Latent Variable Models (LVMs)

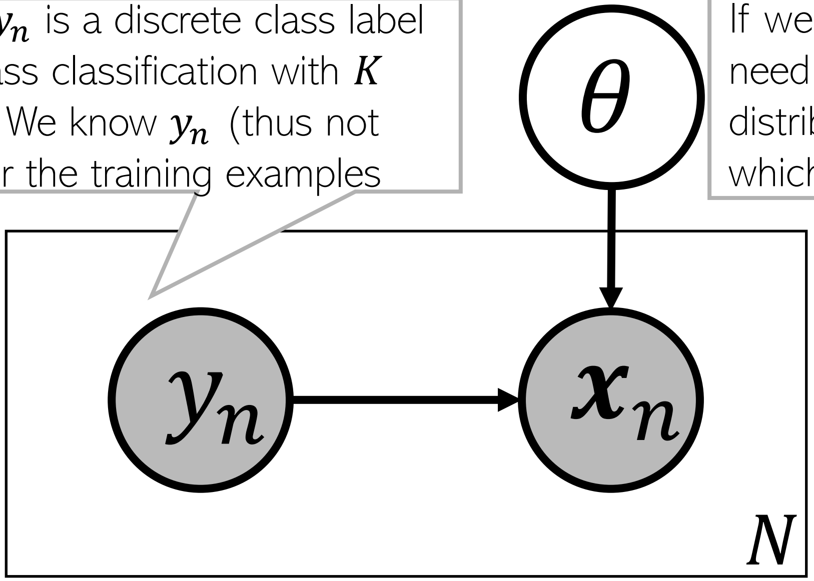
CS771: Introduction to Machine Learning

Piyush Rai

Example: Generative Models with Latent Variables

- Two generative models of inputs $\boldsymbol{x}_n$ without (left) and with (right) latent variables

Assume y_n is a discrete class label (multi-class classification with K classes). We know y_n (thus not latent) for the training examples

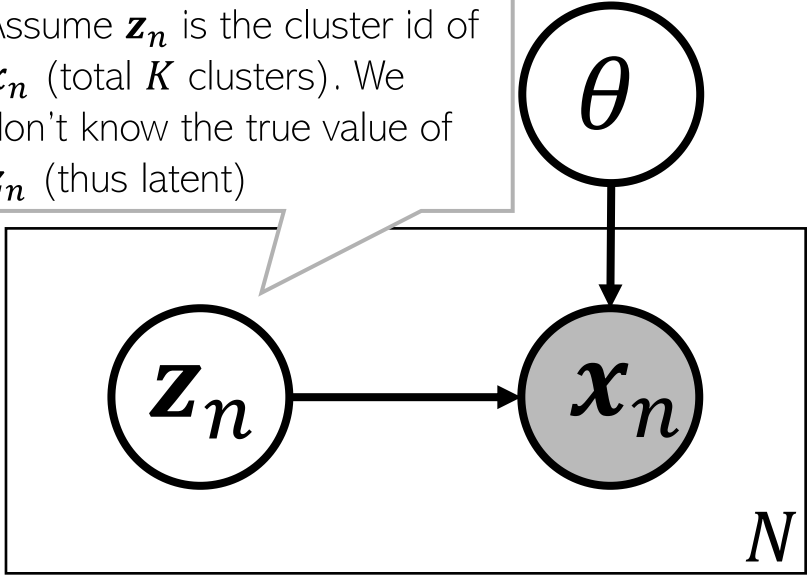


If we knew the value of z_n, we won't need to do MLE/MAP of the mixture distribution but only of $p(\boldsymbol{x}_n|\boldsymbol{z}_n = k, \theta)$ which is much easier

When z_n is not known, we can estimate $\boldsymbol{\theta}$ using an alternating estimating procedure: Estimate $\boldsymbol{z}_n$ given $\boldsymbol{\theta}$ and estimate $\boldsymbol{\theta}$ given $\boldsymbol{z}_n$

This procedure of estimating $\boldsymbol{\theta}$ also gives $\boldsymbol{z}_n$ as a by-product

Assume $\boldsymbol{z}_n$ is the cluster id of $\boldsymbol{x}_n$ (total K clusters). We don't know the true value of $\boldsymbol{z}_n$ (thus latent)



- Suppose we wish to estimate (e.g., using MLE/MAP) params $\boldsymbol{\theta}$ of distribution of $\boldsymbol{x}_n$
- For case 1, the distribution is $p(\boldsymbol{x}_n|\boldsymbol{y}_n, \boldsymbol{\theta})$ and MLE/MAP of $\boldsymbol{\theta}$ easy since $\boldsymbol{y}_n$ is known
- For case 2, distribution is more complex because true $\boldsymbol{z}_n$ is not known

Reason: The functional form of mixture can be messy

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

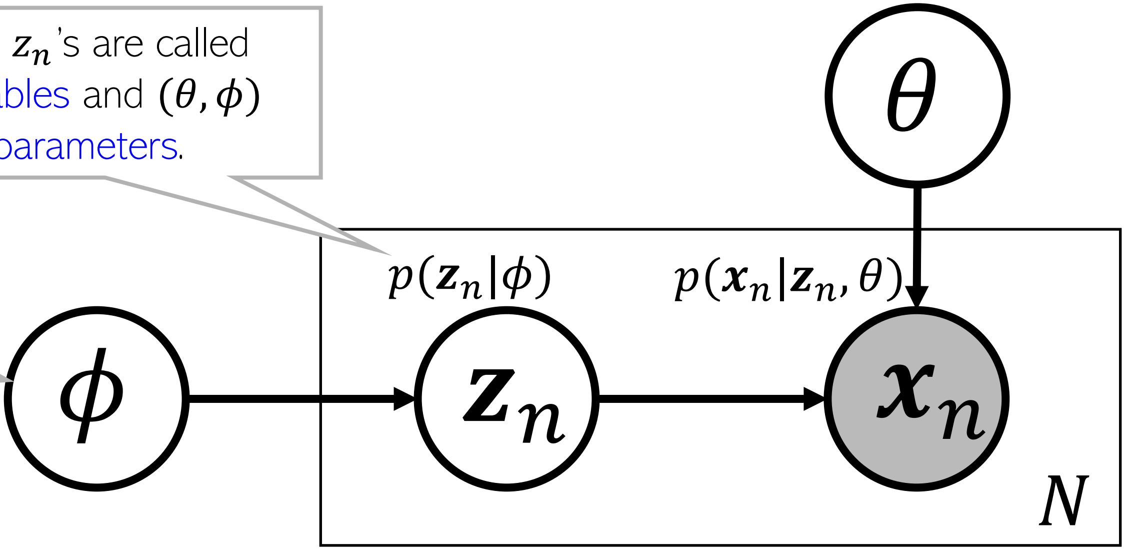
MLE/MAP a bit difficult for this more complex "mixture" of distributions

Components of an LVM

- Recall that the goal is to estimate $\boldsymbol{\theta}$ (and $\boldsymbol{z}_n$ is also unknown)
- In LVM, we treat $\boldsymbol{z}_n$ as a random variable and assume a prior distribution $p(\boldsymbol{z}_n|\phi)$

In an LVM, $\boldsymbol{z}_n$'s are called latent variables and (θ, ϕ) are called parameters.

Will also need to estimate ϕ in addition to θ



This prior tells us what the value of $\boldsymbol{z}_n$ is before we have seen the input $\boldsymbol{x}_n$

Ultimately, we will compute the distribution of $\boldsymbol{z}_n$ conditioned on the input $\boldsymbol{x}_n$

For example, a D-dimensional Gaussian if $\boldsymbol{x}_n \in \mathbb{R}^D$

- We will also assume a suitable conditional distribution $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)$ for $\boldsymbol{x}_n$
- The form of $p(\boldsymbol{z}_n|\phi)$ will depend on the nature of $\boldsymbol{z}_n$, e.g.,
  - If $\boldsymbol{z}_n$ is discrete with K possible values, $p(\boldsymbol{z}_n|\phi) = \textbf{multinoulli}(\boldsymbol{z}_n|\boldsymbol{\pi})$
  - If $\boldsymbol{z}_n \in \mathbb{R}^K$, $p(\boldsymbol{z}_n|\phi) = \mathcal{N}(\boldsymbol{z}_n|\boldsymbol{\mu}, \boldsymbol{\Sigma})$, a K-dim Gaussian

Why Direct MLE/MAP is Hard for LVMs?

- Direct MLE/MAP of parameters $(\theta, \phi) = \Theta$ without estimating $\boldsymbol{z}_n$ is hard
- Reason: Given N observations $\boldsymbol{x}_n, n = 1,2, \ldots, N$, the MLE problem for Θ will be

Also note that $p(\boldsymbol{x}_n, \boldsymbol{z}_n|\Theta) = p(\boldsymbol{z}_n|\phi)p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)$

$$\arg\max_{\Theta} \sum_{n=1}^{N} \log p(\boldsymbol{x}_n|\Theta) = \arg\max_{\Theta} \sum_{n=1}^{N} \log \sum_{\boldsymbol{z}_n} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\Theta)$$

Summing over all possible values $\boldsymbol{z}_n$ can take (would be an integral instead of sum if $\boldsymbol{z}_n$ is continuous

Gaussian Mixture Model (GMM).

- For a mixture of K Gaussians, $p(\boldsymbol{x}_n|\Theta)$ will be

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

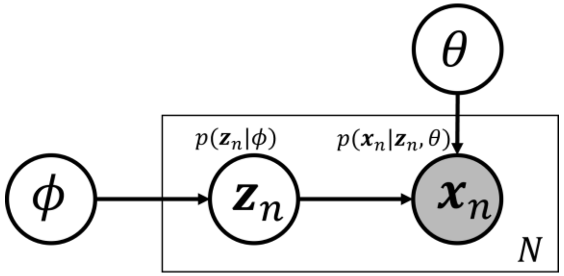
- The MLE problem for GMM would be

ALT-OPT or EM makes it simpler by using hard/soft guesses of $\boldsymbol{z}_n$'s

$$\arg\max_{\Theta} \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(\boldsymbol{x}_n|\mu_k, \Sigma_k)$$

The log of sum doesn't give us a simple expression; MLE can still be done using gradient based methods but updates will be complicated.

How to Guess z_n in an LVM?



- Note that $\boldsymbol{z}_n$ is a random variable with prior distribution $p(\boldsymbol{z}_n|\phi)$
- Can compute its conditional posterior (CP) distribution as

Called conditional posterior because it is conditioned on data as well as Θ (assuming we have already estimated Θ)

$(\theta, \phi) = \Theta$

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta) = \frac{p(\boldsymbol{z}_n|\Theta)p(\boldsymbol{x}_n|\boldsymbol{z}_n, \Theta)}{p(\boldsymbol{x}_n|\Theta)} = \frac{p(\boldsymbol{z}_n|\phi)p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)}{p(\boldsymbol{x}_n|\Theta)}$$

- If we just want the single best (hard) guess of $\boldsymbol{z}_n$ then that can be computed as

Used in ALT-OPT for LVMs

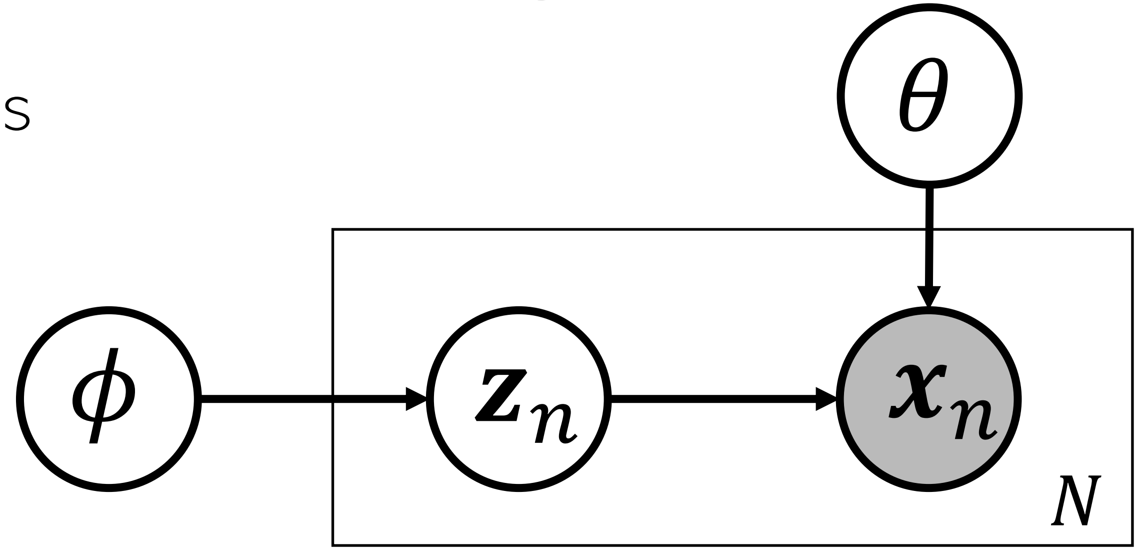
$$\hat{z}_n = \text{argmax}_{z_n}\, p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta) = \text{argmax}_{z_n}\, p(\boldsymbol{z}_n|\phi)p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)$$

Used in Expectation-Maximization (EM) algo for LVMs

- Otherwise, we can compute and use CP $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)$ to get a soft/probabilistic guess
  - Using the CP $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)$ we can compute quantities such as expectation of $\boldsymbol{z}_n$
  - If $p(\boldsymbol{z}_n|\phi)$ and $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \theta)$ are conjugate to each other then CP $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)$ is easy to compute
- Computing hard guess is usually easier but ignores the uncertainty in $\boldsymbol{z}_n$

LVMs: Incomplete vs Complete Data Log Likelihood

- We can define two types of likelihoods for LVMs
  - Incomplete data log likelihood (ILL) $\log p(\boldsymbol{X}|\Theta)$
  - Complete data log likelihood (CLL) $\log p(\boldsymbol{X},\boldsymbol{Z}|\Theta)$



- Named so because we can think of latent $\boldsymbol{Z}$ "completing" the observed data $\boldsymbol{X}$
- Since $\boldsymbol{Z}$ is never observed (is latent), to estimate Θ we must maximize the ILL

$$\underset{\Theta}{\operatorname{argmax}} \log p(\boldsymbol{X}|\Theta) = \underset{\Theta}{\operatorname{argmax}} \log \sum_{\boldsymbol{Z}} p(\boldsymbol{X},\boldsymbol{Z}|\Theta)$$

- But since ILL maximization is hard (log of sum/integral over the unknown $\boldsymbol{Z}$), we instead maximize the CLL $p(\boldsymbol{X},\boldsymbol{Z}|\Theta)$ using hard/soft guesses of $\boldsymbol{Z}$

MLE for LVM

Also, we can use this idea to find MAP solution of Θ if we want. Assume a prior $p(\Theta)$ and simply add a $\log p(\Theta)$ term to these objectives

Note that we aren't solving the original MLE problem $\underset{\Theta}{\operatorname{argmax}} \log p(\boldsymbol{X}|\Theta)$ anymore.
However, what we are solving now is still justifiable theoretically (will see later)

- If using a hard guess

$$\Theta_{MLE} = \underset{\Theta}{\operatorname{argmax}} \log p(\boldsymbol{X}, \hat{\boldsymbol{Z}} | \Theta)$$

- If using a soft (probabilistic) guess

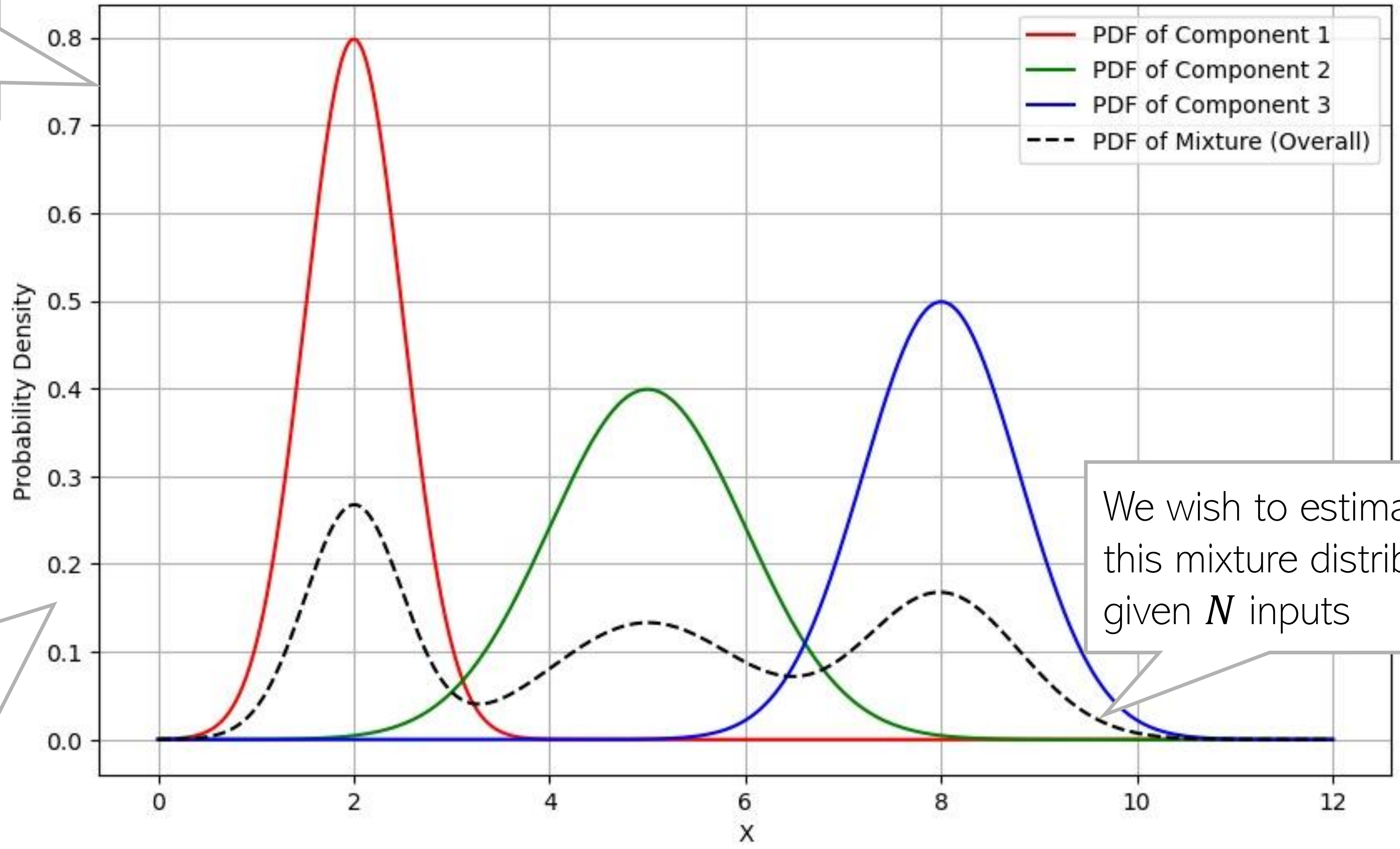
$$\Theta_{MLE} = \underset{\Theta}{\operatorname{argmax}} \mathbb{E}[\log p(\boldsymbol{X}, \boldsymbol{Z} | \Theta)]$$

- In LVMs, hard and soft guesses of $\boldsymbol{Z}$ would depend on Θ (since $\boldsymbol{Z}$ and Θ are coupled)

- Thus we need a procedure which alternates between estimating $\boldsymbol{Z}$ and estimating Θ

An LVM: Gaussian Mixture Model

Inputs are assumed generated from a mixture of Gaussians. But we don't know which input was generated by which Gaussian



If we knew which input came from which Gaussian (akin to knowing their true labels), the problem is easy – simply estimate each Gaussian using the inputs that came from that Gaussian (just like generative classification)

We wish to estimate this mixture distribution given N inputs

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
- Can assume label z_n to be one-hot and then $z_{nk} = 1$ if $z_n = k$, and $z_{nk} = 0$, o/w
  - Note: For each label, using notation z_n instead of y_n
- Assuming class marginal $p(z_n = k) = \pi_k$, the model's params $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$
- The MLE objective $\log p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)$ is (will provide a note for the proof)

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

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

Same as $\frac{N_k}{N}$

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

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

MLE for GMM: Using Guesses of z_n

Will have the exact same form for the expression of MLE objective as generative classification with Gaussian class-conditionals (except z_n is unknown)

- Using a hard guess $\hat{\boldsymbol{z}}_n = \text{argmax}_{z_n}\, p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)$, the MLE problem for GMM

Log likelihood of Θ w.r.t. data $\boldsymbol{X}$ and hard guesses $\hat{\boldsymbol{Z}}$ of cluster ids

Assuming x_n given z_n and Θ are i.i.d.

$$\Theta_{MLE} = \underset{\Theta}{\text{argmax}} \log p\left(\boldsymbol{X}, \hat{\boldsymbol{Z}} \,|\Theta\right) = \text{argmax}_{\Theta} \sum_{n=1}^{N} \sum_{k=1}^{K} \hat{z}_{nk}[\log \pi_k + \log \mathcal{N}(\boldsymbol{x}_n|\mu_k, \Sigma_k)]$$

- Using a soft guess $\mathbb{E}[\boldsymbol{z}_n]$, the MLE problem for GMM

Expected log likelihood of Θ w.r.t. data $\boldsymbol{X}$ and $\boldsymbol{Z}$

z_{nk} appears at only one place in the log likelihood expression so easily replaced by expectation of z_{nk} w.r.t the CP $p(\boldsymbol{z}_n|\boldsymbol{x}_n, \Theta)$

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

- In both cases, the MLE solution for $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$ will be identical to that of generative classification with Gaussian class cond with z_{nk} replaced by $\hat{z}_{nk}$ or $\mathbb{E}[z_{nk}]$
  - Case 1 solved using ALT-OPT alternating b/w estimating Θ_{MLE} and $\hat{\boldsymbol{Z}}$
  - Case 2 solved using Expectation Maximization (EM) alternating b/w estimating Θ_{MLE} and $\mathbb{E}[\boldsymbol{Z}]$

ALT-OPT for GMM

- We will assume we have a "hard" (most probable) guess of z_n, say $\hat{z}_n$
- ALT-OPT which maximizes $\log p(\boldsymbol{X}, \hat{\boldsymbol{Z}} | \Theta)$ would look like this
  - Initialize $\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^K$ as $\widehat{\Theta}$
  - Repeat the following until convergence
    - For each n, compute most probable value (our best guess) of z_n as

$$\hat{z}_n = \text{argmax}_{k=1,2,\ldots,K}\, p(z_n = k | \widehat{\Theta}, \boldsymbol{x}_n)$$

Proportional to prior prob times likelihood, i.e.,
$p(z_n = k|\widehat{\Theta})\, p(x_n|z_n = k, \widehat{\Theta}) = \hat{\pi}_k \mathcal{N}(x_n|\hat{\mu}_k, \hat{\Sigma}_k)$

Posterior probability of point x_n belonging to cluster k, given current Θ

    - Solve MLE problem for Θ using most probable z_n's

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

N_k : Effective number of points in cluster k

$$\hat{\pi}_k = \frac{1}{N} \sum_{n=1}^N \hat{z}_{nk} \qquad \hat{\mu}_k = \frac{1}{N_k} \sum_{n=1}^N \hat{z}_{nk} \boldsymbol{x}_n$$

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

Expectation-Maximization (EM) for GMM

- EM finds Θ_{MLE} by maximizing $\mathbb{E}[\log p(\boldsymbol{X},\boldsymbol{Z}|\Theta)]$
- Note: Expectation will be w.r.t. the CP of $\boldsymbol{Z}$, i.e., $p(\boldsymbol{Z}|\boldsymbol{X},\Theta)$
- 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
    - Compute CP $p(\boldsymbol{Z}|\boldsymbol{X},\hat{\Theta})$ using current estimate of Θ. Since obs are i.i.d, compute for each n (and for $k = 1,2,..K$)

Why w.r.t. this distribution? Will see justification in a bit

Note that EM for GMM also gives a soft clustering $z_n = [\gamma_{n1}, \gamma_{n2}, \dots, \gamma_{nK}]$ for each input x_n

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 $\mathbb{E}[\log p(\boldsymbol{X},\boldsymbol{Z}|\Theta)]$

$$\frac{\hat{\pi}_k \mathcal{N}(x_n|\hat{\mu}_k, \hat{\Sigma}_k)}{\sum_{\ell=1}^{K} \hat{\pi}_\ell \mathcal{N}(x_n|\hat{\mu}_\ell, \hat{\Sigma}_\ell)}$$

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

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

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

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

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

Posterior probability of x_n belonging to k^{th} cluster

EM for GMM (Contd)

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)

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

Accounts for fraction of points in each cluster

Accounts for cluster shapes (since each cluster is a Gaussian

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

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

What is EM Doing?

Maximization of ILL

Assuming $\boldsymbol{Z}$ to be discrete, else replace it by an integral

- The MLE problem was $\Theta_{MLE} = \underset{\Theta}{\operatorname{argmax}} \log p(\boldsymbol{X}|\Theta) = \underset{\Theta}{\operatorname{argmax}} \log \sum_{\boldsymbol{z}} p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)$

As an approximation

- What EM (and ALT-OPT) maximized is expected CLL: $\Theta_{MLE} = \underset{\Theta}{\operatorname{argmax}} \mathbb{E}[\log p(\boldsymbol{X}, \boldsymbol{Z}|\Theta)]$
- We did not solve the original problem (max of ILL). Is it okay?
- Assume $p_z = p(\boldsymbol{Z}|\boldsymbol{X}, \Theta)$ and $q(\boldsymbol{Z})$ to be some prob distribution over $\boldsymbol{Z}$, then

Function of a distribution q and parameter Θ

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

May verify this identity

- In the above $\mathcal{L}(q, \Theta) = \sum_Z q(Z) \log\left\{\frac{p(X,Z|\Theta)}{q(Z)}\right\}$ and $KL(q||p_z) = -\sum_Z q(\boldsymbol{Z}) \log\left\{\frac{p(\boldsymbol{Z}|\boldsymbol{X},\Theta)}{q(\boldsymbol{Z})}\right\}$
- Since KL is always non-negative $\log p(\boldsymbol{X}|\Theta) \geq \mathcal{L}(q, \Theta)$, so $\mathcal{L}(q, \Theta)$ is a lower-bound on ILL
- Thus if we maximize $\mathcal{L}(q, \Theta)$, it will also improve $\log p(\boldsymbol{X}|\Theta)$

What is EM Doing?

The posterior distribution of Z given older parameters Θ^{old} (will need this posterior to get the expectation of CLL)

- As we saw, $\mathcal{L}(q,\Theta)$ depends on q and Θ
- Let's maximize $\mathcal{L}(q,\Theta)$ w.r.t. q with Θ fixed at 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}

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

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

$$\begin{aligned}\Theta^{\text{new}} = \text{argmax}_\Theta \mathcal{L}(\hat{q},\Theta) &= \text{argmax}_\Theta \sum_{Z} p(\boldsymbol{Z}|\boldsymbol{X},\Theta^{\text{old}}) \log\left\{\frac{p(\boldsymbol{X},\boldsymbol{Z}|\Theta)}{p(\boldsymbol{Z}|\boldsymbol{X},\Theta^{\text{old}})}\right\} \\ &= \text{argmax}_\Theta \sum_{Z} p(\boldsymbol{Z}|\boldsymbol{X},\Theta^{\text{old}}) \log p(\boldsymbol{X},\boldsymbol{Z}|\Theta) \\ &= \text{argmax}_\Theta \mathbb{E}_{p(\boldsymbol{Z}|\boldsymbol{X},\Theta^{\text{old}})}[\log p(\boldsymbol{X},\boldsymbol{Z}|\Theta)] \\ &= \text{argmax}_\Theta \mathcal{Q}(\Theta,\Theta^{\text{old}})\end{aligned}$$

Maximization of expected CLL w.r.t. the posterior distribution of Z given older parameters Θ^{old}

Recap: ALT-OPT vs EM

- ALT-OPT does the following
  1. Initialize $\Theta = \hat{\Theta}$
  2. Estimate $\mathbf{Z}$ as $\hat{\mathbf{Z}} = \arg\max_{\mathbf{Z}} \log p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})$
  3. Estimate Θ as $\hat{\Theta} = \arg\max_{\Theta} \log p(\mathbf{X}, \hat{\mathbf{Z}}|\Theta)$
  4. Go to step 2 if not converged

This step could potentially throw away a lot of information about the latent variable $\boldsymbol{Z}$

- EM addresses it using "soft" version of ALT-OPT
  1. Initialize $\Theta = \hat{\Theta}$
  2. Compute the posterior distribution of $\mathbf{Z}$, i.e., $p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})$
  3. Estimate Θ by maximizing the expected CLL $\hat{\Theta} = \mathbb{E}_{p(\mathbf{Z}|\mathbf{X}, \hat{\Theta})}[\log p(\mathbf{X}, \mathbf{Z}|\Theta)]$
  4. Go to step 2 if not converged

ALT-OPT can be seen as as approximation of EM – the posterior $p(\boldsymbol{Z}|\boldsymbol{X}, \Theta)$ is replaced by a point mass at its mode

EM: An Illustration

Alternating between them until convergence to some local optima

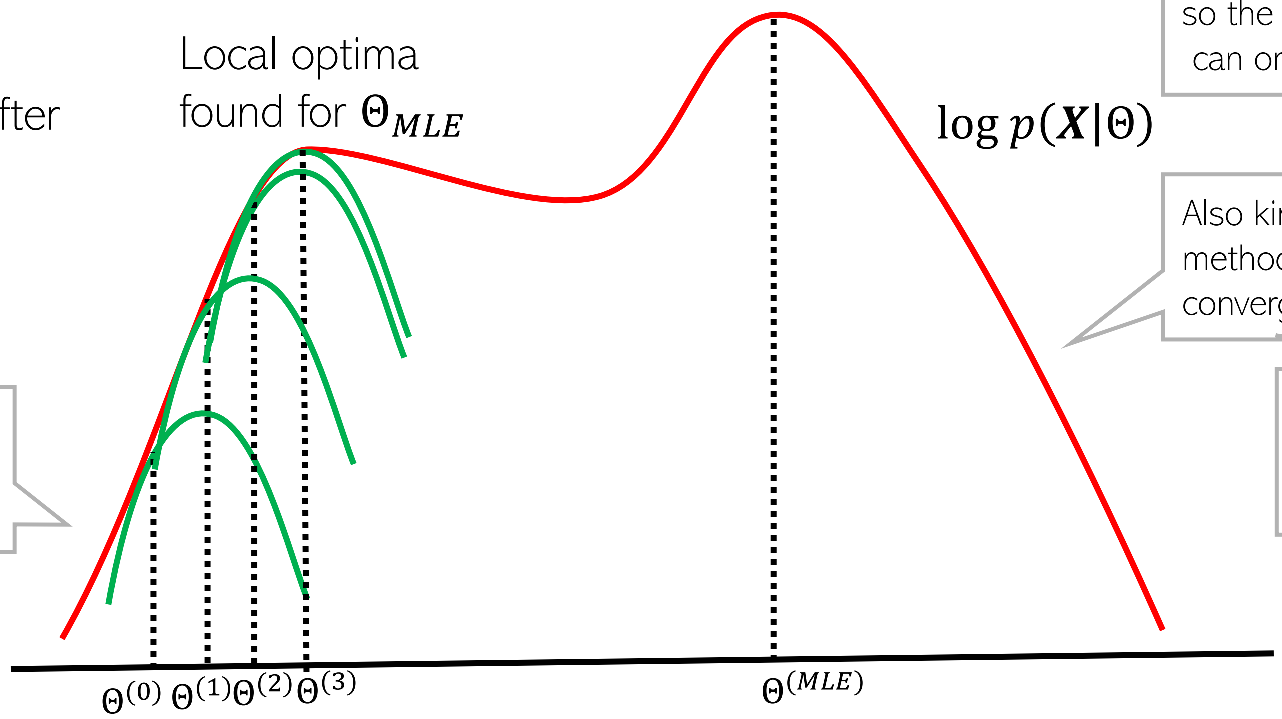
Makes $\mathcal{L}(q, \Theta)$ equal to $\log p(\boldsymbol{X}|\Theta)$; thus the curves touch at current Θ

- As we saw, EM maximizes the lower bound $\mathcal{L}(q, \Theta)$ in two steps
- Step 1 finds the optimal q (call it $\hat{q}$) by setting it 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 $\hat{q}$

Local optima found for Θ_{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

The EM Algorithm in its general form..

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

The EM Algorithm

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

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

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

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

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 2 and maximize the CLL in Step 3 using that, i.e., do $\mathbf{argmax}_\Theta \sum_{n=1}^N \left[\log p\left(\boldsymbol{x}_n,\hat{z}_n^{(t)}\middle|\Theta\right)\right]$ this will be ALT-OPT

The Expected CLL

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

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

Was indeed the case of GMM: $p(\boldsymbol{z}_n|\Theta)$ was multinoulli, $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \Theta)$ was Gaussian

- If $p(\boldsymbol{z}_n|\Theta)$ and $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \Theta)$ are exponential family distributions, then $\mathcal{Q}(\Theta, \Theta^{\text{old}})$ 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 (beyond the score of CS771)

Detour: Exponential Family

- Exponential Family is a family of prob. distributions that have the form

$$p(x|\theta) = h(x)\exp[\theta^\top T(x) - A(\theta)]$$

Even though their standard form may not look like this, they can be rewritten in this form after some algebra

- Many well-known distribution (Bernoulli, Binomial, multinoulli, Poisson, beta, gamma, Gaussian, etc.) are examples of exponential family distributions
- θ is called the natural parameter of the family

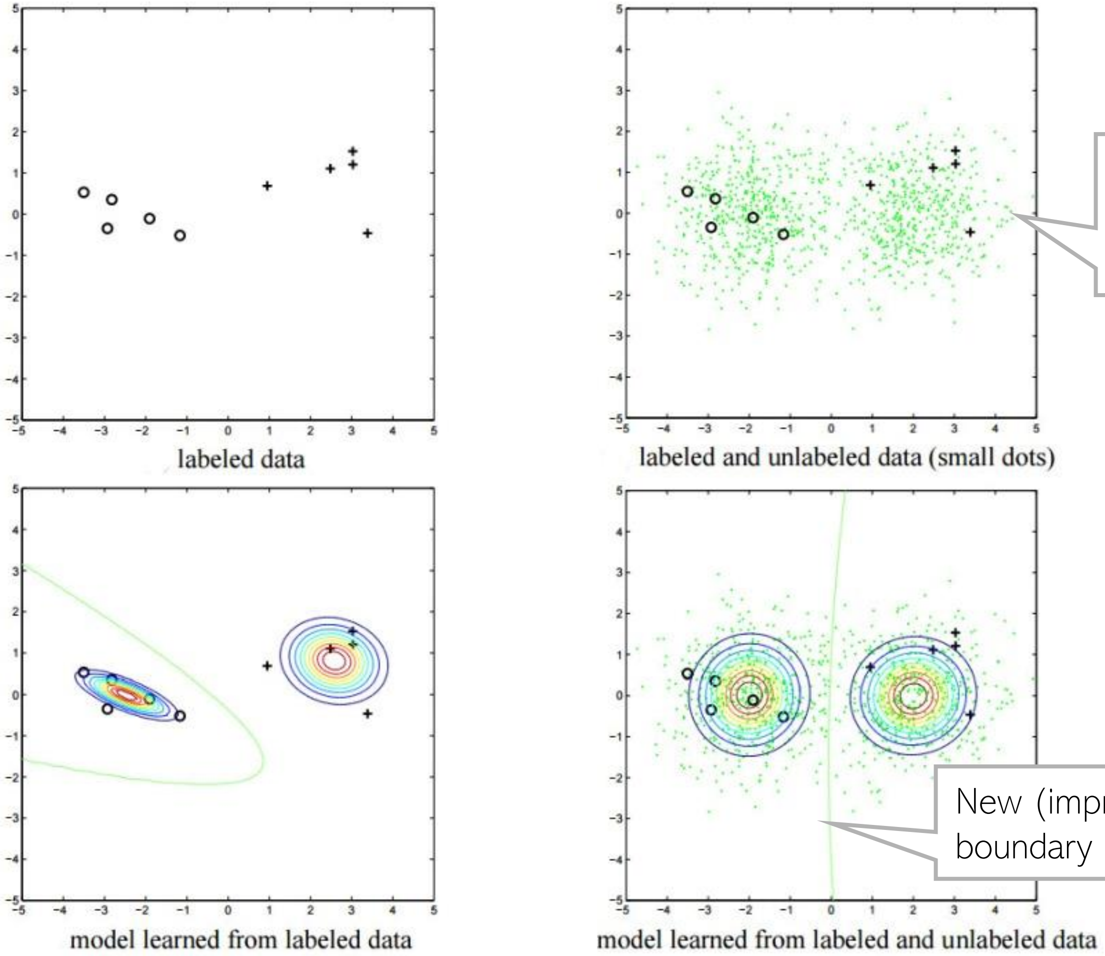
Natural params are a function of the distribution parameters in the standard form

- $h(x), T(x),$ and $A(\theta)$ are known functions (specific to the distribution)
- $T(x)$ is called the sufficient statistics: estimates of θ contain x in form of suff-stats
- Every exp. family distribution also has a conjugate distribution (often also in exp. family)
- Also, MLE/MAP is usually quite simple since $\log p(x|\theta)$ will have a simple expression
- Also useful in fully Bayesian inference since they have conjugate priors

https://en.wikipedia.org/wiki/Exponential_family

LVM for Semi-supervised Learning

- Unlabeled data can help in supervised learning as well



Added some unlabeled inputs (green points)

We don't know their true labels but can treat the true labels as latent variables and estimate them along with the rest of the parameters of the generative classification model

A generative classifier learned using the labelled examples

Decision boundary (green curve)

A small training set may not be able to help learn the true decision boundary

New (improved) decision boundary

LVM for Semi-supervised Learning (SSL)

- Suppose we have N labeled $\{\boldsymbol{x}_n, y_n\}_{n=1}^{N}$ and M unlabeled examples $\{\boldsymbol{x}_n\}_{n=N+1}^{N+M}$
- We wish to learn a classifier using both labelled and unlabeled examples
- We can treat the labels of $\{x_n\}_{n=N+1}^{N+M}$ as latent variables and use ALT-OPT or EM
  - $z_n = y_n$ for labeled examples $n = 1,2,\ldots,N$
  - z_n estimated (hard/soft guess) for unlabeled examples $n = N+1,\ldots,N+M$

This SSL model is a hybrid of supervised generative classification (with Gaussian class-conditionals) and GMM

- Assuming generative classification with Gaussian class-conditional ($\Theta = \{\pi_k, \mu_k, \Sigma_k\}_{k=1}^{K}$)

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

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

Assuming we are using EM (soft guess), otherwise ALT-OPT (hard guess) can be used too, as we did in GMM

Another LVM: Probabilistic PCA (PPCA)

- Assume $\boldsymbol{x}_n \in \mathbb{R}^D$ as a linear mapping of a latent var $\boldsymbol{z}_n \in \mathbb{R}^K$ + Gaussian noise

A "reverse" generative way of thinking about PCA (low-dim z_n generating high-dim x_n

This linear mapping can be replaced by more powerful nonlinear mapping of the form $\boldsymbol{x}_n = f(\boldsymbol{z}_n) + \boldsymbol{\epsilon}_n$ where f can be modeled using a deep neural net (e.g., models like variational autoencoders)

$D \times 1$ offset/bias

$D \times K$ matrix

Drawn from a zero-mean D-dim Gaussian $\mathcal{N}(\mathbf{0}, \sigma^2 I_D)$

$$\boldsymbol{x}_n = \boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n + \boldsymbol{\epsilon}_n$$

- Equivalent to saying $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n, \sigma^2 I_D)$
- Assume a zero-mean K-dim Gaussian prior on $\boldsymbol{z}_n$, so $p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n|\mathbf{0}, I_K)$
- We would like to do MLE for $\Theta = (\boldsymbol{\mu}, \boldsymbol{W}, \boldsymbol{\sigma}^2)$
- ILL for this model $p(\boldsymbol{x}_n|\boldsymbol{\mu}, \boldsymbol{W}, \sigma^2)$ is also a Gaussian (thanks to Gaussian properties)

$$p(\boldsymbol{x}_n|\boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) = \int p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) p(\boldsymbol{z}_n) d\boldsymbol{z}_n = N(\boldsymbol{x}_n\,|\boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^\top + \sigma^2 I_D)$$

PRML 12.2.1

- Maximizing ILL w.r.t. $\Theta = (\boldsymbol{\mu}, \boldsymbol{W}, \boldsymbol{\sigma}^2)$ is possible but requires solving eig decomp. problem
- We can use ALT-OPT/EM to estimate $\Theta = (\boldsymbol{\mu}, \boldsymbol{W}, \boldsymbol{\sigma}^2)$ more efficiently without eig decomp.

Learning PPCA using EM

- Instead of maximizing the ILL $p(\boldsymbol{x}_n|\boldsymbol{\mu}, \boldsymbol{W}, \sigma^2) = N(\boldsymbol{x}_n\,|\boldsymbol{\mu}, \boldsymbol{W}\boldsymbol{W}^\top + \sigma^2 I_D)$, let's use ALT-OPT/EM
- EM will instead maximize expected CLL, with CLL (assume $\boldsymbol{\mu} = \mathbf{0}$) given by

$$\log p(X, Z|W, \sigma^2) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n, \boldsymbol{z}_n|\boldsymbol{W}, \sigma^2) = \log \prod_{n=1}^{N} p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{W}, \sigma^2)\, p(\boldsymbol{z}_n)$$

- Using $p(\boldsymbol{x}_n|\boldsymbol{z}_n, \boldsymbol{W}, \sigma^2) = \mathcal{N}(\boldsymbol{x}_n|\boldsymbol{W}\boldsymbol{z}_n, \sigma^2 I_D)$ and $p(\boldsymbol{z}_n) = \mathcal{N}(\boldsymbol{z}_n|\mathbf{0}, I_K)$

$$\text{CLL} = -\sum_{n=1}^{N} \left\{ \frac{D}{2}\log\sigma^2 + \frac{1}{2\sigma^2}\|\boldsymbol{x}_n\|^2 - \frac{1}{\sigma^2}\boldsymbol{z}_n^\top \boldsymbol{W}^\top \boldsymbol{x}_n + \frac{1}{2\sigma^2}\text{trace}(\boldsymbol{z}_n\boldsymbol{z}_n^\top \boldsymbol{W}^\top \boldsymbol{W}) + \frac{1}{2}\text{trace}(\boldsymbol{z}_n\boldsymbol{z}_n^\top) \right\}$$

- Expected CLL will require $\mathbb{E}[\boldsymbol{z}_n]$ and $\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]$ w.r.t. conditional posterior of $\boldsymbol{z}_n$

Using the fact that $p(\boldsymbol{x}_n|\boldsymbol{z}_n)$ and $p(\boldsymbol{z}_n)$ are Gaussians and the CP is just the reverse conditional $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ and must also be Gaussian

$$
\begin{aligned}
p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) &= \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n, \sigma^2\mathbf{M}^{-1}) \qquad \text{where } \mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}_K \\
\mathbb{E}[\mathbf{z}_n] &= \mathbf{M}^{-1}\mathbf{W}^\top \mathbf{x}_n \\
\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top] &= \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \text{cov}(\mathbf{z}_n) = \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \sigma^2\mathbf{M}^{-1}
\end{aligned}
$$

Learning PPCA using EM

- The EM algo for PPCA alternates between two steps
  - Compute CP of $\boldsymbol{z}_n$ given parameters $\Theta = (\mathbf{W}, \sigma^2)$ and required expectatuions

$$
\begin{aligned}
p(\mathbf{z}_n|\mathbf{x}_n, \mathbf{W}) &= \mathcal{N}(\mathbf{M}^{-1}\mathbf{W}^\top\mathbf{x}_n, \sigma^2\mathbf{M}^{-1}) \qquad \text{where } \mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2\mathbf{I}_K \\
\mathbb{E}[\mathbf{z}_n] &= \mathbf{M}^{-1}\mathbf{W}^\top\mathbf{x}_n \\
\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top] &= \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \text{cov}(\mathbf{z}_n) = \mathbb{E}[\mathbf{z}_n]\mathbb{E}[\mathbf{z}_n]^\top + \sigma^2\mathbf{M}^{-1}
\end{aligned}
$$

Note: This approach does not assume/ensure that $\boldsymbol{W}$ is orthonormal

  - Maximize the expected CLL $\mathbb{E}[\log p(\boldsymbol{X}, \boldsymbol{Z}|\mathbf{W}, \sigma^2)]$ w.r.t. $\boldsymbol{W}$ and $\boldsymbol{\sigma}^2$

$$
\mathbb{E}[\text{CLL}] = -\sum_{n=1}^{N}\left\{\frac{D}{2}\log\sigma^2 + \frac{1}{2\sigma^2}\|\boldsymbol{x}_n\|^2 - \frac{1}{\sigma^2}\mathbb{E}[\boldsymbol{z}_n^\top]\boldsymbol{W}^\top\boldsymbol{x}_n + \frac{1}{2\sigma^2}\text{trace}(\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top]\boldsymbol{W}^\top\boldsymbol{W}) + \frac{1}{2}\text{trace}(\mathbb{E}[\boldsymbol{z}_n\boldsymbol{z}_n^\top])\right\}
$$

Unlike standard PCA (non-probabilistic), no eigendecomposition needed to estimate $\boldsymbol{W}$

$$
\begin{aligned}
\mathbf{W}_{new} &= \left[\sum_{n=1}^{N}\mathbf{x}_n\mathbb{E}[\mathbf{z}_n]^\top\right]\left[\sum_{n=1}^{N}\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top]\right]^{-1} \\
\sigma^2_{new} &= \frac{1}{ND}\sum_{n=1}^{N}\left\{||\mathbf{x}_n||^2 - 2\mathbb{E}[\mathbf{z}_n]^\top\mathbf{W}_{new}^\top\mathbf{x}_n + \text{tr}\left(\mathbb{E}[\mathbf{z}_n\mathbf{z}_n^\top]\mathbf{W}_{new}^\top\mathbf{W}_{new}\right)\right\}
\end{aligned}
$$

Note: setting $\boldsymbol{\sigma}^2 = 0$ makes it equivalent to standard PCA without orthonormality constraint, but EM is more efficient since no eigendecomposition is needed

- Will get ALT-OPT if we use mode of the CP as $\hat{\boldsymbol{z}}_n$ in the CLL

Generative Models can generate synthetic data!

- Once parameters $\Theta = (\boldsymbol{\mu}, \boldsymbol{W}, \boldsymbol{\sigma}^2)$ are learned, we can even generate new data, e.g.,
  - Generate a random $\boldsymbol{z}_n$ from $\mathcal{N}(\boldsymbol{0}, I_K)$
  - Generate $\boldsymbol{x}_n$ condition on $\boldsymbol{z}_n$ from $\mathcal{N}(\boldsymbol{\mu} + \boldsymbol{W}\boldsymbol{z}_n, \sigma^2 I_D)$

In addition to, of course, reducing the data dimensionality

(a) Training data (b) Random samples

Generated using a more sophisticated generative model, not PPCA (but similar in formulation)

Methods such as variational autoencoders, GAN, diffusion models, etc are based on similar ideas

EM: Some Comments

- Good initialization is important
- The E and M steps may not always be possible to perform exactly. Some reasons
  - CP of latent variables $p(\boldsymbol{Z}|\boldsymbol{X},\Theta)$ may not be easy to find and may require approx.
  - Even if $p(\boldsymbol{Z}|\boldsymbol{X},\Theta)$ is easy, expected CLL, i.e., $\mathbb{E}[\log p(\boldsymbol{X},\boldsymbol{Z}|\Theta)]$ may still not be tractable

Monte-Carlo EM

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

..and may need to be approximated, e.g., using Monte-Carlo expectation

Gradient methods may still be needed for this step

  - 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)
- Other advanced probabilistic inference algorithms are based on ideas similar to EM
  - E.g., Variational Bayesian inference a.k.a. Variational Inference (VI)
- EM is also related to non-convex optimization algorithms Majorization-Maximization (MM)
  - MM maximizes a difficult-to-optimize objective function by iteratively constructing surrogate functions that are easier to maximize (in EM, the surrogate function was the CLL)