Latent Variable Models and the Expectation Maximization Algorithm

CS772A: Probabilistic Machine Learning

Piyush Rai

Plan

- Wrap-up the discussion of CP and local conjugacy
  - Gibbs sampling (more on this when we discuss MCMC)
- Latent Variable Models (LVM)
  - The basic formulation of LVMs (specific models later)
  - Parameter Estimation in LVMs
- Expectation Maximization algorithm for param-est/inference in LVMs

Gibbs Sampling (Geman and Geman, 1982)

- A general algo to generate samples from multivar. distr. <u>one variable at a time</u>
  - Not limited to sampling from intractable posteriors only
  - Sometimes can be used even if we can draw from the multivar distr. directly

Note: If posterior, it will be conditioned on other stuff too (e.g., data, other param, etc)

- Assume we want to sample from a joint distribution $p(\theta_1, \theta_2, \ldots, \theta_K)$
- It generates one component θ_k at a time using its conditional $p(\theta_k | \Theta_{-k})$
- Each conditional is assumed to be available in closed form. An example below:

Suppose

A 2-dim Gaussian

$$\theta \sim N_2(0, \Sigma) \qquad \Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

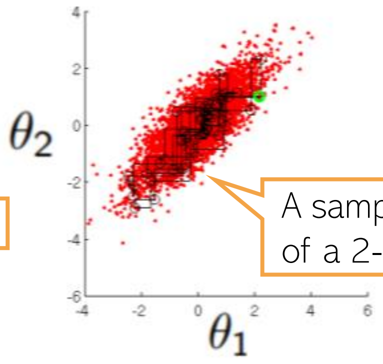
Then

$$\begin{aligned} \theta_1 | \theta_2 &\sim N\left(\rho\theta_2, [1-\rho^2]\right) \\ \theta_2 | \theta_1 &\sim N\left(\rho\theta_1, [1-\rho^2]\right) \end{aligned}$$

A 1-dim Gaussian

A 1-dim Gaussian

are the conditional distributions.



A sample based representation of a 2-dim Gaussian distribution

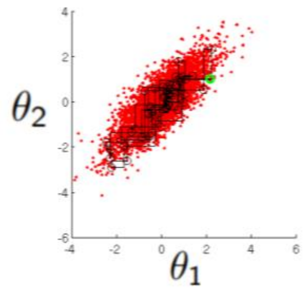
This example is a toy illustration of sampling from a 2-dim Gaussian by sampling from 1-dim Gaussians

Gibbs Sampling (Geman and Geman, 1982)

- Can be used to get a sampling-based approx. of a multi-param. posterior
- Iteratively draws random samples from the CPs in a cyclic order
- When run long enough, the sampler produces samples from the joint posterior
- For the simple two-param case $\theta = (\theta_1, \theta_2)$, the Gibb sampler looks like this
  - Initialize $\theta_2^{(0)}$
  - For $s = 1,2, \dots, S$
    - Draw a random sample for θ_1 as $\theta_1^{(s)} \sim p(\theta_1 | X, \theta_2^{(s-1)})$
    - Draw a random sample for θ_2 as $\theta_2^{(s)} \sim p(\theta_2 | X, \theta_1^{(s)})$

This CP uses the most recent value of θ_2

This CP uses the most recent value of θ_1



- These S random samples $\left(\theta_1^{(s)}, \theta_1^{(s)}\right)_{s=1}^{S}$ represent joint posterior $p(\theta_1, \theta_2 | X)$
- This is just a high-level idea. More on this when we discuss MCMC

Back to Bayesian Matrix Factorization

Bayesian Matrix Factorization: The CPs

- BMF with Gaussian likelihood and Gaussian prior on each user/item params is not fully conjugate but has local conjugacy
- To see this, note that the conditional posterior (CP) for user parameter $\boldsymbol{u_i}$

$$p(\boldsymbol{u}_i|\mathbf{R},\mathbf{V},\mathbf{U}_{-i}) \propto \prod_{j:(i,j)\in\Omega} p(r_{ij}|\boldsymbol{u}_i,\mathbf{v}_j)p(\boldsymbol{u}_i)$$

$$= \prod_{j:(i,j)\in\Omega} \mathcal{N}(r_{ij}|\boldsymbol{u}_i^\top \mathbf{v}_j, \beta^{-1})\mathcal{N}(\boldsymbol{u}_i|\mathbf{0},\lambda_u^{-1}\mathbf{I}_K)$$

Only depends on the ratings of user $\boldsymbol{i}$. Also doesn't depend on $\mathbf{U}_{-i}$

This is due to the structure of the problem and the $\boldsymbol{u}_i$'s being independent a priori

- The above is just like Bayesian linear regression, given $\boldsymbol{R}$ and fixed $\mathbf{V}$
  - Weight vector is $\boldsymbol{u}_i$, training data is $\{(\boldsymbol{v}_j, r_{ij})\}_{j:(i,j)\in\Omega}$, given
  - Also have local conjugacy since likelihood and prior are conjugate (assuming hyperparams fixed)
- Likewise, the CP for the item parameter $\boldsymbol{v_j}$ can be computed as

$$p(\mathbf{v}_j|\mathbf{R},\mathbf{U}) \propto \prod_{i:(i,j)\in\Omega} \mathcal{N}(r_{ij}|\boldsymbol{u}_i^\top \mathbf{v}_j, \beta^{-1})\mathcal{N}(\mathbf{v}_j|\mathbf{0},\lambda_v^{-1}\mathbf{I}_K)$$

Another Bayesian linear regression problem with weight vector $\boldsymbol{v}_j$

Bayesian Matrix Factorization: The CPs

- The CPs will have forms similar to solution of Bayesian linear regression

$$p(\boldsymbol{u}_i|\mathbf{R},\mathbf{V}) = \mathcal{N}(\boldsymbol{u}_i|\boldsymbol{\mu}_{u_i},\boldsymbol{\Sigma}_{u_i})$$

$$\boldsymbol{\Sigma}_{u_i} = (\lambda_u \mathbf{I} + \beta \textstyle\sum_{j:(i,j)\in\Omega} \boldsymbol{v}_j \boldsymbol{v}_j^\top)^{-1}$$

$$\boldsymbol{\mu}_{u_i} = \boldsymbol{\Sigma}_{u_i}(\beta \textstyle\sum_{j:(i,j)\in\Omega} r_{ij}\boldsymbol{v}_j)$$

$$p(\boldsymbol{v}_j|\mathbf{R},\mathbf{U}) = \mathcal{N}(\boldsymbol{v}_j|\boldsymbol{\mu}_{v_j},\boldsymbol{\Sigma}_{v_j})$$

$$\boldsymbol{\Sigma}_{v_j} = (\lambda_v \mathbf{I} + \beta \textstyle\sum_{i:(i,j)\in\Omega} \boldsymbol{u}_i \boldsymbol{u}_i^\top)^{-1}$$

$$\boldsymbol{\mu}_{v_j} = \boldsymbol{\Sigma}_{v_j}(\beta \textstyle\sum_{i:(i,j)\in\Omega} r_{ij}\boldsymbol{u}_i)$$

- These CPs can be updated in an alternating fashion until convergence
  - Many ways. One popular way is to use Gibbs sampling
  - Note: Hyperparameters can also be inferred by computing their CPs[1]

- Can extend Gaussian BMF easily to other exp. family distr. while maintaining local conj.
  - Example: Poisson likelihood and gamma priors on user/item parameters[2]

[1]"Bayesian Probabilistic Matrix Factorization using Markov Chain Monte Carlo" by Salakhutdinov and Mnih (2008)

[2]"Scalable recommendation with Poisson factorization" by Gopalan et al (2013)

BMF: Making Predictions

- PPD for each missing entry of the matrix (assuming hyperparams known)

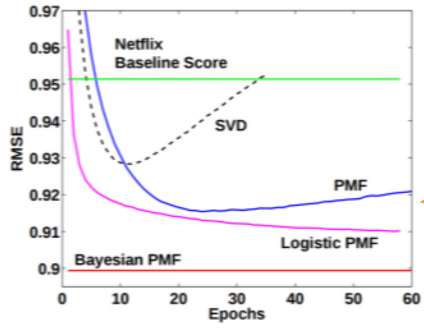
$$p(r_{ij}|\mathbf{R}) = \int\int p(r_{ij}|\boldsymbol{u}_i, \boldsymbol{v}_j)p(\boldsymbol{u}_i, \boldsymbol{v}_j|\mathbf{R})d\boldsymbol{u}_i d\boldsymbol{v}_j$$

- In general, this is intractable and needs approximation
  - If using Gibbs sampling, we can use S samples $(u_i^{(s)}, v_j^{(s)})_{s=1}^S$ to compute mean of r_{ij}
  - For the Gaussian likelihood case, the mean can be computed as

Can also compute the variance in the predicted r_{ij} using these S samples (think how)

$$\mathbb{E}[r_{ij}] \approx \frac{1}{S}\sum_{s=1}^{S} \boldsymbol{u}_i^{(s)\top}\boldsymbol{v}_j^{(s)} \qquad \text{(Monte-Carlo averaging)}$$

- Comparison of Bayesian MF with others (from Salakhutdinov and Mnih (2008))



All other baselines are optimization based or point estimation based probabilistic models (PMF = probabilistic matrix factorization with point estimation)

Summary

- Local conjugacy is helpful even for complex prob. models with many params
  - CPs will have a closed form
  - Easy to implement Gibbs sampling can be used to get the (approx.) posterior
  - Many other approx. inference algos (like variational inference) benefit from local conjugacy
- Helps to choose likelihood and priors on each param as exp. family distr.
  - So if we can't get a globally conjugate model, we can still get a model with local conjugacy
- Even if we can't have local conjugacy, the notion of CPs is applicable
  - We can break the inference problem into estimating CPs (exactly if we have local conjugacy, or approximately if we don't have local conjugacy)
  - Almost all approx. algorithms work by estimating CPs exactly or approximately

Coming Up

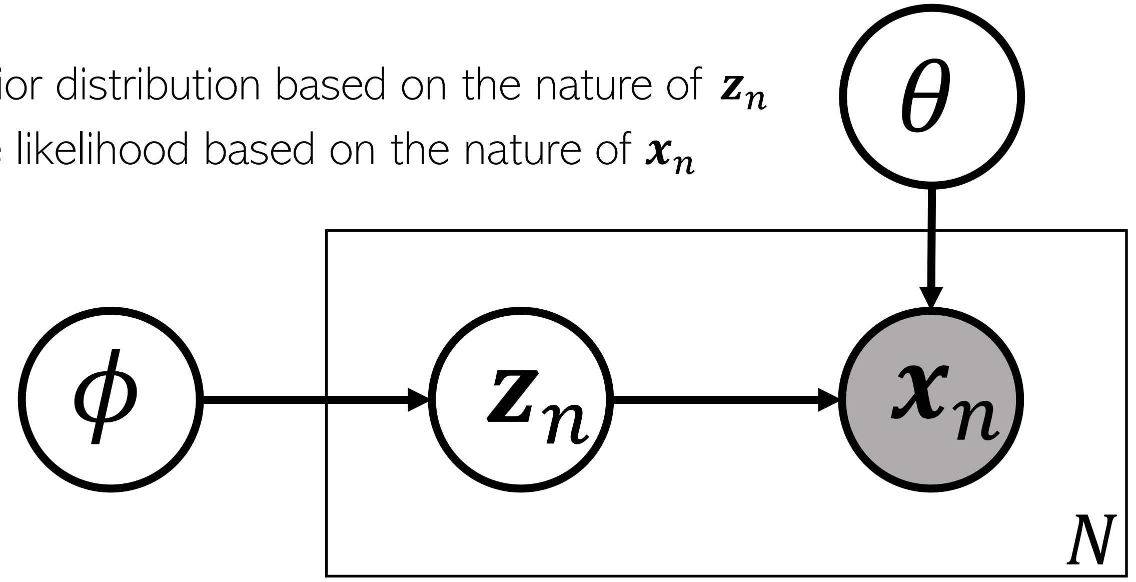
- Latent Variable Models
- Expectation Maximization

Latent Variable Models

- Application 1: Can use these to model 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)

$p(\boldsymbol{z}_n|\phi)$: 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 (hence called latent)
- 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

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

.. 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 τ_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
  - Usual distinction: Latent vars – posterior inferred; parameters – point estimation done
- 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 that MLE-II based inference does that
  - Maximize the marginal likelihood to do point estimation for hyperparams
  - Infer CP over the main parameter given the point estimates of hyperparams

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

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

- 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

Akin to maximizing marg-lik

- 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 (recall the schools problem in HW1)

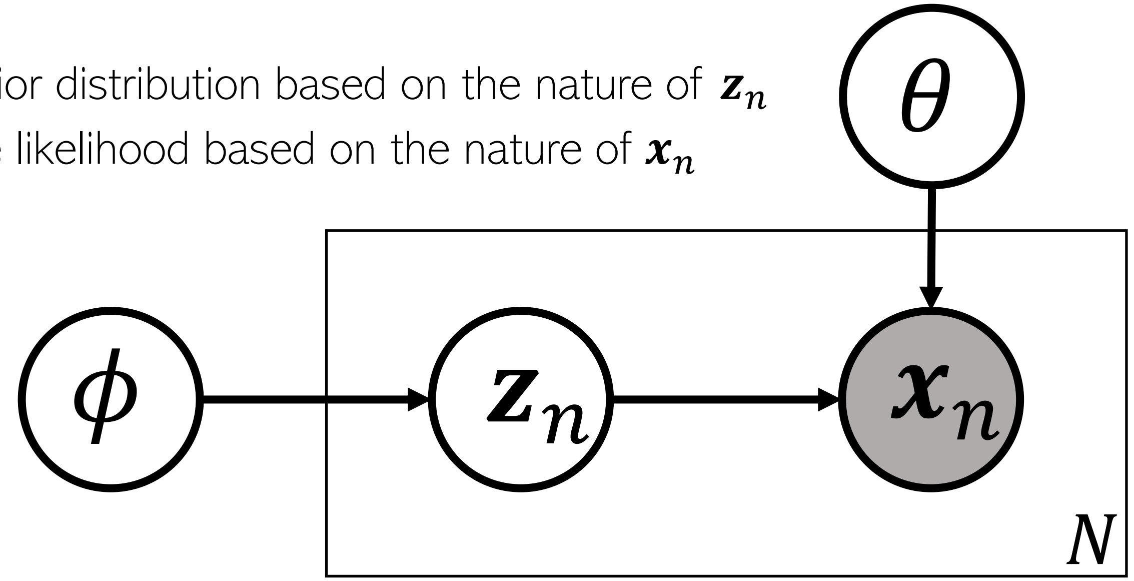
Inference/Parameter Estimation in Latent Variable Models using Expectation-Maximization (EM)

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 parameters $\boldsymbol{\Theta}$ via MLE. If we knew $\boldsymbol{z}_n$, we could solve

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

Easy to solve

- Easy. Usually closed form if $p(\boldsymbol{z}_n|\phi)$ and $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)$ have simple forms

In particular, if they are exp-fam distributions

- However, since in LVMs, $\boldsymbol{z}_n$ is hidden, the MLE problem for $\boldsymbol{\Theta}$ will be the following

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

- $\log p(\boldsymbol{x}_n|\Theta)$ will not have a simple expression since $p(\boldsymbol{x}_n|\Theta)$ requires sum/integral

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

- MLE now becomes difficult, no closed form expression for $\boldsymbol{\Theta}$
- Can we maximize some other quantity instead of $\log p(\boldsymbol{x}_n|\Theta)$ 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 $\mathcal{L}(q,\Theta) = \sum_Z q(Z)\log\left\{\frac{p(X,Z|\Theta)}{q(Z)}\right\}$

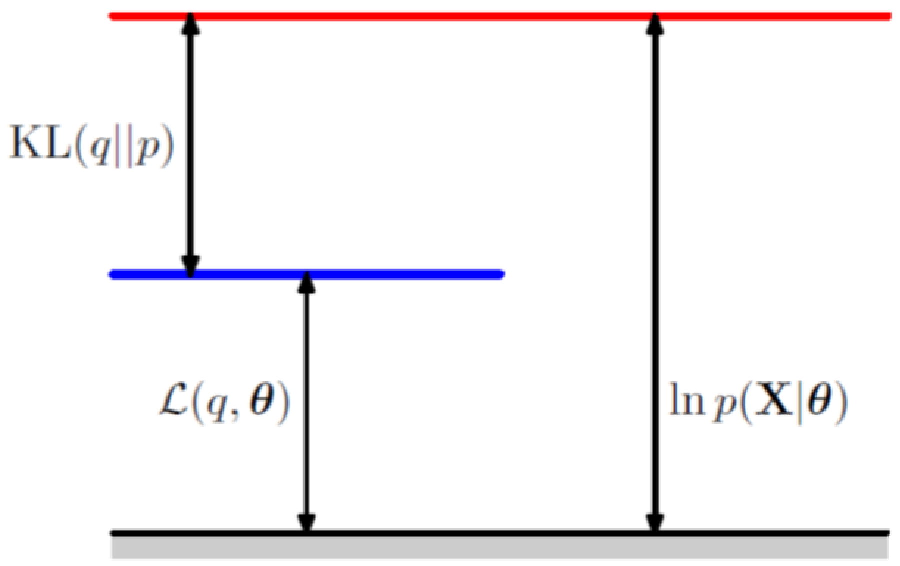
- $KL(q||p_z) = -\sum_Z q(\boldsymbol{Z})\log\left\{\frac{p(\boldsymbol{Z}|\boldsymbol{X},\Theta)}{q(\boldsymbol{Z})}\right\}$

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


KL(q||p)
$\mathcal{L}(q,\boldsymbol{\theta})$
$\ln p(\mathbf{X}|\boldsymbol{\theta})$

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

$\log p(\boldsymbol{X}|\Theta)$ 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)

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

The EM Algorithm

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

$$p(z_n^{(t)}|x_n, \Theta^{(t-1)}) = \frac{p(z_n^{(t)}|\Theta^{(t-1)})p(x_n|z_n^{(t)}, \Theta^{(t-1)})}{p(x_n|\Theta^{(t-1)})} \propto \text{prior} \times \text{likelihood}$$

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

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

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

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 $\text{argmax}_{\Theta} \sum_{n=1}^{N} \left[\log p\left(x_n, \hat{z}_n^{(t)} \middle| \Theta\right)\right]$ 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 $\mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n, \widehat{\Theta})}[\boldsymbol{z}_n]$
  - $\boldsymbol{z}_n \boldsymbol{z}_n^\top$ replaced by $\mathbb{E}_{p(\boldsymbol{z}_n|\boldsymbol{x}_n, \widehat{\Theta})}[\boldsymbol{z}_n \boldsymbol{z}_n^\top]$

- However, in some LVMs, these expectations are intractable to compute and need to be approximated (will see some examples later)

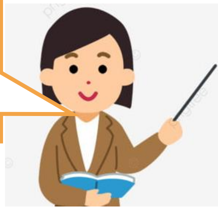
What's Going On?

Alternating between them until convergence to some local optima

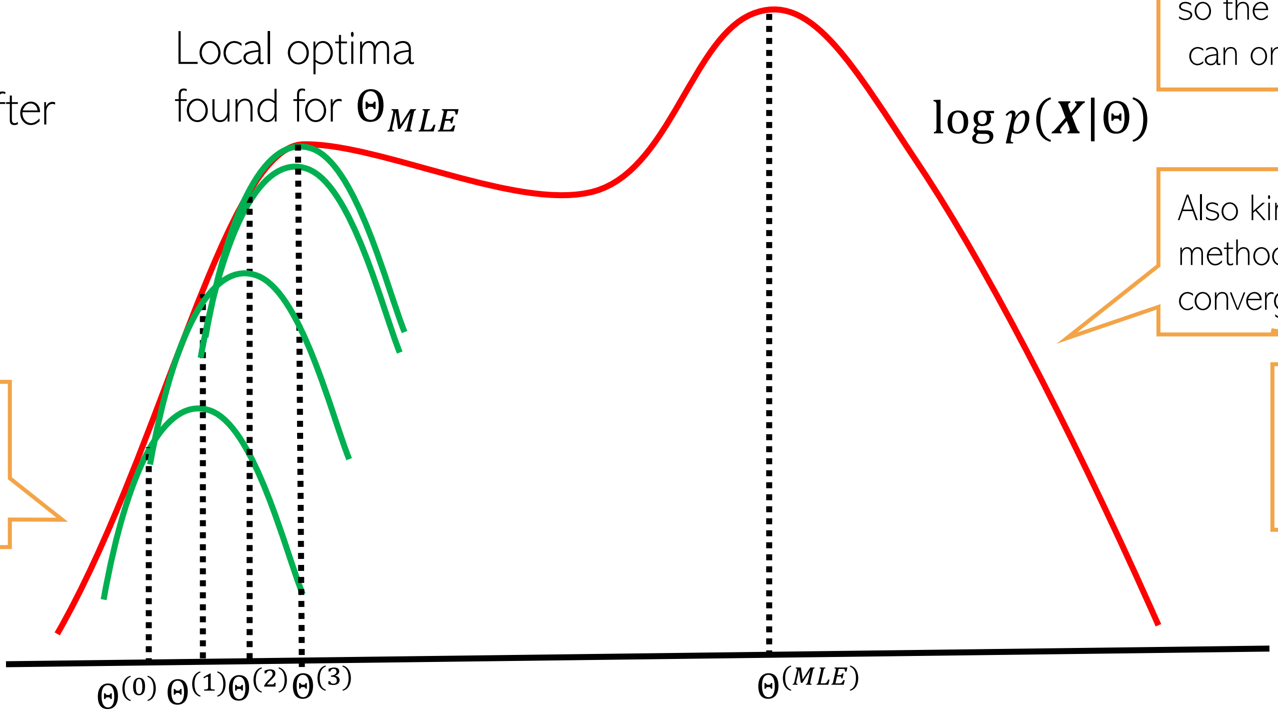
KL becomes zero and $\mathcal{L}(q,\Theta)$ becomes equal to $\log p(\boldsymbol{X}|\Theta)$; thus their curves touch at current Θ

- 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



Green curve: $\mathcal{L}(\hat{q},\Theta)$ after setting q to $\hat{q}$



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 (each data-point comes from one of K mixture components)
  - Examples: Mixture of Gaussians, Mixture of Experts, etc
- Latent variable models for dimensionality reduction or representation learning
  - Probabilistic PCA, Factor Analysis, Variational Autoencoders, etc
- Problems models with missing features/labels (treated as latent variables)
  - An example of problem with missing labels: Semi-supervised learning
- 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 $\boldsymbol{w}$

EM: Some 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

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

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

Results in Monte-Carlo EM

  - 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 Bayes (VB) inference, a.k.a. Variational Inference (VI)