi, \mu, \sigma) + \lambda \sum_{k=1}^{K} \sum_{d=1}^{D} \mathrm{penalty}(\sigma_{kd})}_{\mathcal{L}^{PML}(\pi, \mu, \sigma)}$$

Loss function to minimize: $\mathcal{L}^{PML}$

We'll use this in CP3!

Gradient Descent as an algorithm to estimate GMM Parameters

Goal: Find values of π^*, μ^*, Σ^*
that minimize either:
- ML loss $\mathcal{L}^{ML}$
- penalized ML loss $\mathcal{L}^{PML}$

Idea: All parameters have continuous real domains, so let's use <u>gradient</u> <u>descent</u>

Sketch: <u>Input:</u> initial values π^0, μ^0, Σ^0

<u>Procedure:</u> for iter t in $1, 2, \ldots$ until converged:

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

<u>**Problem:**</u> Both π and Σ are <u>constrained</u>

π must sum to one and be non-negative

Σ_k must be a symmetric, pos def matrix

The GD update may produce values that <u>violate</u> constraints

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

Solution: <u>Reparameterize</u>. Find unconstrained parameterization that maps to our constrained version.

Unconstrained Parameterization for a Probability Vector

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 + sum to one.

Unconstrained Parameterization:

Let $s \in \mathbb{R}^K$ be an unconstrained vector of reals

Define transform function $\pi(\cdot) : \mathbb{R}^K \rightarrow \Delta^K$

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

aka <u>softmax</u>

$$Z = \sum_{l=1}^{K} e^{s_l}$$

by definition, output vector is a probality vector

An Unconstrained Parameterization for a Positive Vector

Let σ_k be a vector that must be positive.

$$\sigma_k = [\sigma_{k1} \cdots \sigma_{kD}]$$

$$\sigma_{kd} > 0$$

OPTION 1

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

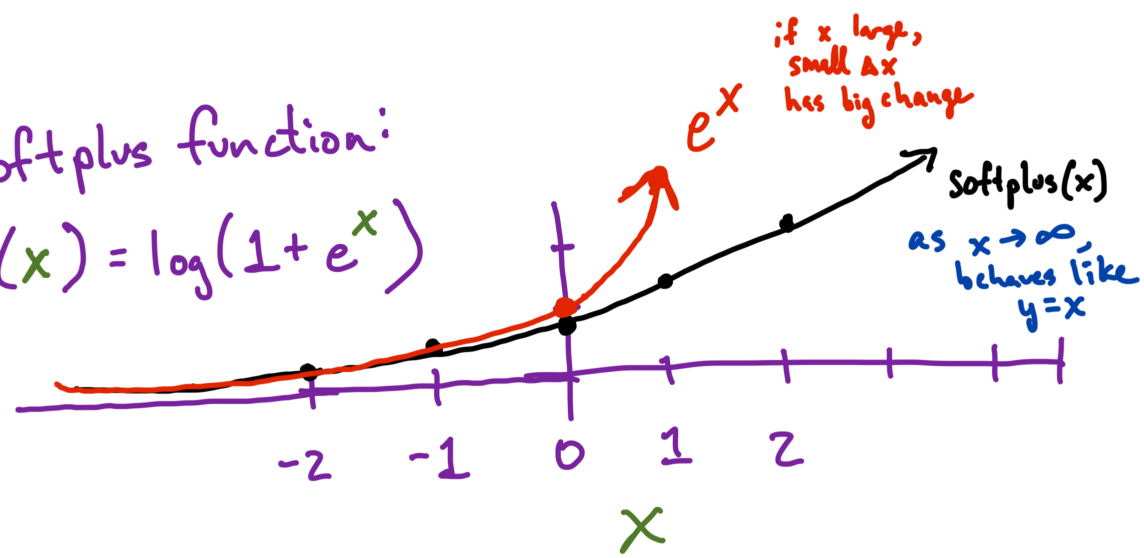
$$\sigma(t_k) \triangleq [\, e^{t_{k1}}, e^{t_{k2}}, \ldots e^{t_{kD}}]$$

by definition, for any input t, $\sigma(t)$ will have all positive entries

OPTION 2

Define softplus function:

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



Give $t_k \in \mathbb{R}^D$, we define transformation $\sigma(\cdot)$:

$$\sigma(t_k) = [\text{softplus}(t_{k1}), \ldots, \text{softplus}(t_{kD})]$$

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

Gradient Descent for GMMs.

Parameters: $s : s_{1:K}, \; s_k \in \mathbb{R}$ all unconstrained real values

$\mu : \mu_{1:K}, \; \mu_k \in \mathbb{R}^D$

$t : t_{1:K}, \; t_k \in \mathbb{R}^D$

Goal: $s^*, \mu^*, t^* = \arg\min \mathcal{L}^{TPML}(s, \mu, t)$

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

R for re-parameterized

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

Grad:
$$\nabla_s \mathcal{L}^{RPML} = \nabla_s \pi(s) \cdot \nabla_\pi \mathcal{L}^{PML}(\pi(s), \cdots)$$
$$\nabla_t \mathcal{L}^{RPML} = \underbrace{\nabla_t \sigma(t) \cdot \nabla_\sigma \mathcal{L}^{PML}(\cdots, \sigma(t))}_{\text{chain rule!}}$$

GD is now possible!

Can do gradients by hand <u>or</u> via automatic differentiation

Keep in Mind:

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

Towards a Coordinate Descent Algorithm to Estimate Parameters

Recall our ML objective (ignore penalty for simplicity here)

$$\pi^*, \mu^*, \Sigma^* = \underset{\substack{\pi \\ \mu \\ \Sigma}\,:\,\text{valid GMM params}}{\arg\min} \underbrace{-1 \cdot \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \, \text{MVNPDF}(x_n | \mu_k, \Sigma_k)}_{\mathcal{L}^{ML}}$$

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

To be optimal, they **must** satisfy these blue eqns.

$$\nabla_{\pi} \mathcal{L}^{ML}(\pi^*, \mu^*, \Sigma^*) = \vec{0}_K \quad \text{all zeros shape } (K,)$$

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

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

Let's expand this out:

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

$$= \sum_{n=1}^{N} \frac{1}{\text{GMMPDF}(x_n)} \cdot \left[\sum_{j=1}^{K} \pi_j \cdot \nabla_{\mu_k} N(x_n | \mu_j, \Sigma_j) \right]$$

$$= \sum_{n=1}^{N} \frac{1}{p(x_n)} \cdot \pi_k N(x_n | \mu_k, \Sigma_k) \cdot \nabla_{\mu_k} \left[-\tfrac{1}{2} (x_n - \mu_k)^T \Sigma_k^{-1} (x_n - \mu_k) \right]$$

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

$$\nabla_{\mu_k} \mathcal{L}^{ML} = \sum_{n=1}^{N} \gamma_{nk} \Sigma_k^{-1} (x_n - \mu_k)$$

Insight: Can simplify using soft "responsibilities" $\gamma_n \in \Delta^K$ fix these, and becomes easy to solve.

Continuing to solve blue equations

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

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

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

Can write all gradients in terms of optimal parameters π^*, μ^*, Σ^* and responsibilities $\gamma_n \in \Delta^K$ (treat as additional variable)

For example, solving for μ gives

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

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

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

multiply both sides by Σ_k

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

(PRML Eq. 9.19)

Interpretation

weighted empirical mean of data assigned to cluster k

Similar derivations yield

$$\pi_k^* = \frac{1}{N} \sum_{n=1}^{N} \gamma_{nk}$$

(PRML 9.22)

fraction of all data points assigned to cluster k

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

(PRML 9.19)

weighted empirical covariance of data assigned to cluster k

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 $t \in 1, 2, \ldots$ $\boxed{\text{until converged}}$:

E step
update responsibilities

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

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

$$\gamma_n \leftarrow \left[\frac{\tilde{\gamma}_{n1}}{\sum_\ell \tilde{\gamma}_{n\ell}}, \frac{\tilde{\gamma}_{n2}}{\sum_\ell \tilde{\gamma}_{n\ell}}, \cdots \frac{\tilde{\gamma}_{nK}}{\sum_\ell \tilde{\gamma}_{n\ell}} \right]$$

M step
update parameters

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

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

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

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

could do with diagonal parametrization
$\Sigma_k = \begin{bmatrix} \sigma_1^2 & & \\ & \ddots & \\ & & \sigma_D^2 \end{bmatrix}$
can adjust to penalty term

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