Latent Variable Models

CS771: Introduction to Machine Learning
Nisheeth

Coin toss example

- Say you toss a coin N times
- You want to figure out its bias
- Bayesian approach
  - Find the generative model
  - Each toss ~ Bern(θ)
  - θ ~ Beta(α,β)
- Draw the generative model in plate notation

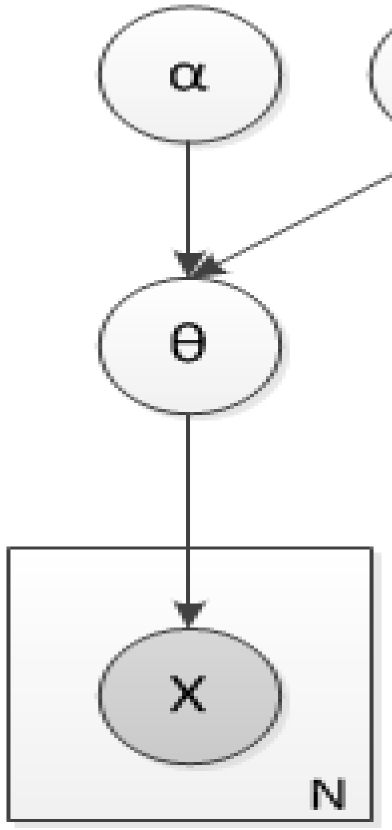


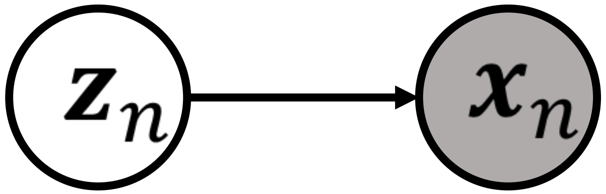
Plate notation

- Random variables as circles
- Parameters, fixed values as squares
- Repetitions of conditional probability structures as rectangular ‘plates’
- *Switch* conditioning as squiggles
- Random variables observed in practice are shaded

Generative Models with Latent Variables

- Have already looked at generative models for supervised learning
- Generative models are even more common/popular for unsupervised learning, e.g.,
  - Clustering
  - Dimensionality Reduction
  - Probability density estimation

Latent variable z_n usually encodes some latent properties of the observation $\boldsymbol{x}_n$



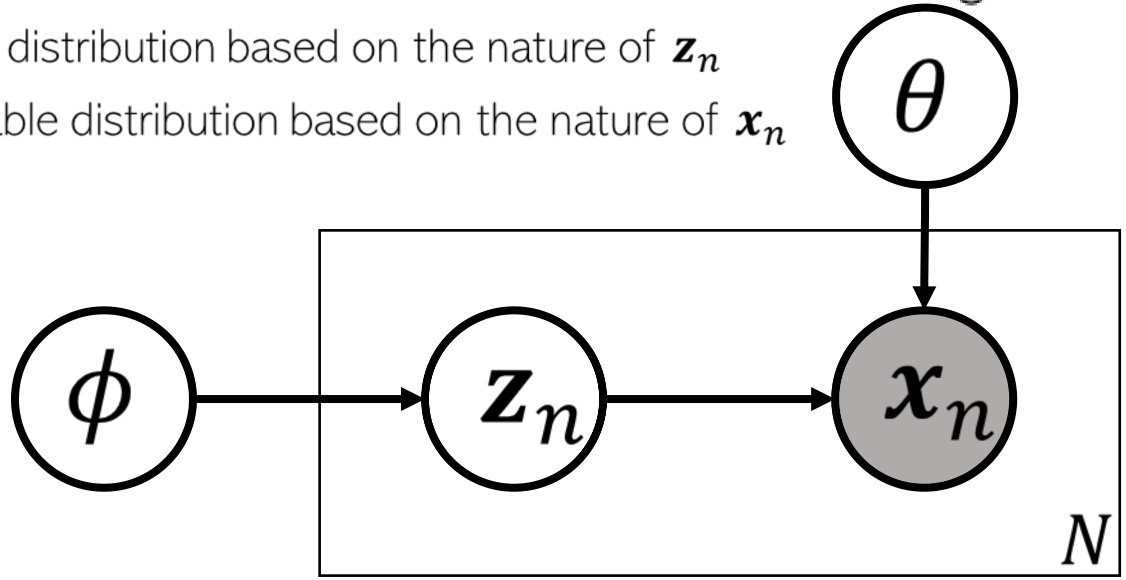
- In such models, each data point is associated with a latent variable
  - Clustering: The cluster id $\boldsymbol{z}_n$ (discrete, or a K-dim one-hot rep, or a vector of cluster membership probabilities)
  - Dimensionality reduction: The low-dim representation $\boldsymbol{z}_n \in \mathbb{R}^K$
- These latent variables will be treated as random variables, not just fixed unknowns
- Will therefore assume a suitable prior distribution on these and estimate their posterior
  - If we only need a point estimate (MLE/MAP) of these latent variables, that can be done too

Generative Models with Latent Variables

- A typical generative model with latent variables might look like this

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

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



Need probability distributions on both

- In this generative model, observations $\boldsymbol{x}_n$ assumed generated via latent variables $\boldsymbol{z}_n$
- The unknowns in such latent var models (LVMs) are of two types
  - Global variables: Shared by all data points (θ and ϕ in the above diagram)
  - Local variables: Specific to each data point ($\boldsymbol{z}_n$'s in the above diagram)
- Note: Both global and local unknowns can be treated as r.v.'s

However, here we will only treat the local variables $\boldsymbol{z}_n$'s as random latent variable and regard θ and ϕ as other unknown "parameters" of the model

An Example of a Generative LVM

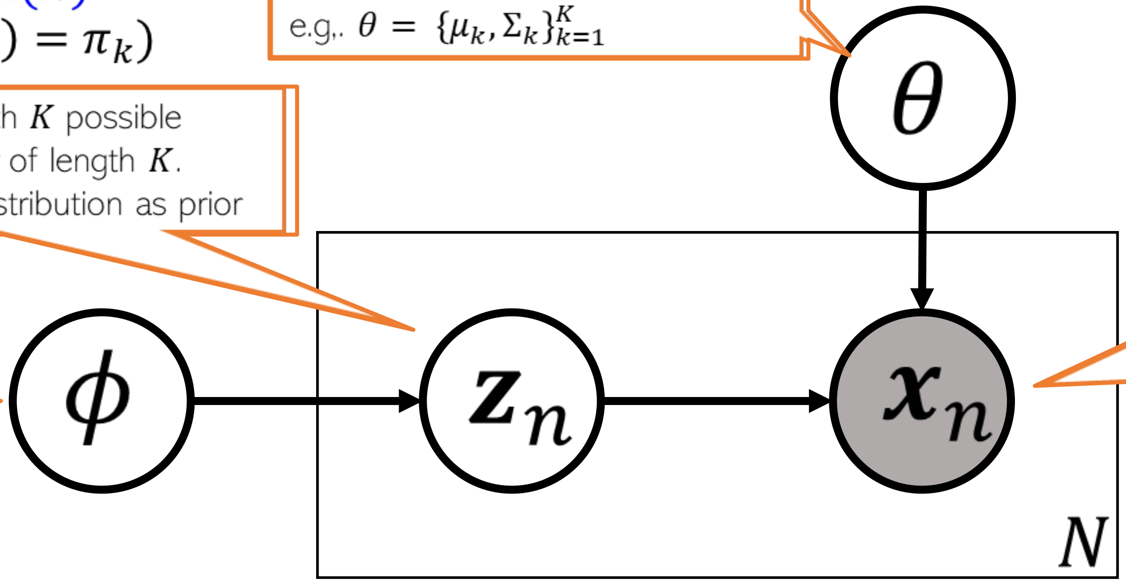
- Probabilistic Clustering can be formulated as a generative latent variable model
- Assume K probability distributions (e.g., Gaussians), one for each cluster

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

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

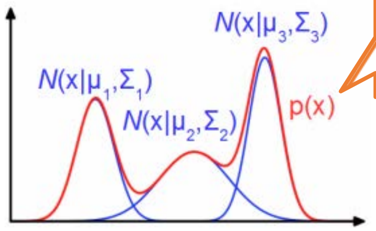
The parameter vector $\boldsymbol{\pi} = [\pi_1, \pi_2, \ldots, \pi_K]$ of the multinoulli distribution



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

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

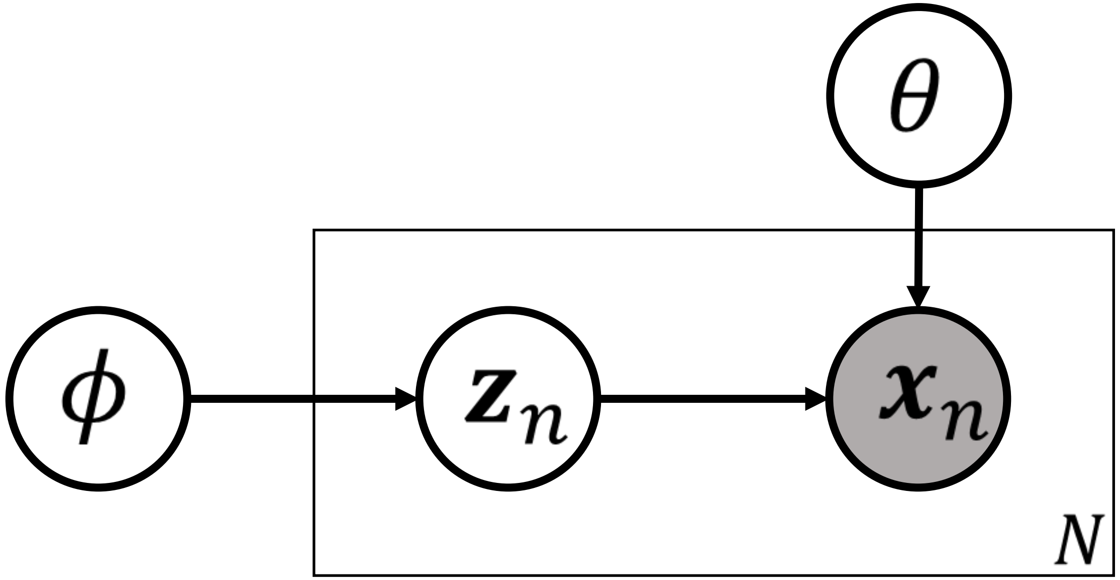


$p(x)$ is a Gaussian mixture model (GMM)

- In any such LVM, $\boldsymbol{\phi}$ denotes parameters of the prior distribution on $\boldsymbol{z}_n$
- .. and $\boldsymbol{\theta}$ denotes parameters of the likelihood distribution on $\boldsymbol{x}_n$

Parameter Estimation for Generative LVM

- So how do we estimate the parameters of a generative LVM, say prob. clustering?



- The guess about $\boldsymbol{z}_n$ can be in one of the two forms
  - A "hard" guess – a fixed value (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 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

Parameter Estimation for Generative LVM

- Can we estimate parameters $(\theta, \phi) = \Theta$ (say) of an LVM without estimating z_n?
- In principle yes, but it is harder
- Given N observations $x_n, n = 1,2, \dots, N$, the MLE problem for Θ will be

The discussion here is also true for MAP estimation of Θ

$$\arg\max_{\Theta} \sum_{n=1}^{N} \log p(x_n|\Theta) = \arg\max_{\Theta} \sum_{n=1}^{N} \log \sum_{z_n} p(x_n, z_n|\Theta)$$

Also note that $p(x_n, z_n|\Theta) = p(z_n|\phi)p(x_n|z_n, \theta)$

After the summation/integral on the RHS, $p(x_n|\Theta)$ is no longer exp. family even if $p(z_n|\phi)$ and $p(x_n|z_n, \phi)$ are in exp-fam ☹

Summing over all possible values z_n can take (would be an integral instead of sum if z_n is continuous

- For the probabilistic clustering model (GMM) we saw, $p(x_n|\Theta)$ will be

Convex combination (mixture) of K Gaussians. No longer an exp-family distribution

$$p(x_n|\Theta) = \sum_{k=1}^{K} p(x_n, z_n = k|\Theta) = \sum_{k=1}^{K} p(z_n = k|\phi)p(x_n|z_n = k, \theta) = \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$$

- MLE problem thus will be $\arg\max_{\Theta} \sum_{n=1}^{N} \log \sum_{k=1}^{K} \pi_k \mathcal{N}(x_n|\mu_k, \Sigma_k)$

The log of sum doesn't give us a simple expression; MLE can still be done using gradient based methods but update will be complicated. ALT-OPT or EM make it simpler by using guesses of z_n's

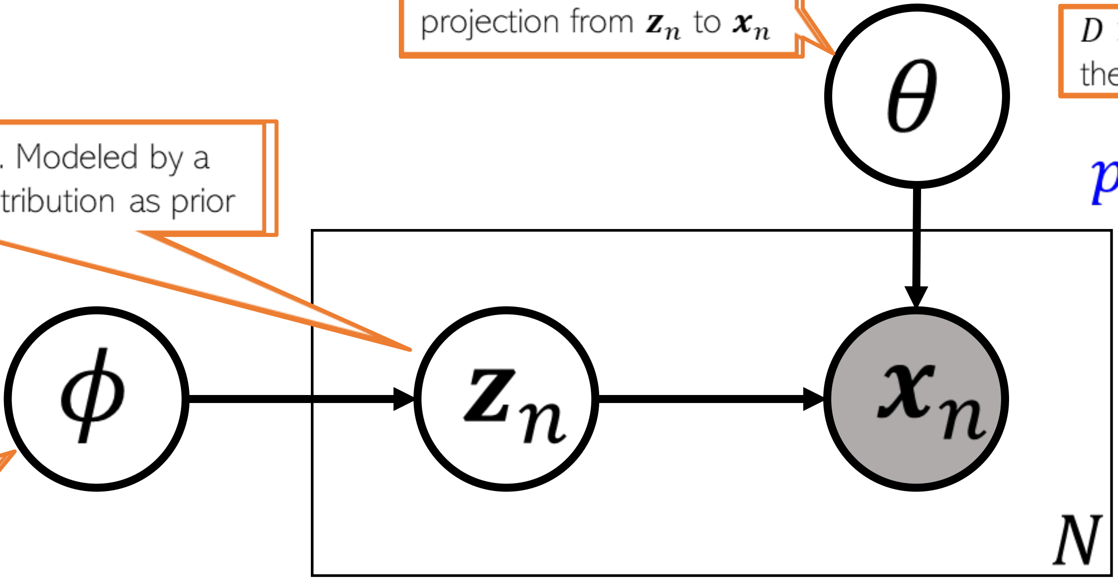
Another Example of a Gen. LVM

If the $\boldsymbol{z}_n$ were known, it just becomes a probabilistic version of the multi-output regression problem where $\boldsymbol{z}_n \in \mathbb{R}^K$ are the observed input features and $\boldsymbol{x}_n \in \mathbb{R}^K$ are the vector-valued outputs

- Probabilistic PCA (PPCA) is another example of a generative latent var model
- Assume a K-dim latent var $\boldsymbol{z}_n$ mapped to a D-dim observation $\boldsymbol{x}_n$ via a prob. mapping

$$p(z_n \mid \phi) = \mathcal{N}(0, \mathbf{I}_K)$$

Parameters defining the projection from $\boldsymbol{z}_n$ to $\boldsymbol{x}_n$



Real-valued vector of length K. Modeled by a zero-mean K-dim Gaussian distribution as prior

The parameters of the Gaussian prior on $\boldsymbol{z}_n$. In this example, no such parameters are actually needed since mean is zero and cov matrix is identity, but can use nonzero mean and more general cov matrix for the Gaussian prior

$D \times K$ mapping matrix

$K \times 1$

$D \times 1$ mean of the mapping

$$\boldsymbol{\mu}_n = \boldsymbol{W}\boldsymbol{z}_n$$

$$p(\boldsymbol{x}_n \mid \boldsymbol{z}_n, \boldsymbol{W}, \sigma^2) = \mathcal{N}(\boldsymbol{\mu}_n, \sigma^2 \mathbf{I}_D)$$

Probabilistic mapping means that will be not exactly but somewhere around the mean (in some sense, it is a noisy mapping):

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

Also, instead of a linear mapping $\boldsymbol{W}\boldsymbol{z}_n$, the z_n to x_n mapping can be defined as a nonlinear mapping (variational autoencoders, kernel based latent variable models)

Added Gaussian noise just like probabilistic linear regression

- PPCA has several benefits over PCA, some of which include
  - Can use suitable distributions for $\boldsymbol{x}_n$ to better capture properties of data
  - Parameter estimation can be done faster without eigen-decomposition (using ALT-OPT/EM algos)

Generative Models and Generative Stories

- Data generation for a generative model can be imagined via a generative story
- This story is just our hypothesis of how "nature" generated the data
- For the Gaussian mixture model (GMM), the (somewhat boring) story is as follows
  - For each data point $\boldsymbol{x}_n$ with index $n = 1,2,\ldots,N$
    - Generate its cluster assignment by drawing from prior $p(z_n|\phi)$
    $$\boldsymbol{z}_n \sim \text{multinoulli}(\boldsymbol{\pi})$$
    - Assuming $z_n = k$, generate the data point $\boldsymbol{x}_n$ from $p(\boldsymbol{x}_n|\boldsymbol{z}_n,\theta)$
    $$\boldsymbol{x}_n \sim \mathcal{N}(\mu_k, \Sigma_k)$$
- Can imagine a similar story for PPCA with $\boldsymbol{z}_n$ generated from $\mathcal{N}(\mathbf{0}, \mathbf{I}_K)$ and then conditioned on $\boldsymbol{z}_n$, the observation $\boldsymbol{x}_n$ generated from $p(\boldsymbol{x}_n|\boldsymbol{z}_n,\boldsymbol{W},\sigma^2) = \mathcal{N}(\boldsymbol{W}\boldsymbol{z}_n, \sigma^2\mathbf{I}_D)$
- For GMM/PPCA, the story is rather simplistic but for more sophisticated models, gives an easy way to understand/explain the model, and data generation process