Latent Variable Models and EM Algorithm

CS772A: Probabilistic Machine Learning

Piyush Rai

Conditional Posterior

- Consider a model with K unknown params/hyperparams $\Theta = (\theta_1, \theta_2, \dots, \theta_K)$

Joint posterior

$$p(\Theta|\boldsymbol{X}) = \frac{p(\Theta)p(\boldsymbol{X}|\Theta)}{p(\boldsymbol{X})} = \frac{p(\Theta)p(\boldsymbol{X}|\Theta)}{\int p(\Theta)p(\boldsymbol{X}|\Theta)d\theta_1 d\theta_2 \dots d\theta_K}$$

Usually intractable integral so the full posterior can't be computed exactly

- We can however compute conditional posteriors (CP) which for each θ_i looks like

$$p(\theta_i|\text{whatever } \theta_i \text{ depends on})$$

Can be data and/or other params/hyperparams given their fixed values (or current estimates)

- To compute each CP, look at the joint distribution $p(\boldsymbol{X}, \Theta)$

$$p(\boldsymbol{X}, \Theta) = p(\boldsymbol{X}, \theta_1, \theta_2, \dots, \theta_K) = p(\boldsymbol{X}|\theta_1, \theta_2, \dots, \theta_K)p(\theta_1|\theta_2, \dots, \theta_K)p(\theta_2|\theta_3, \dots, \theta_K) \dots p(\theta_K)$$

- CP of θ_i will be proportional to the product of all the terms involving θ_i
  - If those terms are conjugate to each other, it is called local conjugacy. CP is then easy to compute
- Many algorithms for computing point estimate/full posterior use the CPs
  - Expectation Maximization, Variational Inference , MCMC (especially Gibbs sampling)

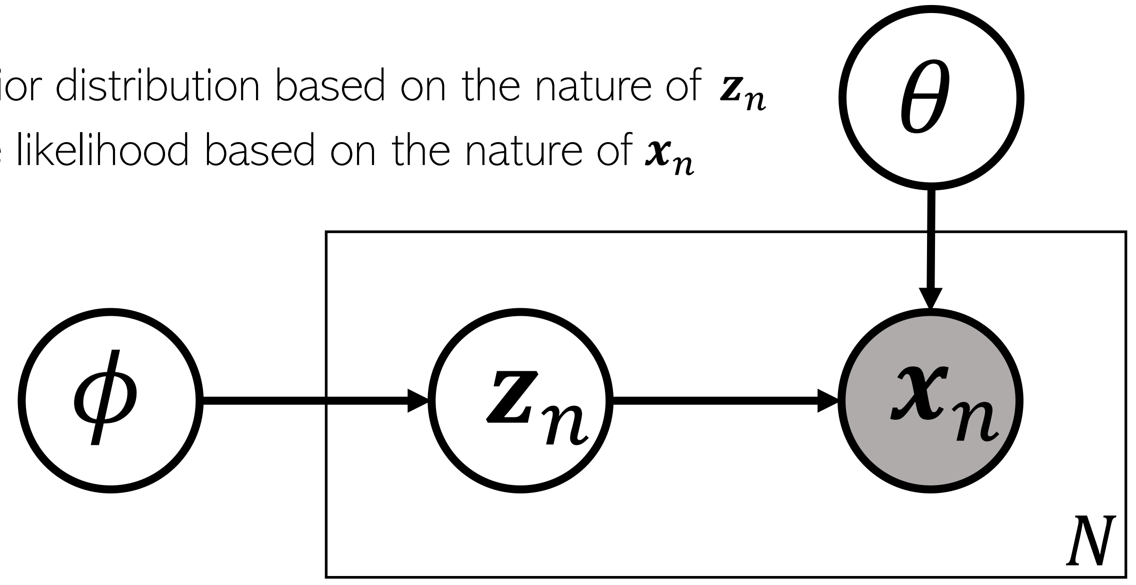
Latent Variable Models

- Application 1: Can use latent variables to learn latent properties/features of data, e.g.,
  - Cluster assignment of each observation (in mixture models)
  - Low-dim rep. or "code" of each observation (e.g., prob. PCA, variational autoencoders, etc)

Plate notation of a generic LVM

$p(\boldsymbol{z}_n|\phi)$: A suitable prior distribution based on the nature of $\boldsymbol{z}_n$

$p(\boldsymbol{x}_n|\boldsymbol{z}_n,\theta)$: A suitable likelihood based on the nature of $\boldsymbol{x}_n$



- In such apps, latent variables ($\boldsymbol{z}_n$'s) are called "local variables" (specific to individual obs.)and other unknown parameters/hyperparams ($\boldsymbol{\theta},\boldsymbol{\phi}$ above) are called "global var"

Latent Variable Models

- Application 2: Sometimes, augmenting a model by latent variables simplifies inference
  - These latent variables aren't part of the original model definition
- Some of the popular examples of such augmentation include
  - In Probit regression for binary classification, we can model each label $y_n \in \{0,1\}$ as

$$y_n = \mathbb{I}[z_n > 0] \qquad \text{where} \qquad z_n \sim \mathcal{N}(\boldsymbol{w}^\top \boldsymbol{x}_n, 1) \;\; \text{is an auxiliary latent variable}$$

.. and use EM etc, to infer the unknowns $\boldsymbol{w}$ and z_n's (PML-2, Sec 15.4)

  - Many sparse priors on weights can be thought of as Gaussian "scale-mixtures"

$$\text{Laplace}(w_d|0, 1/\gamma) = \frac{\gamma}{2}\exp(-\gamma|w_d|) = \int \mathcal{N}(w_d|0, \tau_d^2)\text{Gamma}(\tau_d^2|1, \gamma^2/2)d\tau_d^2$$

.. where $\boldsymbol{\tau}_d$'s are latent vars. Can use EM to infer $\boldsymbol{w}$, $\boldsymbol{\tau}$ (MLAPP 13.4.4 - EM for LASSO)

- Such augmentations can often make a non-conjugate model a locally conjugate one
  - Conditional posteriors of the unknowns often have closed form in such cases

Nomenclature/Notation Alert

- Why call some unknowns as parameters and others as latent variables?
- Well, no specific reason. Sort of a convention adopted by some algorithms
  - EM: Unknowns estimated in E step referred to as latent vars; those in M step as params
  - Usually: Latent vars – (Conditional) posterior computed; parameters – point estimation
- Some algos won't make such distinction and will infer posterior over all unknowns
- Sometimes the "global" or "local" unknown distinction makes it clear
  - Local variables = latent variables, global variables = parameters
- But remember that this nomenclature isn't really cast in stone, no need to be confused so long as you are clear as to what the role of each unknown is, and how we want to estimate it (posterior or point estimate) and using what type of inference algorithm

Hybrid Inference (posterior infer. + point est.)

- In many models, we infer posterior on some unknowns and do point est. for others
- We have already seen MLE-II for lin reg. which alternates between
  - Inferring CP over the main parameter given the point estimates of hyperparams
  - Maximizing the marginal lik. to do point estimation for hyperparams

CP of w: $p(\boldsymbol{w}|\boldsymbol{X},\boldsymbol{y},\hat{\lambda},\hat{\beta})$

$\{\hat{\lambda},\hat{\beta}\} = \text{argmax}_{\lambda,\beta}\, p(\boldsymbol{y}|\boldsymbol{X},\lambda,\beta)$

- The Expectation-Maximization algorithm (will see today) also does something similar
  - In E step, the CP of latent variables is inferred, given current point-est of params
  - M step maximizes expected complete data log-lik. to get point estimates of params
- If we can't (due to computational or other reasons) infer posterior over all unknowns, how to decide which variables to infer posterior on, and for which to do point-est?
- Usual approach: Infer posterior over local vars and point estimates for global vars
  - Reason: We typically have plenty of data to reliably estimate the global variables so it is okay even if we just do point estimation for those

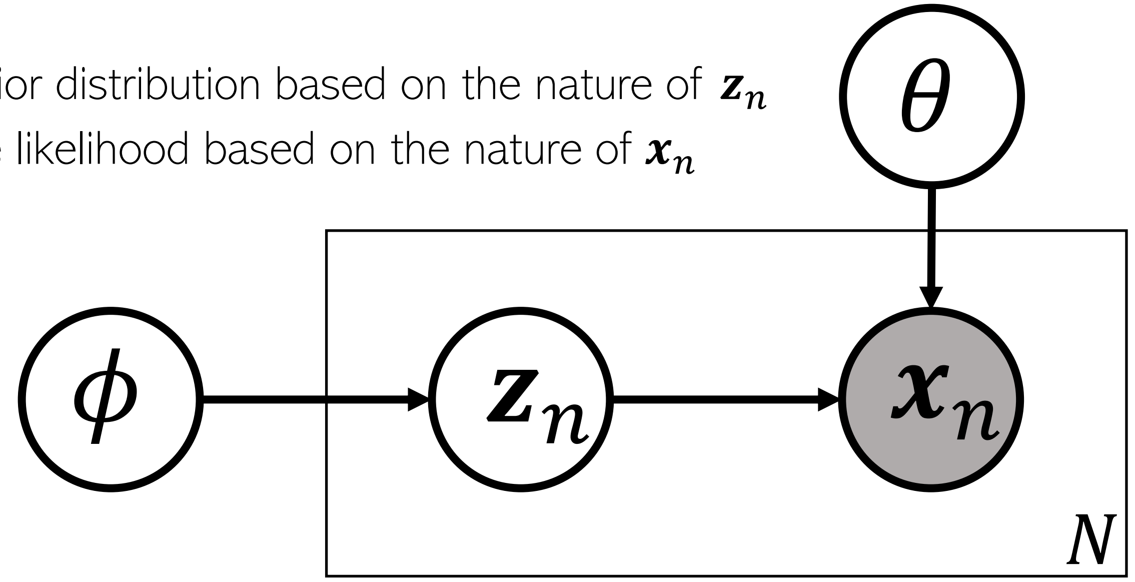
Inference/Parameter Estimation in Latent Variable Models using Expectation-Maximization (EM)

Parameter Estimation in Latent Variable Models

- Assume each observation $\boldsymbol{x}_n$ to be associated with a "local" latent variable $\boldsymbol{z}_n$

$p(\boldsymbol{z}_n|\phi)$: A suitable prior distribution based on the nature of $\boldsymbol{z}_n$

$p(\boldsymbol{x}_n|\boldsymbol{z}_n,\theta)$: A suitable likelihood based on the nature of $\boldsymbol{x}_n$



- Although we can do fully Bayesian inference for all the unknowns, suppose we only want a point estimate of the "global" parameters $\Theta = (\theta, \phi)$ via MLE/MAP
- Such MLE/MAP problems in LVMs are difficult to solve in a "clean" way
  - Would typically require gradient based methods with no closed form updates for Θ
  - However, EM gives a clean way to obtain closed form updates for Θ

Why MLE/MAP of Params is Hard for LVMs?

- Suppose we want to estimate $\Theta = (\theta, \phi)$ via MLE. If we knew z_n, we could solve

$$\Theta_{MLE} = \arg\max_{\Theta} \sum_{n=1}^{N} \log p(\mathbf{x}_n, \mathbf{z}_n|\Theta) = \arg\max_{\Theta} \sum_{n=1}^{N} [\log p(\mathbf{z}_n|\phi) + \log p(\mathbf{x}_n|\mathbf{z}_n, \theta)]$$

Easy to solve

- Easy. Usually closed form if $p(z_n|\phi)$ and $p(x_n|z_n, \theta)$ have simple forms

In particular, if they are exp-fam distributions

- However, since in LVMs, z_n is hidden, the MLE problem for Θ will be the following

Basically, the marginal likelihood after integrating out z_n

$$\Theta_{MLE} = \arg\max_{\Theta} \sum_{n=1}^{N} \log p(\mathbf{x}_n|\Theta) = \arg\max_{\Theta} \log p(\mathbf{X}|\Theta)$$

- $\log p(x_n|\Theta)$ will not have a simple expression since $p(x_n|\Theta)$ requires sum/integral

$$p(\mathbf{x}_n|\Theta) = \sum_{\mathbf{z}_n} p(\mathbf{x}_n, \mathbf{z}_n|\Theta) \quad \text{... or if } \mathbf{z}_n \text{ is continuous:} \quad p(\mathbf{x}_n|\Theta) = \int p(\mathbf{x}_n, \mathbf{z}_n|\Theta) d\mathbf{z}_n$$

- MLE now becomes difficult (basically MLE-II now), no closed form expression for Θ.
- Can we maximize some other quantity instead of $\log p(x_n|\Theta)$ for this MLE?

An Important Identity

- Assume $p_z = p(\boldsymbol{Z}|\boldsymbol{X},\Theta)$ and $q(\boldsymbol{Z})$ to be some prob distribution over $\boldsymbol{Z}$, then

$$\log p(\boldsymbol{X}|\Theta) = \mathcal{L}(q,\Theta) + KL(q||p_z)$$

Verify the identity

Assume $\boldsymbol{Z}$ discrete

- In the above $\mathcal{L}(q,\Theta) = \sum_Z q(Z)\log\left\{\frac{p(X,Z|\Theta)}{q(Z)}\right\}$

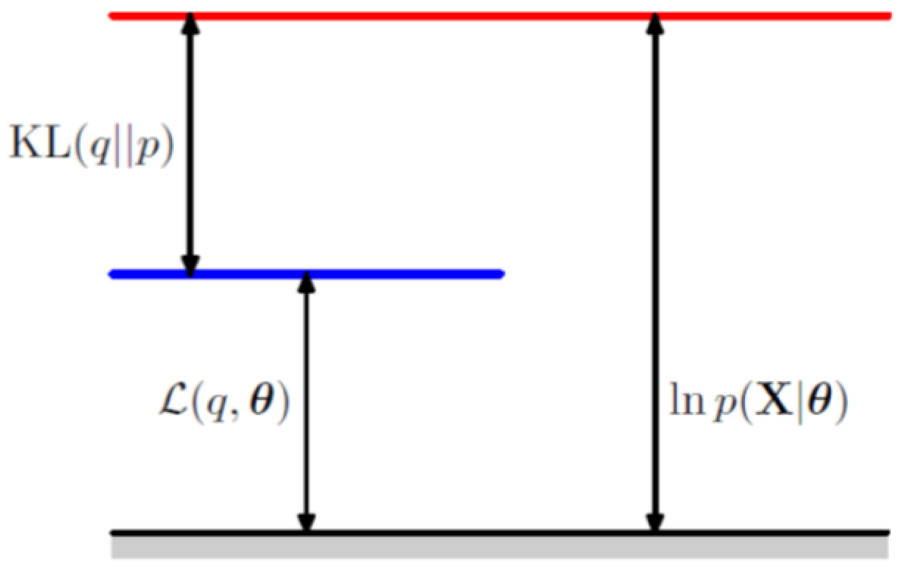
- $KL(q||p_z) = -\sum_Z q(\boldsymbol{Z})\log\left\{\frac{p(\boldsymbol{Z}|\boldsymbol{X},\Theta)}{q(\boldsymbol{Z})}\right\}$

- KL is always non-negative, so $\log p(\boldsymbol{X}|\Theta) \geq \mathcal{L}(q,\Theta)$

- Thus $\mathcal{L}(q,\Theta)$ is a lower-bound on $\log p(\boldsymbol{X}|\Theta)$

- Thus if we maximize $\mathcal{L}(q,\Theta)$, it will also improve $\log p(\boldsymbol{X}|\Theta)$

- Also, as we'll see, it's easier to maximize $\mathcal{L}(q,\Theta)$



Maximizing $\mathcal{L}(q, \Theta)$

Basically, log of marginal likelihood w.r.t. $\boldsymbol{\Theta}$ with $\boldsymbol{Z}$ integrated out

$\log p(\boldsymbol{X}|\Theta)$ is called Incomplete-Data Log Likelihood (ILL)

- $\mathcal{L}(q, \Theta)$ depends on q and Θ. We'll use ALT-OPT to maximize it
- Let's maximize $\mathcal{L}(q, \Theta)$ w.r.t. q with Θ fixed at some Θ^{old}

Since $\log p(\boldsymbol{X}|\Theta) = \mathcal{L}(q, \Theta) + KL(q||p_z)$ is constant when Θ is held fixed at Θ^{old}

$$\hat{q} = \text{argmax}_q \mathcal{L}(q, \Theta^{\text{old}}) = \text{argmin}_q KL(q||p_z) = p_z = p(\boldsymbol{Z}|\boldsymbol{X}, \Theta^{\text{old}})$$

The posterior distribution of Z given current parameters Θ^{old}

- Now let's maximize $\mathcal{L}(q, \Theta)$ w.r.t. Θ with q fixed at $\hat{q} = p_z = p(\boldsymbol{Z}|\boldsymbol{X}, \Theta^{\text{old}})$

$$\Theta^{\text{new}} = \text{argmax}_\Theta \mathcal{L}(\hat{q}, \Theta) = \text{argmax}_\Theta \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}, \Theta^{\text{old}}) \log \left\{ \frac{p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)}{p(\boldsymbol{Z}|\boldsymbol{X}, \Theta^{\text{old}})} \right\}$$

$$= \text{argmax}_\Theta \sum_{\boldsymbol{Z}} p(\boldsymbol{Z}|\boldsymbol{X}, \Theta^{\text{old}}) \log p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)$$

$$= \text{argmax}_\Theta \mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X}, \Theta^{\text{old}})}[\log p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)]$$

$$= \text{argmax}_\Theta \mathcal{Q}(\Theta, \Theta^{\text{old}})$$

Complete-Data Log Likelihood (CLL)

Maximization of expected CLL where the expectation is w.r.t. the posterior distribution of Z given current parameters Θ^{old}

Much easier than maximizing ILL since CLL will have simple expressions (since it is akin to knowing Z)

The Expectation-Maximization (EM) Algorithm

- ALT-OPT of $\mathcal{L}(q,\Theta)$ w.r.t. q and Θ gives the EM algorithm (Dempster, Laird, Rubin, 1977)

The EM Algorithm

Primarily designed for doing point estimation of the parameters Θ but also gives (CP of) latent variables z_n

Usually computing CP + expected CLL is referred to as the E step, and max. of exp-CLL w.r.t. Θ as the M step

1. Initialize Θ as $\Theta^{(0)}$, set $t = 1$
2. Step 1: Compute posterior of latent variables given current parameters $\Theta^{(t-1)}$

Conditional posterior of each latent variable z_n

Latent variables also assumed indep. a priori

$$p(\boldsymbol{z}_n^{(t)}|\boldsymbol{x}_n,\Theta^{(t-1)}) = \frac{p(\boldsymbol{z}_n^{(t)}|\Theta^{(t-1)})p(\boldsymbol{x}_n|\boldsymbol{z}_n^{(t)},\Theta^{(t-1)})}{p(\boldsymbol{x}_n|\Theta^{(t-1)})} \propto \text{prior} \times \text{likelihood}$$

3. Step 2: Now maximize the expected complete data log-likelihood w.r.t. Θ

Assuming the (expected) CLL $\mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X},\Theta^{\text{old}})}[\log p(\boldsymbol{X},\boldsymbol{Z}|\Theta)]$ factorizes over all observations

$$\Theta^{(t)} = \arg\max_{\Theta} \mathcal{Q}(\Theta,\Theta^{(t-1)}) = \arg\max_{\Theta} \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{z}_n^{(t)}|\boldsymbol{x}_n,\Theta^{(t-1)})}[\log p(\boldsymbol{x}_n,\boldsymbol{z}_n^{(t)}|\Theta)]$$

4. If not yet converged, set $t = t+1$ and go to step 2.

- Note: If we can take the MAP estimate $\hat{z}_n$ of z_n (not full posterior) in Step 1 and maximize the CLL in Step 2 using that, i.e., do $\text{argmax}_{\Theta} \sum_{n=1}^{N} \left[\log p\left(\boldsymbol{x}_n, \hat{z}_n^{(t)}\middle|\Theta\right)\right]$ this will be ALT-OPT

The Expected CLL

- Expected CLL in EM is given by (assume observations are i.i.d.)

$$\begin{aligned}\mathcal{Q}(\Theta, \Theta^{old}) &= \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n,\Theta^{old})}[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\Theta)] \\ &= \sum_{n=1}^{N} \mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n,\Theta^{old})}[\log p(\boldsymbol{x}_n|\boldsymbol{z}_n, \Theta) + \log p(\boldsymbol{z}_n|\Theta)]\end{aligned}$$

- If $p(\boldsymbol{z}_n|\Theta)$ and $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \Theta)$ are exp-family distributions, $\mathcal{Q}(\Theta, \Theta^{\text{old}})$ has a very simple form
- In resulting expressions, replace terms containing $\boldsymbol{z}_n$'s by their respective expectations, e.g.,
  - $\boldsymbol{z}_n$ replaced by $\mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n, \hat{\Theta})}[\boldsymbol{z}_n]$
  - $\boldsymbol{z}_n\boldsymbol{z}_n^\top$ replaced by $\mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n, \hat{\Theta})}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]$
- However, in some LVMs, these expectations are intractable to compute and need to be approximated (will see some examples later)

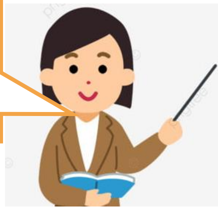
What's Going On?

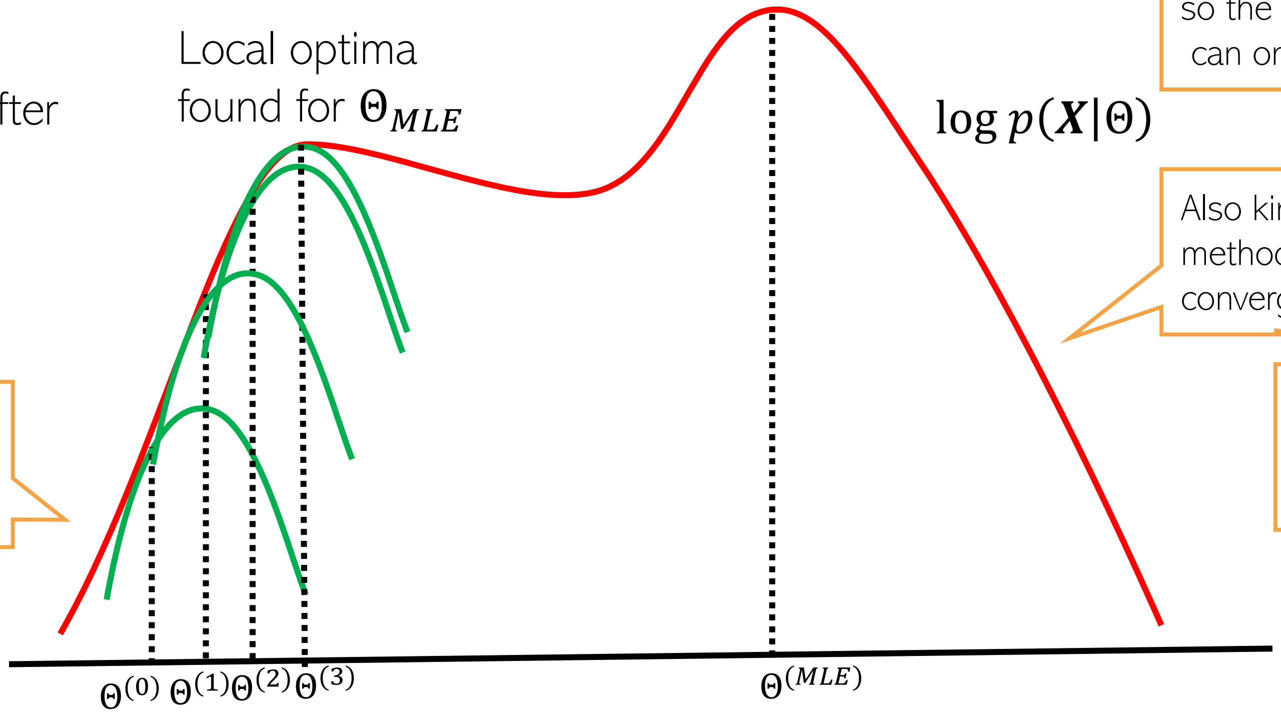
Alternating between them until convergence to some local optima

KL becomes zero and $\mathcal{L}(q, \Theta)$ becomes equal to $\log p(\boldsymbol{X}|\Theta)$; thus their curves touch at current Θ

- As we saw, the maximization of lower bound $\mathcal{L}(q, \Theta)$ had two steps
- Step 1 finds the optimal q (call it $\hat{q}$) by setting it as the posterior of $\boldsymbol{Z}$ given current Θ
- Step 2 maximizes $\mathcal{L}(\hat{q}, \Theta)$ w.r.t. Θ which gives a new Θ.

Note that Θ only changes in Step 2 so the objective $\log p(\boldsymbol{X}|\Theta)$ can only change in Step 2





Local optima found for Θ_{MLE}
Green curve: $\mathcal{L}(\hat{q}, \Theta)$ after setting q to $\hat{q}$
$\log p(\boldsymbol{X}|\Theta)$
$\Theta^{(0)}$ $\Theta^{(1)}$ $\Theta^{(2)}$ $\Theta^{(3)}$
$\Theta^{(MLE)}$

Good initialization matters; otherwise would converge to a poor local optima

Also kind of similar to Newton's method (and has second order like convergence behavior in some cases)

Unlike Newton's method, we don't construct and optimize a quadratic approximation, but a lower bound

Even though original MLE problem $\text{argmax}_{\Theta} \log p(\boldsymbol{X}|\Theta)$ could be solved using gradient methods, EM often works faster and has cleaner updates

EM vs Gradient-based Methods

- Can also estimate params using gradient-based optimization instead of EM
  - We can usually explicitly sum over or integrate out the latent variables $\boldsymbol{Z}$, e.g.,

$$\mathcal{L}(\Theta) = \log p(\mathbf{X}|\Theta) = \log \sum_{\mathbf{Z}} p(\mathbf{X}, \mathbf{Z}|\Theta)$$

  - Now we can optimize $\mathcal{L}(\Theta)$ using first/second order optimization to find the optimal Θ

- EM is usually preferred over this approach because
  - The M step has often simple closed-form updates for the parameters Θ
  - Often constraints (e.g., PSD matrices) are automatically satisfied due to form of updates
  - In some cases[†], EM usually converges faster (and often like second-order methods)
    - E.g., Example: Mixture of Gaussians with when the data is reasonably well-clustered
  - EM applies even when the explicit summing over/integrating out is expensive/intractable
  - EM also provides the conditional posterior over the latent variables Z (from E step)

[†]Optimization with EM and Expectation-Conjugate-Gradient (Salakhutdinov et al, 2003), On Convergence Properties of the EM Algorithm for Gaussian Mixtures (Xu and Jordan, 1996), Statistical guarantees for the EM algorithm: From population to sample-based analysis (Balakrishnan et al, 2017)

Some Applications of EM

- Mixture Models and Dimensionality Reduction/Representation Learning
  - Mixture Models: Mixture of Gaussians, Mixture of Experts, etc
  - Dim. Reduction/Representation Learning: Probabilistic PCA, Variational Autoencoders
- Problems with missing features or missing labels (which are treated as latent variables)
  - $\widehat{\Theta} = \text{argmax}_\Theta \log p(\boldsymbol{x}^{obs}|\Theta) = \text{argmax}_\Theta \log \int p([\boldsymbol{x}^{obs}, \boldsymbol{x}^{miss}]|\Theta) d\boldsymbol{x}^{miss}$
  - $\widehat{\Theta} = \text{argmax}_\Theta \sum_{n=1}^{N} \log p(x_n, y_n|\Theta) + \sum_{n=N+1}^{N+M} \log \sum_{c=1}^{K} p(x_n, y_n = c|\Theta)$
- Hyperparameter estimation in probabilistic models (an alternative to MLE-II)
  - MLE-II estimates hyperparams by maximizing the marginal likelihood, e.g.,

$$\{\hat{\lambda}, \hat{\beta}\} = \text{argmax}_{\lambda,\beta}\, p(\boldsymbol{y}|\boldsymbol{X}, \lambda, \beta) = \text{argmax}_{\lambda,\beta} \int p(\boldsymbol{y}|\boldsymbol{w}, \boldsymbol{X}, \beta) p(\boldsymbol{w}|\lambda) d\boldsymbol{w}$$

For a Bayesian linear regression model

  - With EM, can treat $\boldsymbol{w}$ as latent var, and $\boldsymbol{\lambda}, \boldsymbol{\beta}$ as "parameters"
    - E step will estimate the CP of $\boldsymbol{w}$ given current estimates of $\boldsymbol{\lambda}, \boldsymbol{\beta}$
    - M step will re-estimate $\boldsymbol{\lambda}, \boldsymbol{\beta}$ by maximizing the expected CLL

$$\mathbb{E}[\log p(\boldsymbol{y}, \boldsymbol{w}|\mathbf{X}, \beta, \lambda)] = \mathbb{E}[\log p(\boldsymbol{y}|\boldsymbol{w}, \mathbf{X}, \beta) + \log p(\boldsymbol{w}|\lambda)]$$

Expectations w.r.t. the CP of $\boldsymbol{w}$

An Example: Mixture Models

If the $\boldsymbol{z}_n$ were known, it just becomes **generative classification**, for which which we know how to estimate $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$, given training data

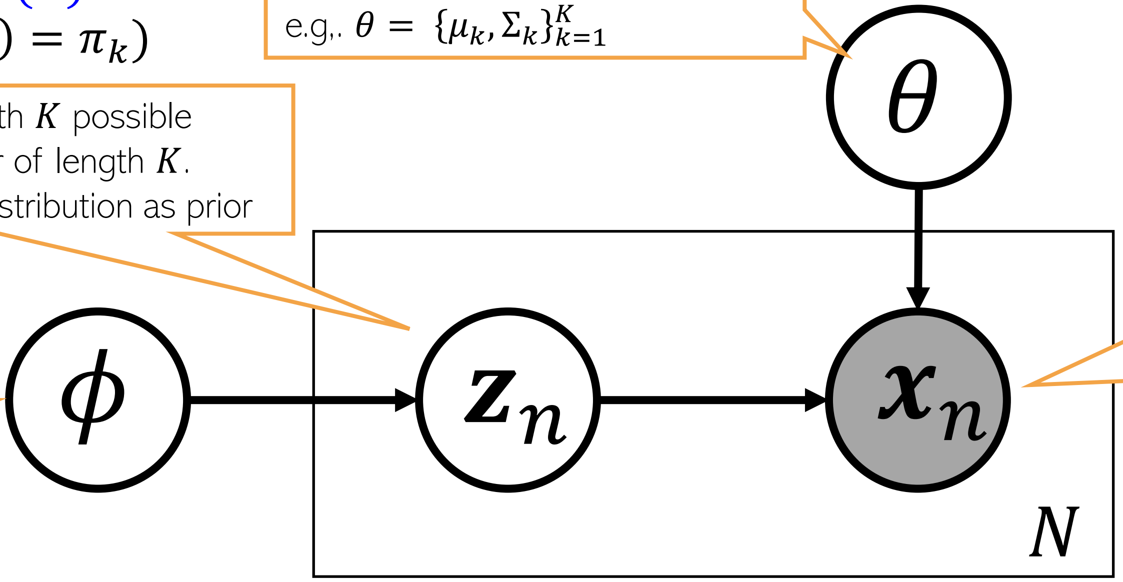
- Assume K probability distributions (e.g., Gaussians), one for each cluster

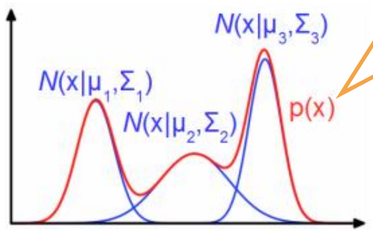
$p(z_n \mid \phi) = \text{multinoulli}(\boldsymbol{\pi})$
(also means $p(z_n = k \mid \phi) = \pi_k$)

Parameters of the K distributions, e.g,. $\theta = \{\mu_k, \Sigma_k\}_{k=1}^{K}$

Discrete latent variable (with K possible values) or a one-hot vector of length K. Modeled by a multinoulli distribution as prior

The parameter vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_K]$ of the multinoulli distribution





$p(x)$ is a Gaussian mixture model (GMM)

Assumed generated from one of the K distributions depending on the true (but unknown) value of z_n (which clustering will find))

The likelihood distributions

$p(\boldsymbol{x}_n \mid \boldsymbol{z}_n = k, \theta) = \mathcal{N}(\mu_k, \Sigma_k)$

- The log-likelihood will be

$$\log p(\boldsymbol{x}_n|\Theta) = \log \sum_{k=1}^{K} p(\boldsymbol{x}_n, \boldsymbol{z}_n = k|\Theta)$$

$$= \log \sum_{k=1}^{K} p(\boldsymbol{z}_n = k|\phi) p(\boldsymbol{x}_n|\boldsymbol{z}_n = k, \theta) = \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\mu_k, \Sigma_k)$$

MLE on this objective won't give closed form solution for the parameters

Detour: MLE for Generative Classification

- Assume a K class generative classification model with Gaussian class-conditionals
- Assume class $k = 1,2,\dots,K$ is modeled by a Gaussian with mean μ_k and cov matrix Σ_k
- The labels $\boldsymbol{z}_n$ (known) are one-hot vecs. Also, $z_{nk} = 1$ if $\boldsymbol{z}_n = k$, and $\boldsymbol{z}_{nk} = \mathbf{0}$, o/w
- Assuming class prior as $p(\boldsymbol{z}_n = k) = \pi_k$, the model has params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$
- Given training data $\{\boldsymbol{x}_n, \boldsymbol{z}_n\}_{n=1}^{N}$, the MLE solution will be

$$\hat{\pi}_k = \frac{1}{N}\sum_{n=1}^{N} z_{nk}$$

Same as $\frac{N_k}{N}$ where N_k is # of training ex. for which $y_n = k$

$$\hat{\mu}_k = \frac{1}{N_k}\sum_{n=1}^{N} z_{nk}\boldsymbol{x}_n$$

Same as $\frac{1}{N_k}\sum_{n:\boldsymbol{z}_n=k}^{N} \boldsymbol{x}_n$

$$\hat{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^{N} z_{nk}(\boldsymbol{x}_n - \hat{\mu}_k)(\boldsymbol{x}_n - \hat{\mu}_k)^\top$$

Same as $\frac{1}{N_k}\sum_{n:\boldsymbol{z}_n=k}^{N}(\boldsymbol{x}_n - \hat{\mu}_k)(\boldsymbol{x}_n - \hat{\mu}_k)^\top$

Detour: MLE for Generative Classification

- Here is a formal derivation of the MLE solution for $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$

$$\hat{\Theta} = \text{argmax}_\Theta\, p(\boldsymbol{X}, \boldsymbol{Z}|\Theta) = \text{argmax}_\Theta \prod_{n=1}^{N} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\Theta)$$

multinoulli

Gaussian

$$= \text{argmax}_\Theta \prod_{n=1}^{N} p(\boldsymbol{z}_n|\Theta)\, p(x_n|\boldsymbol{z}_n, \Theta)$$

$$= \text{argmax}_\Theta \prod_{n=1}^{N} \prod_{k=1}^{K} \pi_k^{z_{nk}} \prod_{k=1}^{K} p(x_n|\boldsymbol{z}_n = k, \Theta)^{z_{nk}}$$

$$= \text{argmax}_\Theta \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k p(x_n|\boldsymbol{z}_n = k, \Theta)]^{z_{nk}}$$

$$= \text{argmax}_\Theta \log \prod_{n=1}^{N} \prod_{k=1}^{K} [\pi_k p(x_n|\boldsymbol{z}_n = k, \Theta)]^{z_{nk}}$$

$$= \text{argmax}_\Theta \sum_{n=1}^{N} \sum_{k=1}^{K} z_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\mu_k, \Sigma_k)]$$

In general, in models with probability distributions from the exponential family, the MLE problem will usually have a simple analytic form

Also, due to the form of the likelihood (Gaussian) and prior (multinoulli), the MLE problem had a nice separable structure after taking the log

Can see that, when estimating the parameters of the k^{th} Gaussian (π_k, μ_k, Σ_k), we only will only need training examples from the k^{th} class, i.e., examples for which $z_{nk} = 1$

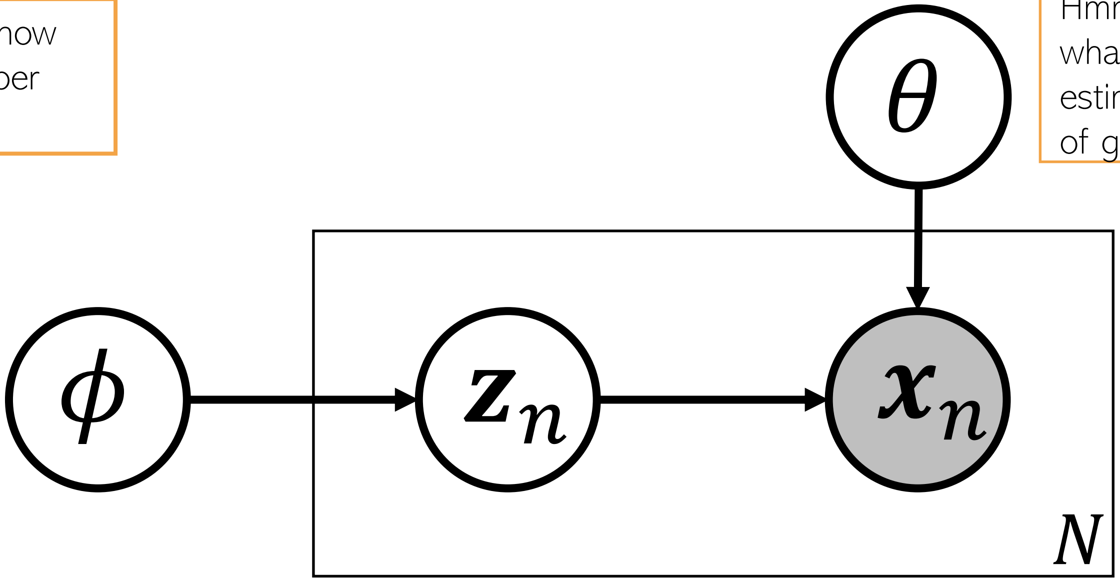
The form of this expression is important; will encounter this in GMM too

EM for Mixture Models

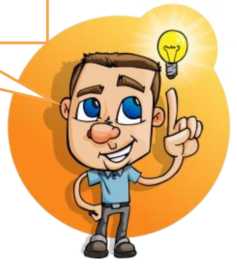
- So how do we estimate the parameters of a GMM where $\boldsymbol{z}_n$'s are <u>unknown</u>?

Well, you kind of already know how to do this. ☺ Remember generative classification?

Yes, exactly. ☺ However, just like in gen-class, you will need to repeat the guess and estimate them a few times until you converge



Hmmmm.. So can we make a guess what the value of each $\boldsymbol{z}_n$ and then estimate $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ as we do in case of generative classification??



- The guess about $\boldsymbol{z}_n$ can be in one of the two forms
  - A "hard" guess – a single best value $\hat{\boldsymbol{z}}_n$ (some "optimal" value of the random variable $\boldsymbol{z}_n$)
  - The "expected" value $\mathbb{E}[\boldsymbol{z}_n]$ of the random variable $\boldsymbol{z}_n$
- Using the hard guess $\hat{\boldsymbol{z}}_n$ of $\boldsymbol{z}_n$ will result in an ALT-OPT like algorithm
- Using the expected value of $\boldsymbol{z}_n$ will give the so-called Expectation-Maximization (EM) algo

EM is pretty much like ALT-OPT but with soft/expected values of the latent variables

EM for Gaussian Mixture Model (GMM)

Expectation of CLL

- EM finds Θ_{MLE} by maximizing $\mathbb{E}[\log p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)]$ rather than $\log p(\boldsymbol{X}, \hat{\boldsymbol{Z}}|\Theta)$
- Note: Expectation will be w.r.t. the conditional posterior distribution of $\boldsymbol{Z}$, i.e., $p(\boldsymbol{Z}|\boldsymbol{X}, \Theta)$

It is "conditional" posterior because it is also conditioned on Θ, not just data X

Requires knowing Θ

- The EM algorithm for GMM operates as follows
  - Initialize $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ as $\hat{\Theta}$
  - Repeat until convergence

Needed to get the expected CLL

    - Compute conditional posterior $p(\boldsymbol{Z}|\boldsymbol{X}, \hat{\Theta})$. Since obs are i.i.d, compute separately for each n (and for $k = 1,2,..K$)

Same as $p(z_{nk} = 1|\boldsymbol{x}_n, \hat{\Theta})$, just a different notation

$$p(\boldsymbol{z}_n = k|\boldsymbol{x}_n, \hat{\Theta}) \propto p(\boldsymbol{z}_n = k|\hat{\Theta})\, p(\boldsymbol{x}_n|\boldsymbol{z}_n = k, \hat{\Theta}) = \hat{\pi}_k \mathcal{N}(x_n|\hat{\mu}_k, \hat{\Sigma}_k)$$

    - Update Θ by maximizing the expected complete data log-likelihood

Solution has a similar form as ALT-OPT (or gen. class.), except we now have the expectation of z_{nk} being used

$$\hat{\Theta} = \operatorname{argmax}_\Theta \mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X},\hat{\Theta})}[\log p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)] = \sum_{n=1}^N \mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n,\hat{\Theta})}[\log p(\boldsymbol{x}_n, \boldsymbol{z}_n|\Theta)]$$

$$= \operatorname{argmax}_\Theta \mathbb{E}\left[\sum_{n=1}^N \sum_{k=1}^K z_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\mu_k, \Sigma_k)]\right]$$

$$= \operatorname{argmax}_\Theta \sum_{n=1}^N \sum_{k=1}^K \mathbb{E}[z_{nk}][\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\mu_k, \Sigma_k)]$$

$$\hat{\pi}_k = \frac{1}{N}\sum_{n=1}^N \mathbb{E}[z_{nk}] \quad \hat{\mu}_k = \frac{1}{N_k}\sum_{n=1}^N \mathbb{E}[z_{nk}]\boldsymbol{x}_n$$

N_k : Effective number of points in cluster k

$$\hat{\Sigma}_k = \frac{1}{N_k}\sum_{n=1}^N \mathbb{E}[z_{nk}](\boldsymbol{x}_n - \hat{\mu}_k)(\boldsymbol{x}_n - \hat{\mu}_k)^\top$$

EM for GMM (Contd)

Reason: $\sum_{k=1}^{K} \gamma_{nk} = 1$

- The EM algo for GMM required $\mathbb{E}[z_{nk}]$. Note $z_{nk} \in \{0,1\}$

Need to normalize: $\mathbb{E}[z_{nk}] = \frac{\hat{\pi}_k \mathcal{N}(x_n|\hat{\mu}_k,\hat{\Sigma}_k)}{\sum_{\ell=1}^{K} \hat{\pi}_\ell \mathcal{N}(x_n|\hat{\mu}_\ell,\hat{\Sigma}_\ell)}$

$$\mathbb{E}[z_{nk}] = \gamma_{nk} = 0 \times p(z_{nk}=0|x_n,\widehat{\Theta}) + 1 \times p(z_{nk}=1|x_n,\widehat{\Theta}) = p(z_{nk}=1|x_n,\widehat{\Theta}) \propto \hat{\pi}_k \mathcal{N}(x_n|\hat{\mu}_k,\hat{\Sigma}_k)$$

EM for Gaussian Mixture Model

1. Initialize $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ as $\Theta^{(0)}$, set $t=1$
2. E step: compute the expectation of each $\boldsymbol{z}_n$ (we need it in M step)

Accounts for fraction of points in each cluster

Accounts for cluster shapes (since each cluster is a Gaussian

Soft K-means, which is more of a heuristic to get soft-clustering, also gave us probabilities but doesn't account for cluster shapes or fraction of points in each cluster

$$\mathbb{E}[z_{nk}^{(t)}] = \gamma_{nk}^{(t)} = \frac{\pi_k^{(t-1)} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_k^{(t-1)}, \boldsymbol{\Sigma}_k^{(t-1)})}{\sum_{\ell=1}^{K} \pi_\ell^{(t-1)} \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu}_\ell^{(t-1)}, \boldsymbol{\Sigma}_\ell^{(t-1)})} \quad \forall n, k$$

3. Given "responsibilities" $\gamma_{nk} = \mathbb{E}[z_{nk}]$, and $N_k = \sum_{n=1}^{N} \gamma_{nk}$, re-estimate Θ via MLE

Effective number of points in the k^{th} cluster

M-step:

$$\boldsymbol{\mu}_k^{(t)} = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}^{(t)} \boldsymbol{x}_n$$

$$\boldsymbol{\Sigma}_k^{(t)} = \frac{1}{N_k}\sum_{n=1}^{N} \gamma_{nk}^{(t)} (\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})(\boldsymbol{x}_n - \boldsymbol{\mu}_k^{(t)})^\top$$

$$\pi_k^{(t)} = \frac{N_k}{N}$$

4. Set $t = t+1$ and go to step 2 if not yet converged

EM: Some Final Comments

- The E and M steps may not always be possible to perform exactly. Some reasons
  - The conditional posterior of latent variables $p(Z|X,\Theta)$ may not be easy to compute
    - Will need to approximate $p(Z|X,\Theta)$ using methods such as MCMC or variational inference
  - Even if $p(Z|X,\Theta)$ is easy, the expected CLL may not be easy to compute

Results in Monte-Carlo EM

$$\mathbb{E}[\log p(\mathbf{X},\mathbf{Z}|\Theta)] = \int \log p(\mathbf{X},\mathbf{Z}|\Theta)p(\mathbf{Z}|\mathbf{X},\Theta)d\mathbf{Z}$$

Can often be approximated by Monte-Carlo using sample from the CP of Z

  - Maximization of the expected CLL may not be possible in closed form
- EM works even if the M step is only solved approximately (Generalized EM)
- If M step has multiple parameters whose updates depend on each other, they are updated in an alternating fashion - called Expectation Conditional Maximization (ECM)
- Other advanced probabilistic inference algos are based on ideas similar to EM
  - E.g., Variational EM, Variational Bayes (VB) inference, a.k.a. Variational Inference (VI)