(Shallow and Deep) Generative Models

CS772A: Probabilistic Machine Learning

Piyush Rai

Plan for today

- Latent Factor Models
- Latent Dirichlet Allocation (LDA)
- Deep generative models: Variational Autoencoders

Factor Analysis and Probabilistic PCA

- Assumption: Latent variables $\boldsymbol{z}_n \in \mathbb{R}^K$ typically assumed to have a Gaussian prior
  - If we want sparse latent variable, can use Laplace or spike-and-slab prior on $\boldsymbol{z}_n$
  - More complex extensions of FA/PPCA use a mixture of Gaussians prior on $\boldsymbol{z}_n$
- Assumption: Observations $\boldsymbol{x}_n \in \mathbb{R}^D$ typically assumed to have a Gaussian likelihood
  - Other likelihood models (e.g., exp-family) can also be used if data not real-valued
- Relationship between $\boldsymbol{z}_n$ and $\boldsymbol{x}_n$ modeled by a noisy linear mapping

$$\boldsymbol{x}_n = \boldsymbol{W}\boldsymbol{z}_n + \epsilon_n = \sum_{k=1}^{K} \boldsymbol{w}_k z_{nk} + \epsilon_n$$

Zero-mean and diagonal or spherical Gaussian noise

Linear combination of the columns of $\boldsymbol{W}$

$$p(\mathbf{z}_n) = \mathcal{N}(\mathbf{z}_n|0, \mathbf{I})$$
$$p(\mathbf{x}_n|\mathbf{z}_n) = \mathcal{N}(\boldsymbol{x}_n|\mathbf{W}\mathbf{z}_n, \Psi)$$

Diagonal for FA, spherical for PPCA

- Unknowns $\boldsymbol{W}$, $\boldsymbol{z}_n$'s, and Ψ can be learned
  - EM, VI, MCMC

Some Other Classical Models

- Gamma-Poisson latent factor model
  - Assumes K-dim non-negative latent variable $\mathbf{z_n}$ and D-dim count-valued observations $\mathbf{x_n}$
  - An example: Each $\mathbf{x_n}$ is the word-count vector representing a document

Popular for modeling count-valued data (in text analysis, recommender systems, etc)

Non-negative priors often give a nice interpretability to such latent variable models (will see some more examples of such models shortly)

$$p(\mathbf{z_n}) = \prod_{k=1}^{K} \text{Gamma}(z_{nk}|a_k, b_k))$$
$$p(\mathbf{x_n}|\mathbf{z_n}) = \prod_{d=1}^{D} \text{Poisson}(x_{nd}|f(\mathbf{w_d}, \mathbf{z_n}))$$

This is the rate of the Poisson. It should be non-negative, $\exp(\mathbf{w}_d^\top \boldsymbol{z}_n)$, or simply $\mathbf{w}_d^\top \boldsymbol{z}_n$ if $\boldsymbol{w}_d$ is also non-negative (e.g., using a gamma/Dirichlet prior on it)

  - This can be thought of as a probabilistic non-negative matrix factorization model

- Dirichlet-Multinomial/Multinoulli PCA
  - Assumes K-dim non-negative latent variable $\mathbf{z_n}$ and D categorical obs $\mathbf{x_n} = \{\boldsymbol{x}_{nd}\}_{d=1}^{D}$
  - An example: Each $\mathbf{x_n}$ is a document with D words in it (each word is a categorical value)

Also sums to 1

$$p(\mathbf{z_n}) = \text{Dirichlet}(\mathbf{z_n}|\boldsymbol{\alpha})$$
$$p(\mathbf{x_n}|\mathbf{z_n}) = \prod_{d=1}^{D} \text{Multinoulli}(x_{nd}|f(\mathbf{w_d}, \mathbf{z_n}))$$

This should give the probability vector of the multinoulli over x_{nd}. It should be non-negative and should sums to 1

Latent Dirichlet Allocation (LDA) a.k.a. “Topic Model”

Motivation: Multinomial Mixture Model for Text

- Assume D documents, and document d has N_d words in it
- We can represent doc d by a word count vector $\boldsymbol{w}_d$
- Assuming a vocab of V unique words, $\boldsymbol{w}_d$ is a $V \times 1$ vector of counts
  - w_{dv} = no of times word v appears in doc d
- Let's model the docs by a mixture of K multinomial distributions, each V-dim
  - The k^{th} multinomial modeled by a V-dim prob vector $\boldsymbol{\phi}_k$ (sums to 1)
  - $\boldsymbol{\phi}_k$ can be thought of as a "topic vector" (or just "topic"), ϕ_{kv}: prob of word v in topic k
- Generative model and plate diagram below

Each topic is a prob. distribution over word tokens

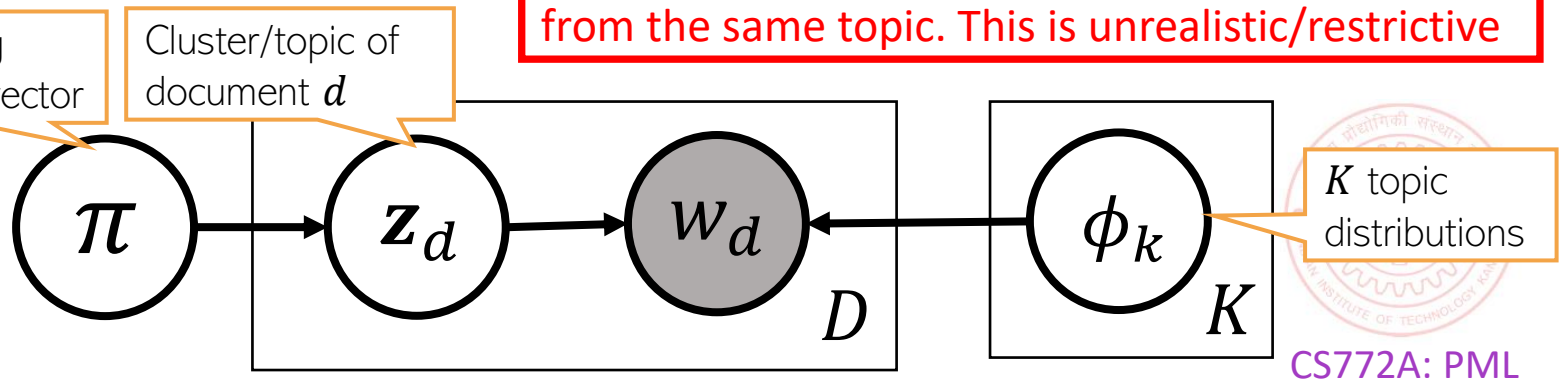
Each representing a "topic" (K topics)

Limitation: Each doc d belongs to a single cluster $\boldsymbol{z}_d$ and all words in a document assumed to be from the same topic. This is unrealistic/restrictive

$$\boldsymbol{z}_d \sim \text{multinoulli}(\pi)$$

$$\boldsymbol{w}_d \sim \text{multinomial}(\phi_{z_d}, N_d)$$

Counts will sum to N_d

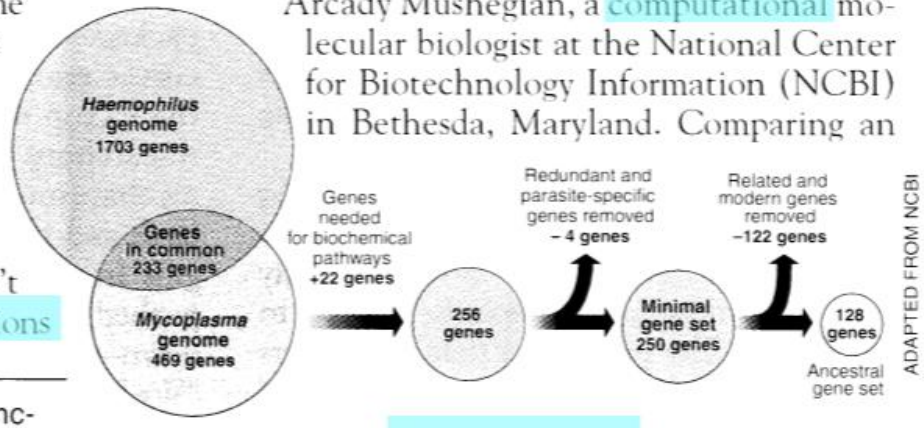


Documents can be about multiple topics

Seeking Life's Bare (Genetic) Necessities

COLD SPRING HARBOR, NEW YORK—How many genes does an organism need to survive? Last week at the genome meeting here,* two genome researchers with radically different approaches presented complementary views of the basic genes needed for life. One research team, using computer analyses to compare known genomes, concluded that today's organisms can be sustained with just 250 genes, and that the earliest life forms required a mere 128 genes. The other researcher mapped genes in a simple parasite and estimated that for this organism, 800 genes are plenty to do the job—but that anything short of 100 wouldn't be enough.

Although the numbers don't match precisely, those predictions "are not all that far apart," especially in comparison to the 75,000 genes in the human genome, notes Siv Andersson of Uppsala University in Sweden, who arrived at the 800 number. But coming up with a consensus answer may be more than just a genetic numbers game, particularly as more and more genomes are completely mapped and sequenced. "It may be a way of organizing any newly sequenced genome," explains Arcady Mushegian, a computational molecular biologist at the National Center for Biotechnology Information (NCBI) in Bethesda, Maryland. Comparing an



* Genome Mapping and Sequencing, Cold Spring Harbor, New York, May 8 to 12.

Stripping down. Computer analysis yields an estimate of the minimum modern and ancient genomes.

SCIENCE • VOL. 272 • 24 MAY 1996

How do we find the word-topic associations in each document?

How do we use them to learn topics in the given text collection?

How do we learn low-dim document representations in terms of the topics they represent?

A More Fine-Grained Mixture Model for Text

- Assume a corpus-level topic mixing proportions $\boldsymbol{\alpha}$ ($K \times 1$ prob vector)
- Also assume doc-level topic mixing props $\boldsymbol{\theta}_d$ ($K \times 1$ prob vector)
- Instead of assuming a single cluster $\boldsymbol{z}_d$ for doc $\boldsymbol{d}$, cluster each word in it
  - $\boldsymbol{z}_{d,n} \in \{1,2,\ldots,K\}$ denotes the cluster/topic of word $\boldsymbol{w}_{d,n} \in \{1,2,\ldots,V\}$
- Can obtain the "average" clustering for doc $\boldsymbol{d}$ using $\boldsymbol{\theta}_d$ or $\bar{\boldsymbol{z}}_d = \frac{1}{N_d}\sum_{n=1}^{N_d} \boldsymbol{z}_{d,n}$

Each assumed a one-hot $K \times 1$ vector

- The generative model is as follows

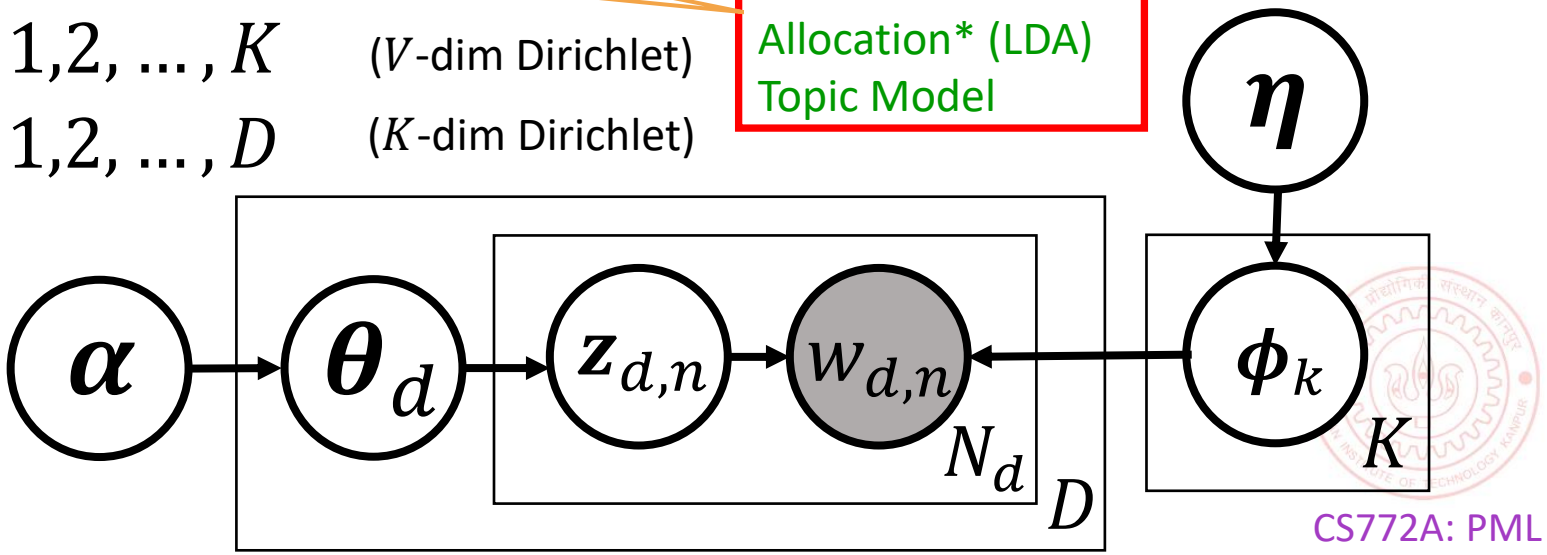
$$\phi_k \sim \text{Dirichlet}(\boldsymbol{\eta}) \quad k = 1,2,\ldots,K \quad (V\text{-dim Dirichlet})$$
$$\theta_d \sim \text{Dirichlet}(\boldsymbol{\alpha}) \quad d = 1,2,\ldots,D \quad (K\text{-dim Dirichlet})$$

Locally-conjugate. Easy Gibbs sampling, VI, etc

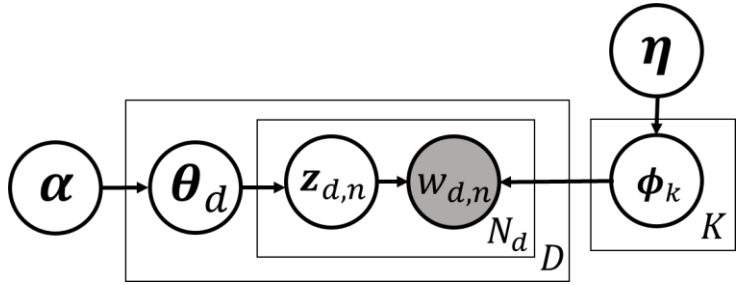
Latent Dirichlet Allocation* (LDA) Topic Model

Somewhat similar to Dir-Mult PCA model

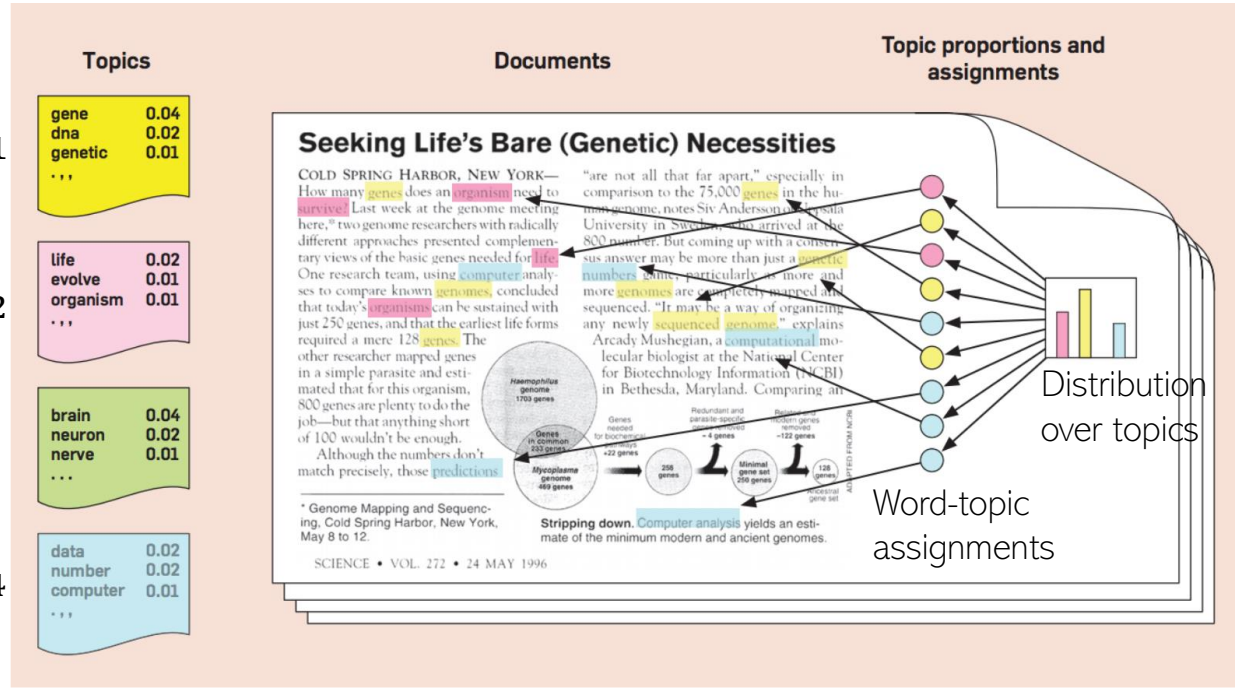
$$\boldsymbol{z}_{d,n} \sim \text{multinoulli}(\boldsymbol{\theta}_d)$$
$$\boldsymbol{w}_{d,n} \sim \text{multinoulli}(\phi_{z_{d,n}})$$



Latent Dirichlet Allocation (LDA)

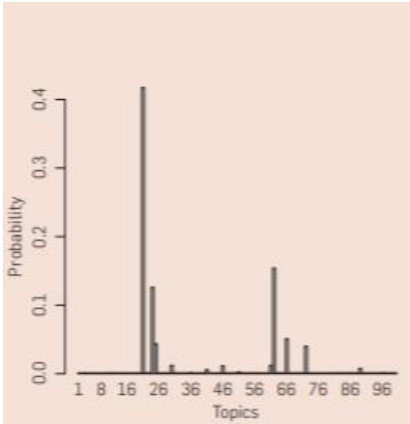


- A very widely used probabilistic model for text data
- Nice and easy insights into the text collection
  - Each $\phi_k = [\phi_{k1}, \ldots, \phi_{kV}]$ can be interpreted as topic (ϕ_{kv} = prob. of word v in topic k)
  - $\theta_d = [\theta_{d1}, \ldots, \theta_{dK}]$: how much each topic is present in document d (topic distribution)
  - $\bar{z}_d = \frac{1}{N_d}\sum_{n=1}^{N_d} z_{d,n}$ also has a similar interpretation as θ_d



A topic is a set of words that tend to co-occur together

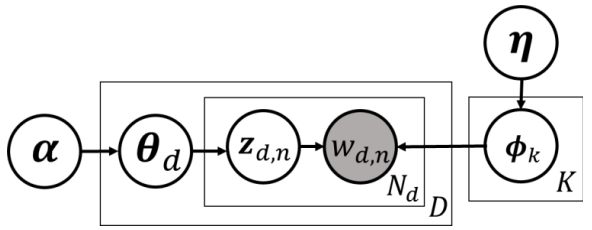
15 most frequent (most probable) words from four most prominent topics in this doc



"Genetics"	"Evolution"	"Disease"	"Computers"
human	evolution	disease	computer
genome	evolutionary	host	models
dna	species	bacteria	information
genetic	organisms	diseases	data
genes	life	resistance	computers
sequence	origin	bacterial	system
gene	biology	new	network
molecular	groups	strains	systems
sequencing	phylogenetic	control	model
map	living	infectious	parallel
information	diversity	malaria	methods
genetics	group	parasite	networks
mapping	new	parasites	software
project	two	united	new
sequences	common	tuberculosis	simulations

Topic distribution for the document on left

LDA: Inference and Evaluation



- LDA is locally conjugate. Many inference methods (VI, variational EM, Gibbs samp, etc)

$$p(\mathbf{Z}, \Theta, \Phi | \mathbf{W}, \alpha, \eta) = \frac{p(\mathbf{W}|\Phi, \mathbf{Z}) p(\mathbf{Z}|\Theta) p(\Phi|\eta) p(\Theta|\alpha)}{p(\mathbf{W}|\alpha, \eta)} \quad \text{(assuming hyperparams } \alpha, \eta \text{ are fixed)}$$

  - Can even collapse some variables and do collapsed Gibbs or collapsed VB
    - E.g., collapse $\boldsymbol{\theta_d}$ and $\boldsymbol{\phi_k}$ (if needed, these can be approximated using $\mathbf{Z}$)
- Many ways to evaluate how well LDA performs on some data
  - Extrinsic measures: Perform LDA and use its output for another task (e.g., classification)
  - Perplexity is another intrinsic measure to evaluate LDA-style models

Lower is better

Test set with $\boldsymbol{M}$ docs

Marginal likelihood of all words in the $\boldsymbol{d^{th}}$ test doc

$$perplexity(D_{\text{test}}) = \exp\left\{ -\frac{\sum_{d=1}^{M} \log p(\mathbf{w}_d)}{\sum_{d=1}^{M} N_d} \right\}$$

LDA: Limitations and Extensions

- LDA assumes topics remain static over time (improvement: Dynamic Topic Model)

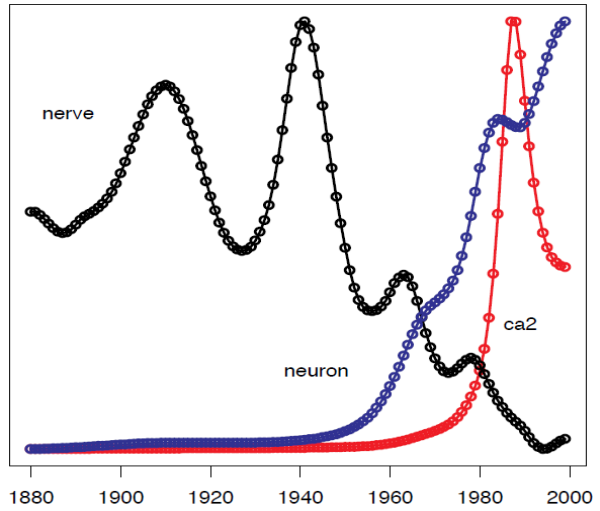
Assume a first-order Markov evolution for each topic w.r.t. time

$$w_k^t \sim \mathcal{N}(w_k^{t-1}, \sigma^2 I) \qquad \phi_k^t = \mathcal{S}(w_k^t)$$

Simplex transformation (convert w_k^t into a probability vector)

1881	1890	1900	1910	1920	1930	1940	1950	1960	1970	1980	1990	2000
brain	movement	brain	movement	movement	stimulate	record	respons	response	respons	cell	cell	neuron
movement	eye	eye	brain	sound	muscle	nerve	record	stimulate	cell	neuron	channel	active
action	right	movement	sound	muscle	sound	stimulate	stimulate	record	potential	response	neuron	brain
right	hand	right	nerve	active	movement	response	nerve	condition	stimul	active	ca2	cell
eye	brain	left	active	nerve	response	muscle	muscle	active	neuron	brain	active	fig
hand	left	hand	muscle	stimulate	nerve	electrode	active	potential	active	stimul	brain	response
left	action	nerve	left	fiber	frequency	active	frequency	stimulus	nerve	muscle	receptor	channel
muscle	muscle	vision	eye	reaction	fiber	brain	electrode	nerve	eye	system	muscle	receptor
nerve	sound	sound	right	brain	active	fiber	potential	subject	record	nerve	respons	synapse
sound	experiment	muscle	nervous	response	brain	potential	study	eye	abstract	receptor	current	signal

Evolution of topic "Neuroscience" (learned from the journal Science)



- LDA assumes topics are uncorrelated (improvement: Corr-LDA)
  - Use a logistic normal distribution on θ_d (cov matrix of log-normal makes component correlated)
- LDA ignores the sequential structure in the text (improvement: HMM-LDA)

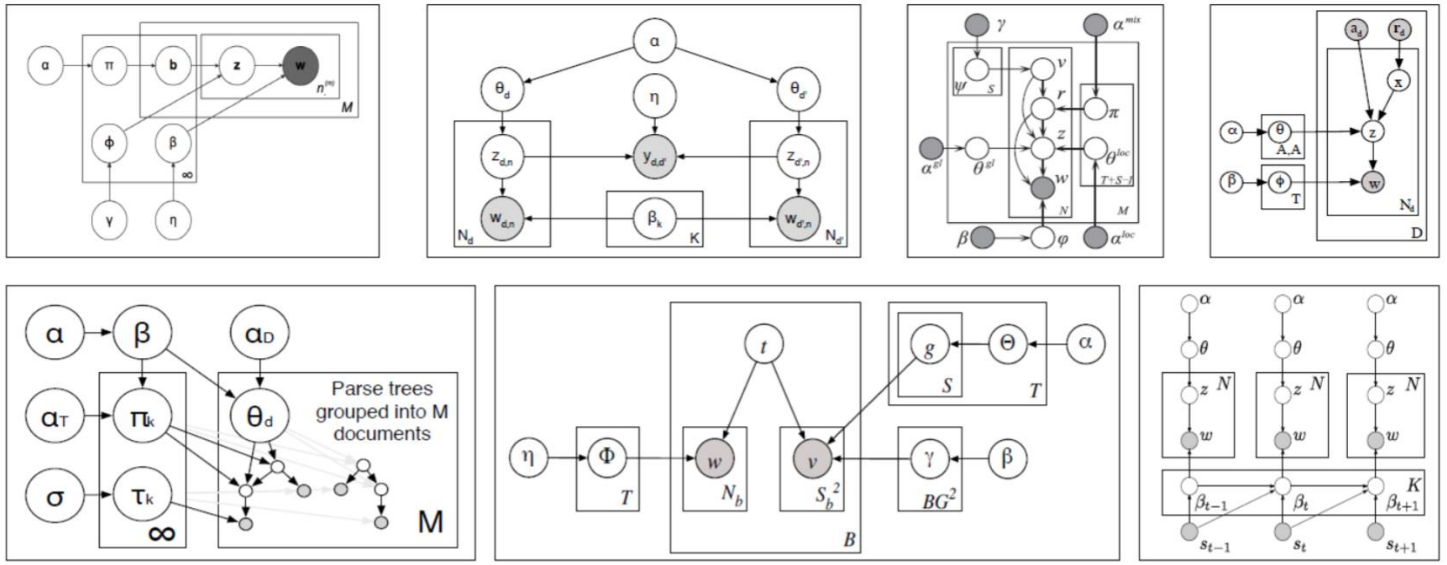
Fig courtesy: Dynamic Topic Models (Blei and Lafferty, 2006)

LDA Extensions (Contd)

Also: "Neural" Topic Models are popular nowadays (z to x mapping and vice-versa modeled via deep nets)

- LDA for non-text data, e.g., images
  - Each image can be represented as a bag of "visual words" and LDA can be applied
- Supervised/Labeled LDA (when we have have a label for each document)
- LDA for paired/multimodality data (e.g., images and text caption)
- LDA for graph-structured data instead of documents

Plate diagrams for some LDA extensions



LDA is also equivalent to doing a non-negative matrix fact. of the $V \times D$ word-document matrix $\mathbf{X}$ using a Poisson likelihood model*

$$\mathbf{X} \sim \text{Poisson}(\mathbf{\Phi\Theta})$$

$\mathbf{\Phi}$ $(V \times K)$ and $\mathbf{\Theta}$ $(K \times D)$ can be given any non-negative priors (Dirichlet/gamma)

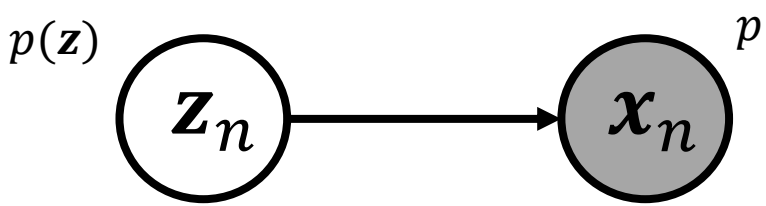
This can be extended to "deep" matrix factorization** (modeling $\mathbf{\Theta}$ using many layers)

*Sec 4 and 5 of "Beta-Negative Binomial Process and Poisson Factor Analysis" (Zhou et al, 2012)

** Poisson-gamma belief networks" (Zhou et al, 2015)

Constructing Generative Models using Neural Nets

- We can use a neural net to define the mapping from a K-dim $\boldsymbol{z}_n$ to D-dim $\boldsymbol{x}_n$



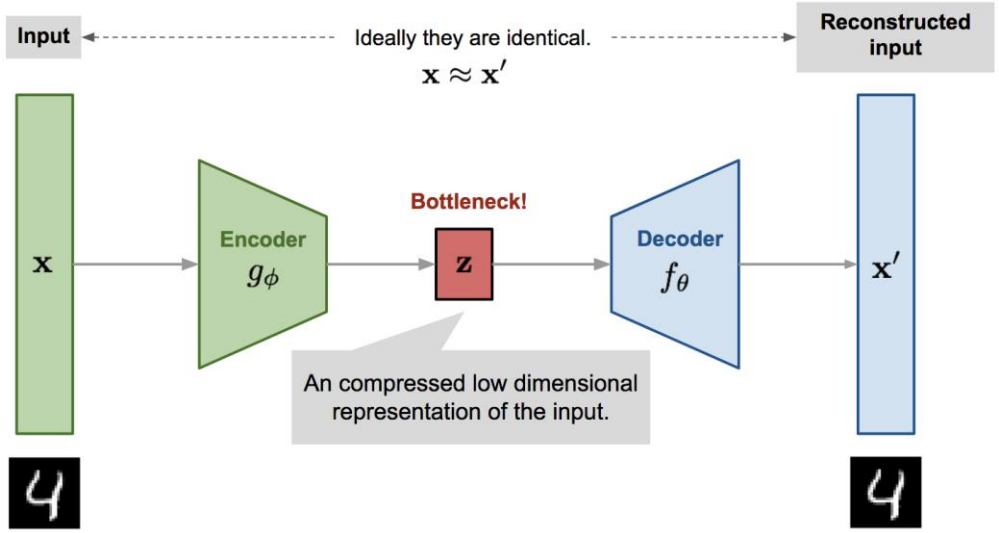
$$\boldsymbol{x}_n \sim \mathcal{N}(\text{NN}(\boldsymbol{z}_n;\mathbf{W}), \sigma^2\mathbf{I}_D)$$
$$\boldsymbol{x}_n \sim \text{ExpFam}(\text{NN}(\boldsymbol{z}_n;\mathbf{W}))$$

- If $\boldsymbol{z}_n$ has a Gaussian prior, such models are called deep linear Gaussian models (DLGM)
- Since NN mapping can be very powerful, DLGM can generate very high-quality data
  - Take the trained network, generate a random $\boldsymbol{z}$ from prior, pass it through the model to generate $\boldsymbol{x}$

Some sample images generated by Vector Quantized Variational AutoEncoder (VQ-VAE), a state-of-the-art DLGM

Variational Autoencoder (VAE)

- VAE* is a probabilistic extension of autoencoders (AE)



$$L_{\text{AE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2$$

- The basic difference is that VAE assumes a prior $\boldsymbol{p(z)}$ on the latent code $\boldsymbol{z}$
  - This enables it to not just compress the data but also generate synthetic data
  - How: Sample $\boldsymbol{z}$ from a prior and pass it through the decoder
- Thus VAE can learn good latent representation + generate novel synthetic data
- The name has "Variational" in it since it is learned using VI principles

*Autoencoding Variational Bayes (Kingma and Welling, 2013)

Pic source: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

Variational Autoencoder (VAE)

- VAE has three main components
  - A prior $p_\theta(z)$ over latent codes
  - A probabilistic decoder $p_\theta(x|z)$
  - A posterior or probabilistic encoder $p_\theta(z|x)$ approx. by an "inference network" $q_\phi(z|x)$

Here θ collectively denotes all the parameters of the prior and likelihood

Using the idea of "Amortized Inference" (next slide)

- VAE is learned by maximizing the ELBO

Here ϕ collectively denotes all the parameters that define the inference network

ELBO for a single data point

$$\mathcal{L}(\theta, \phi|\mathbf{x}) = \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z}|\mathbf{x})\right]$$

$$= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_\theta(\mathbf{x}|\mathbf{z})\right] - \mathbb{KL}\left(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})\right)$$

Maximized to find the optimal θ and ϕ

q_ϕ should reconstruct the data x well from z (high log-lik)

q_ϕ should also be simple (close to the prior)

- The Reparametrization Trick is commonly used to optimize the ELBO
- Posterior is inferred only over z, and usually only point estimate on θ and ϕ

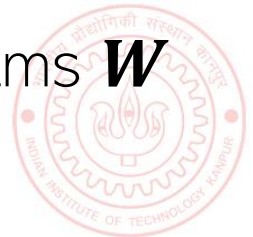
Amortized Inference

- Latent variable models need to infer the posterior $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$ for each observation $\boldsymbol{x}_n$
- This can be slow if we have lots of observations because
  1. We need to iterate over each $p(\boldsymbol{z}_n|\boldsymbol{x}_n)$
  2. Learning the global parameters needs wait for step 1 to finish for all observations
- One way to address this is via Stochastic VI (already saw)
- Amortized inference is another appealing alternative (used in VAE and other LVMs too)

$$p(\boldsymbol{z}_n|\boldsymbol{x}_n) \approx q(\boldsymbol{z}_n|\boldsymbol{\phi}_n) = q(\boldsymbol{z}_n|\text{NN}(\boldsymbol{x}_n;\boldsymbol{W}))$$

If q is Gaussian then the NN will output a mean and a variance

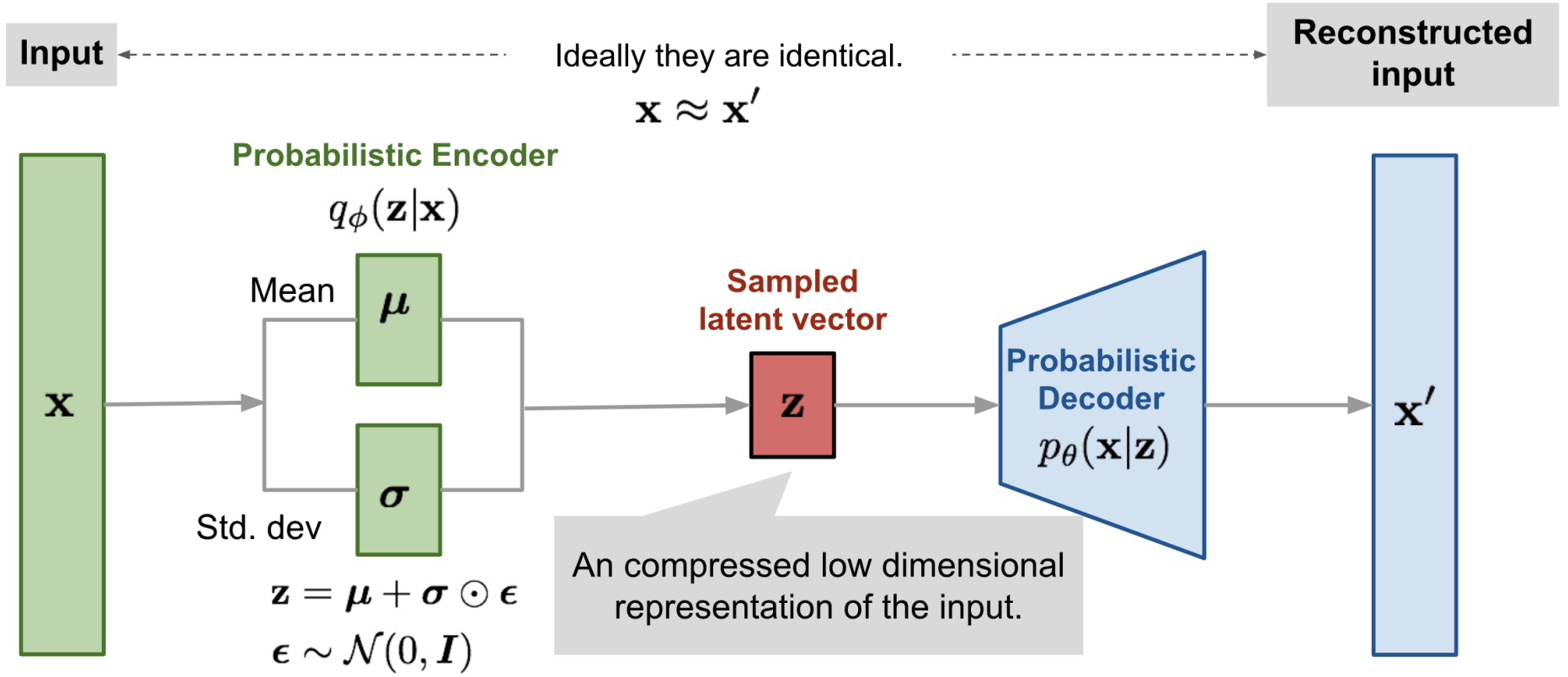
- Thus no need to learn $\boldsymbol{\phi}_n$'s (one per data point) but just a single NN with params $\boldsymbol{W}$
  - This will be our "encoder network" for learning $\boldsymbol{z}_n$
  - Also very efficient to get $p(\boldsymbol{z}_*|\boldsymbol{x}_*)$ for a new data point $\boldsymbol{x}_*$



Variational Autoencoder: The Complete Pipeline

- Both probabilistic encoder and decoder learned jointly by maximizing the ELBO

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right]$$
$$= \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z})\right)$$



Pic source: https://lilianweng.github.io/lil-log/2018/08/12/from-autoencoder-to-beta-vae.html

VAE and Posterior Collapse

Decoder is a neural net and can be arbitrarily powerful making this term very large

Consequently, KL will become close to zero collapsing posterior to the prior

- VAEs may suffer from posterior collapse

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \mathbb{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})\right)$$

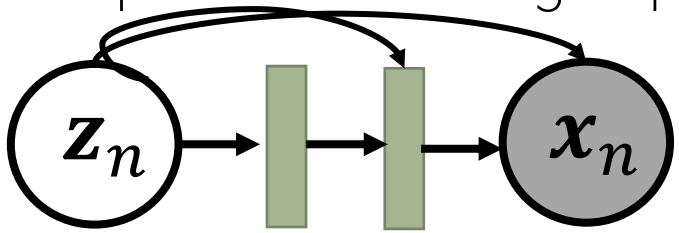
- Thus, due to posterior collapse, reconstruction will still be good but the code $\boldsymbol{z}$ may be garbage (not useful as a representation for $\boldsymbol{x}$)
- Several ways to prevent posterior collapse, e.g.,
  - Use KL annealing

A carefully tuned value between 0 and 1

$$\mathcal{L}(\boldsymbol{\theta}, \boldsymbol{\phi}|\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})\right] - \boldsymbol{\beta}\,\mathbb{KL}\left(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z})\right)$$

For example, keep the variance of $\boldsymbol{q}$ as fixed

  - Avoid KL from becoming 0 using some $\boldsymbol{q}$ doesn't collapse to the prior
  - More tightly couple $\boldsymbol{z}$ with $\boldsymbol{x}$ using skip-connections (Skip-VAE)



Besides these, MCMC (sometimes used for inference in VAE), or improved VI techniques can also help in preventing posterior collapse in VAEs