Bayesian Deep Learning (contd), (Shallow and Deep) Generative Models

CS772A: Probabilistic Machine Learning

Piyush Rai

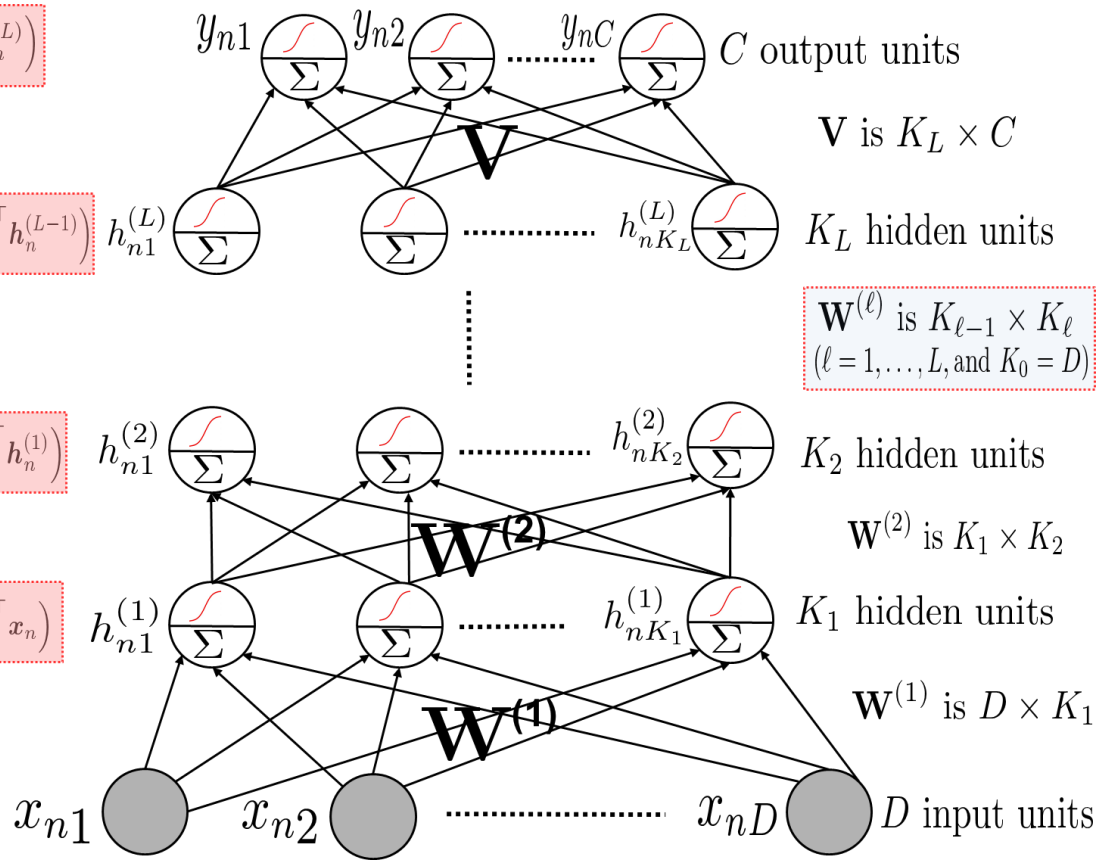
(Deep) Neural Networks

- These are nonlinear function approximators
- Consists of an input layer, one or more hidden layers, and an output layer

Can think of the last hidden layer's node values being used as features in a GLM (linear/logistic/softmax, etc) modeled by the output layer

Hidden layers act as feature extractors

Network weights typically learned by backpropagation (basically, gradient descent + chain rule)

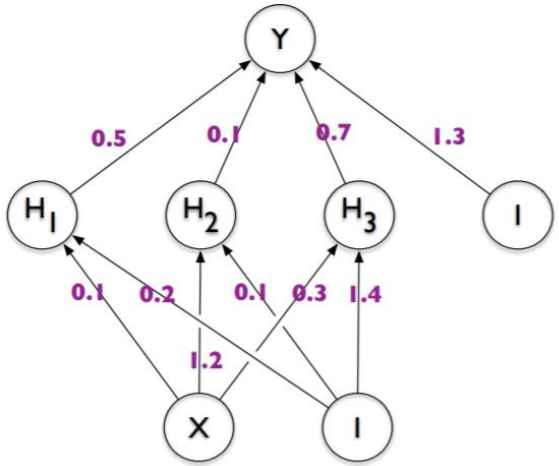


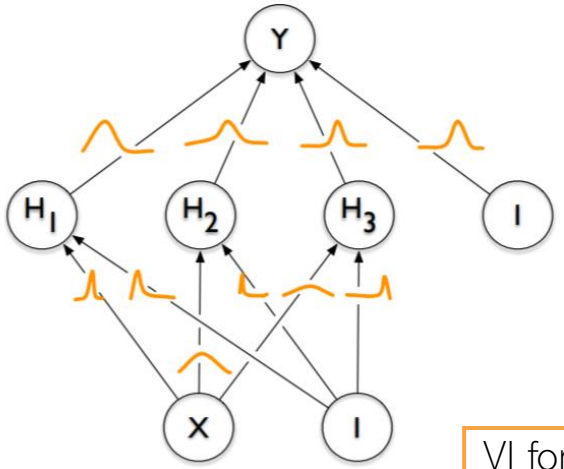
Bayesian Neural Networks

- Backprop for neural nets only gives us point estimates for the weights
- Another alternative is to be Bayesian and learn the posterior distribution over weights

Standard neural net: Each weight has a fixed value, learned by backprop

Note: Just having a likelihood and prior will still give us a standard neural net if we choose to do MLE/MAP only





Bayesian neural net: Each weight has a posterior distribution inferred by some Bayesian inference algo (VI/MCMC/Laplace approx., etc)

Also, test time will require computing PPD, not just a plug-in prediction

$$\mathbf{w}^{\text{MLE}} = \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w})$$
$$= \arg\max_{\mathbf{w}} \sum_i \log P(\mathbf{y}_i|\mathbf{x}_i, \mathbf{w})$$
$$\mathbf{w}^{\text{MAP}} = \arg\max_{\mathbf{w}} \log P(\mathbf{w}|\mathcal{D})$$
$$= \arg\max_{\mathbf{w}} \log P(\mathcal{D}|\mathbf{w}) + \log P(\mathbf{w})$$

VI for Bayesian neural net

Using reparametrization trick (known as "Bayes by Backprop"* in this context), BBVI etc

$$\theta^\star = \arg\min_\theta \mathrm{KL}[q(\mathbf{w}|\theta)||P(\mathbf{w}|\mathcal{D})]$$
$$= \arg\min_\theta \int q(\mathbf{w}|\theta) \log \frac{q(\mathbf{w}|\theta)}{P(\mathbf{w})P(\mathcal{D}|\mathbf{w})} \mathbf{dw}$$
$$= \arg\min_\theta \mathrm{KL}\left[q(\mathbf{w}|\theta) \,||\, P(\mathbf{w})\right] - \mathbb{E}_{q(\mathbf{w}|\theta)}\left[\log P(\mathcal{D}|\mathbf{w})\right]$$

Pic from: *Weight Uncertainty in Neural Networks (Blundell et al, 2015)

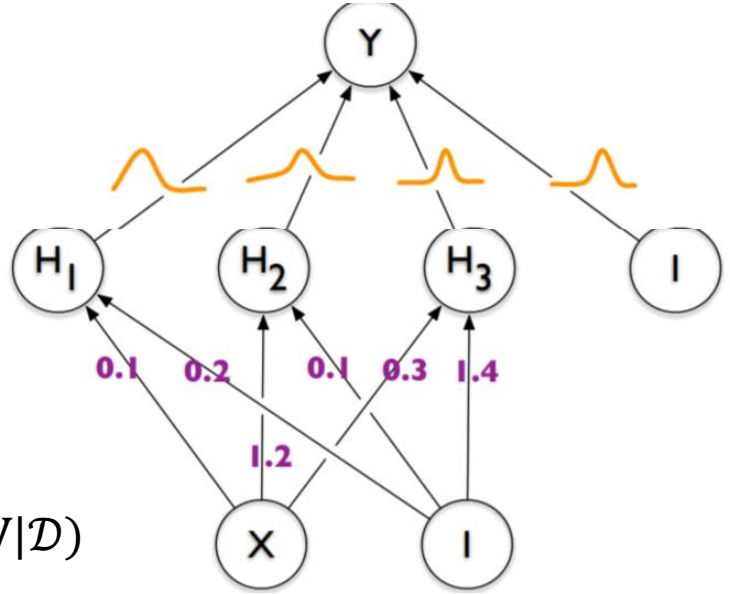
A Hybrid Bayesian Neural Net

$$p(y_*|x_*, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y_*|x_*, \theta^{(s)}) \quad \text{where } \theta^{(s)} \sim p(\theta|\mathcal{D})$$

- Learning the posterior for all weights can be expensive
- PPD computation is also slow if using Monte Carlo approximation for PPD
- A cheaper practical alternative is
  - Do point estimation for hidden layer weights ($\mathbf{W}$)
  - Infer the full posterior for output layer weights ($\mathbf{V}$)
  - The PPD will then be

$$p(y_*|x_*, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y_*|x_*, \mathbf{V}^{(s)}, \widehat{\mathbf{W}}) \quad \text{where } \mathbf{V}^{(s)} \sim p(\mathbf{V}|\mathcal{D})$$

Faster because the posterior of $\mathbf{V}$ is much lower dimensional



- A rough approximation of the above is the following
  - Use a pretrained neural net to extract feature
  - Train Bayesian linear model (e.g., Bayesian linear/logistic/softmax/GLM reg.) on these features

Approximation since in the hybrid approach, we still learn $\mathbf{W}$ and $\mathbf{V}$ together, unlike this approach where it is a two-step process

Bayesian Neural Networks: The Priors

- Zero-mean isotropic Gaussian priors are common and convenient
  - Corresponds to weight-decay or ℓ_2 regularizer
- Another alternative is to use sparsity-inducing priors, e.g.,

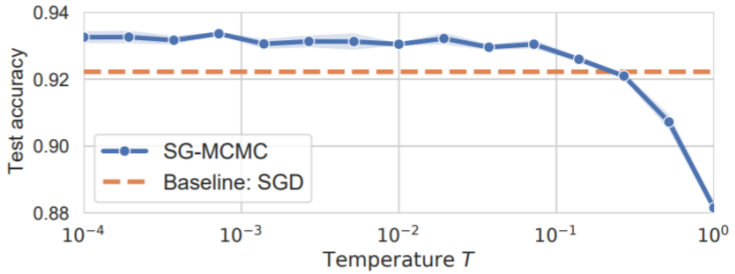
$$p(\mathbf{w}) = \prod_j \pi \mathcal{N}(w_j|0,\sigma_1^2) + (1-\pi)\mathcal{N}(w_j|0,\sigma_2^2) \qquad \sigma_1 > \sigma_2 \text{ and } \sigma_2 \ll 1$$

- Gaussian priors have been found somewhat problematic in recent work
  - Cold-posterior effect

$$\log p(w|x,y)^{\frac{1}{T}} = \frac{1}{T}[\log p(y|w,x) + \log p(w)] + Z(T)$$

T is like temperature

$T = 1$ is the standard Bayesian inference



Recent work has shown that BNNs with standard Gaussian priors work poorly for $T = 1$ but $T \ll 1$ improves performance

Maybe Gaussian priors aren't really ideal??

Pic from: *How Good is the Bayes Posterior in Deep Neural Networks Really? (Wenzel et al, 2020)

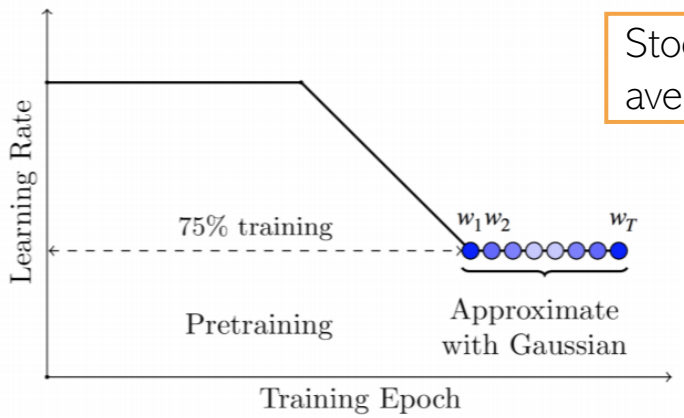
Other Inference Methods for Bayesian Neural Nets

- Laplace approximation is very common: $p(W|\mathcal{D}) \approx \mathcal{N}(W_{MAP}, \mathbf{H}^{-1})$
  - However, can be slow since the number of parameters is very large
  - One option is to use a simpler covariance matrix (e.g,, diagonal or block-diag)
  - Another option is to use the hybrid Bayesian neural net
    - Use MAP estimates for the hidden layer weights
    - Use Laplace approximation only for the output layer weights

Extension: A mixture of Gaussian approximation: Multi-SWAG – Run SGD M times and use a mixture of M such Gaussians

SWA based Gaussian approximation: SWAG

- Using SGD iterates obtained from backprop



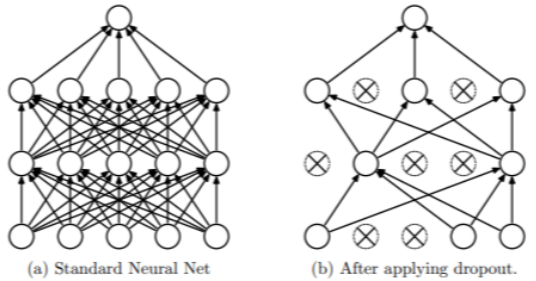
Stochastic weight averaging (SWA)

$$p(w|\mathcal{D}) \approx q(w|\mathcal{D}) = \mathcal{N}(\bar{w}, K)$$

$$\bar{w} = \frac{1}{T}\sum_t w_t\,, \quad K = \frac{1}{2}\left(\frac{1}{T-1}\sum_t (w_t - \bar{w})(w_t - \bar{w})^{\mathrm{T}} + \frac{1}{T-1}\sum_t diag(w_i - \bar{w})^2\right)$$

Pic from: *A Simple Baseline for Bayesian Uncertainty in Deep Learning (Maddox et al, 2019)

Other Inference Methods for Bayesian Neural Nets



- Monte Carlo Dropout is another popular and efficient way
- Standard Dropout
  - Drop some weights randomly (with some "drop" probability) during training
  - At test time, multiply each weight by the "keep" probability
  - Note: Dropout applied only at training time
- Monte Carlo Dropout*

$$p(y_*|x_*, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y_*|x_*, \theta^{(s)})$$

where $\theta^{(s)} \sim p(\theta|\mathcal{D})$

Can be seen as learning a variational approximation of the weights (see paper for details, if interested)

$$p(y_*|x_*, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} p(y_*|x_*, \theta^{(s)})$$

where $\theta^{(s)} = \epsilon^{(s)} \odot \hat{\theta}$

Vector of Bernoulli or Gaussian noise

Elementwise product

Point estimate

*Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning (Gal and Ghahramani, 2016)

Other Inference Methods for Bayesian Neural Nets

- SGMCMC methods like SGLD and SGHMC are also used nowadays (very efficient)

$$\theta^{(t)} = \theta^{(t-1)} + \frac{\eta_t}{2}\nabla_\theta[\log p(\mathcal{D}|\theta) + \log p(\theta)]\big|_{\theta^{(t-1)}} + \epsilon_t$$

- Recently, SGMCMC with cyclic step sizes (cSGLD) was proposed (Zhang et al, 2020)
  - Use big steps to explore different modes
  - Use small steps later to sample once a mode is localized

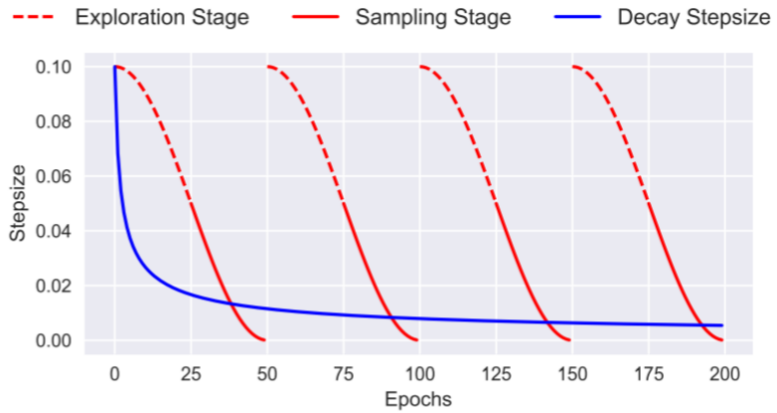
Step size in iteration k

$$\alpha_k = \frac{\alpha_0}{2}\left[\cos\left(\frac{\pi \bmod(k-1, \lceil K/M \rceil)}{\lceil K/M \rceil}\right) + 1\right]$$

K is the total number of iterations and M is the number of cycles

(a) Target (b) SGLD (c) cSGLD

A complex mixture of Gaussian distributions



	CIFAR-10	CIFAR-100
SGD	5.29±0.15	23.61±0.09
SGDM	5.17±0.09	22.98±0.27
Snapshot-SGD	4.46±0.04	20.83±0.01
Snapshot-SGDM	4.39±0.01	20.81±0.10
SGLD	5.20±0.06	23.23±0.01
cSGLD	4.29±0.06	20.55±0.06
SGHMC	4.93±0.1	22.60±0.17
cSGHMC	**4.27**±0.03	**20.50**±0.11

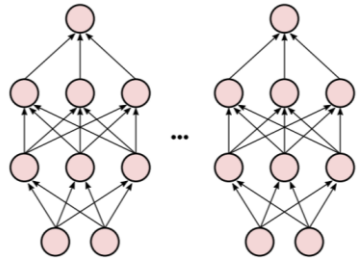
Pic from: *Cyclical Stochastic Gradient MCMC for Bayesian Deep Learning (Zhang et al, 2020)

Deep Ensembles

- Most inference methods tend to produce local approximations only
  - VI methods typically learn an approximation around one of the modes
  - Sampling methods may give most samples near one of the modes (though in principle they may explore other modes as well)
  - Thus the uncertainties may be underestimated in general

Both VI and Sampling may be prone to capturing only a single "Basin of attraction"

- Deep Ensembles* is a method that tries to address this issue
  - Train the network M times with different seeds and permutations of training data
  - Denote the learned weights by $\theta_1, \theta_2, \ldots, \theta_M$ (assuming these are M modes)
  - Approximate the posterior by the following



$$p(\theta|\mathcal{D}) = \frac{1}{M}\sum_{m=1}^{M} \delta_{\theta_m}(\theta)$$

Akin to Bayesian Model Averaging using M models

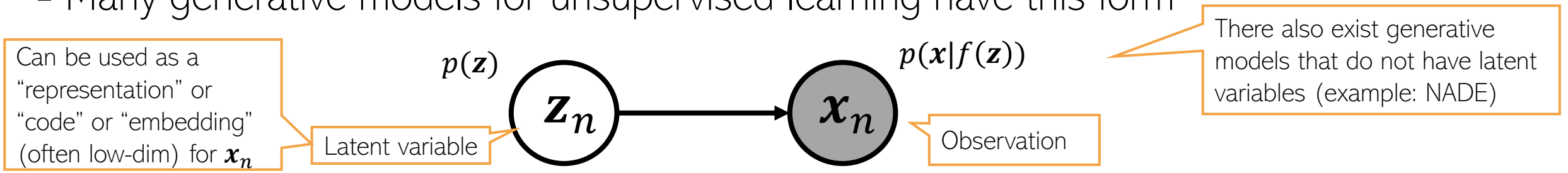
  - This approach is considered non-Bayesian but often performs better (in terms of more diversity in the set of parameters learned) than other inference methods

*Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles (Lakshminarayanan, 2017)

Deep Generative Models (for unsupervised learning)

Generative Models for Unsupervised Learning

- Many generative models for unsupervised learning have this form



- Depending on the prior, likelihood, and f, various latent factor models arise, e.g.,
  - Factor Analysis and Probabilistic PCA: $p(x|f(z)) = N(x|Wz, \Sigma)$
  - Gaussian Process Latent Variable Models (GPLVM) – f is nonlinear modeled by a GP
  - Deep generative models (constructed using deep neural nets)
    - Variational Autoencoders (VAE) - f is nonlinear modeled by a neural net
    - Generative Adversarial Network (GAN) – f is nonlinear modeled by a neural net and the likelihood is only implicitly defined
    - Denoising Diffusion Models
    - .. and several others..

Some Classical Models

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

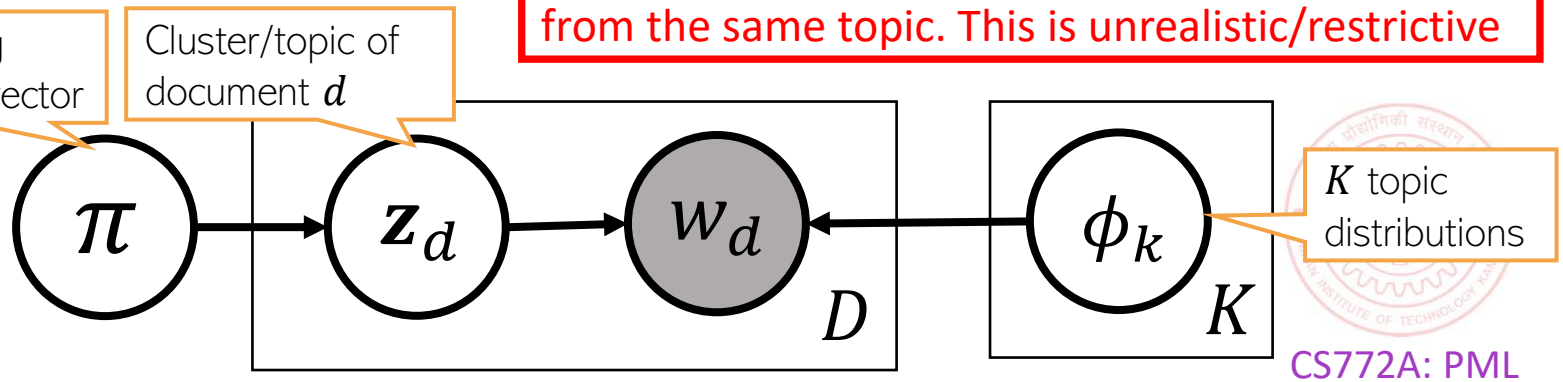
Each representing a "topic" (K topics)

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

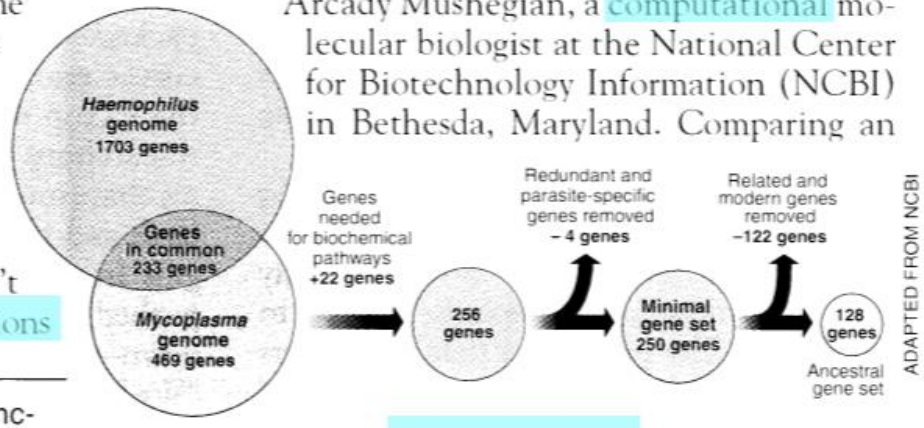


Documents can be about multiple topics

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

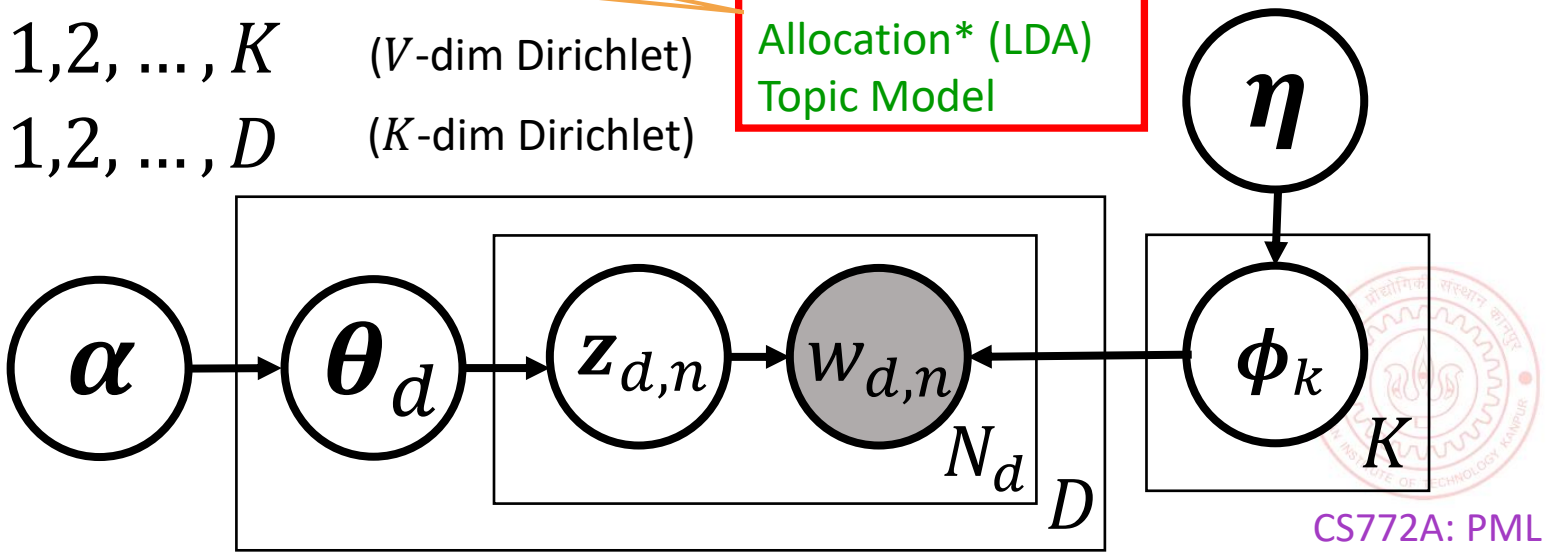
$$\phi_k \sim \text{Dirichlet}(\boldsymbol{\eta}) \quad k = 1,2,\ldots,K \quad (V\text{-dim Dirichlet})$$
$$\theta_d \sim \text{Dirichlet}(\boldsymbol{\alpha}) \quad d = 1,2,\ldots,D \quad (K\text{-dim Dirichlet})$$

Locally-conjugate. Easy Gibbs sampling, VI, etc

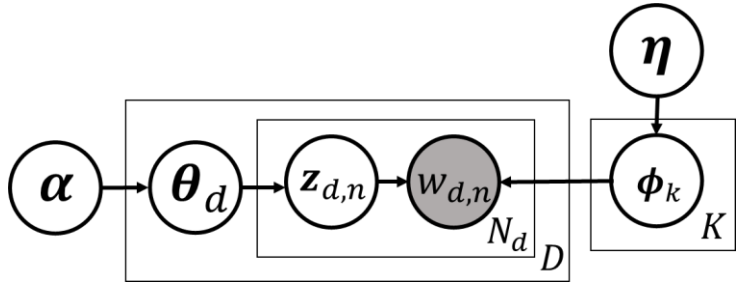
Somewhat similar to Dir-Mult PCA model

Latent Dirichlet Allocation* (LDA) Topic Model

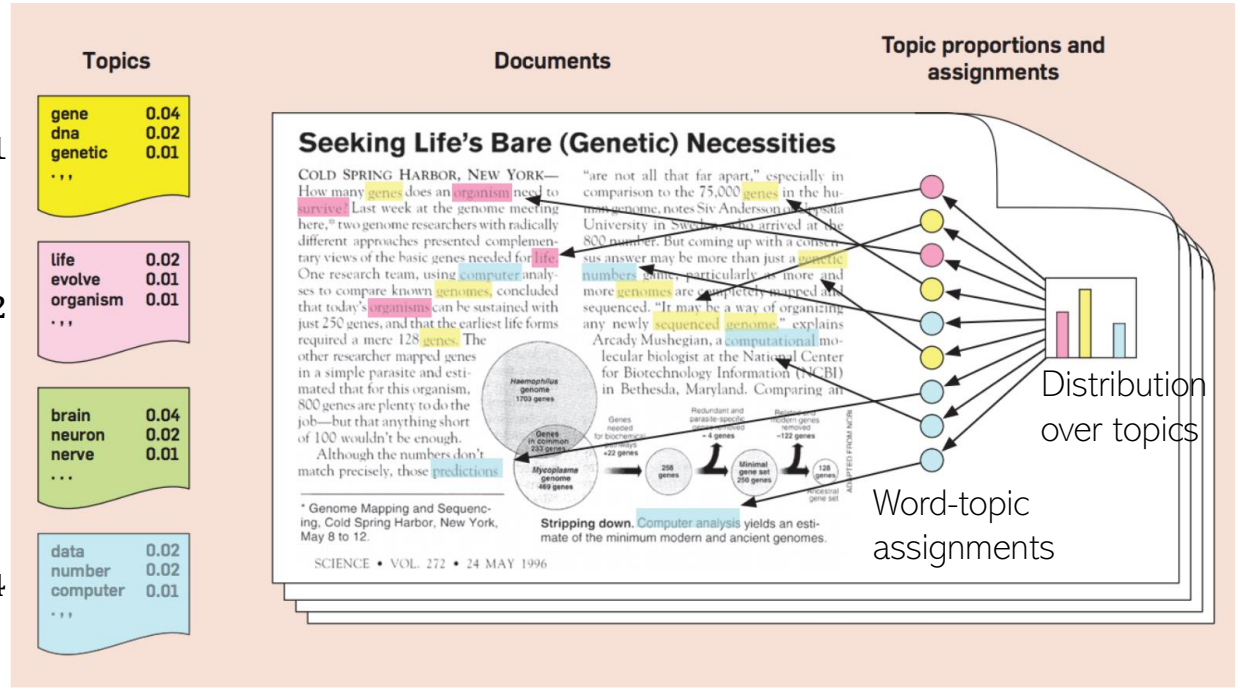
$$\boldsymbol{z}_{d,n} \sim \text{multinoulli}(\boldsymbol{\theta}_d)$$
$$\boldsymbol{w}_{d,n} \sim \text{multinoulli}(\phi_{z_{d,n}})$$



Latent Dirichlet Allocation (LDA)

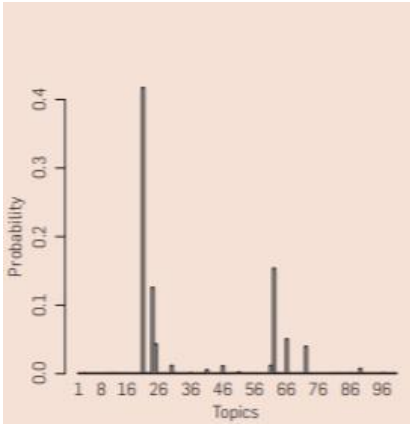


- A very widely used probabilistic model for text data
- Nice and easy insights into the text collection
  - Each $\phi_k = [\phi_{k1}, \ldots, \phi_{kV}]$ can be interpreted as topic (ϕ_{kv} = prob. of word v in topic k)
  - $\theta_d = [\theta_{d1}, \ldots, \theta_{dK}]$: how much each topic is present 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

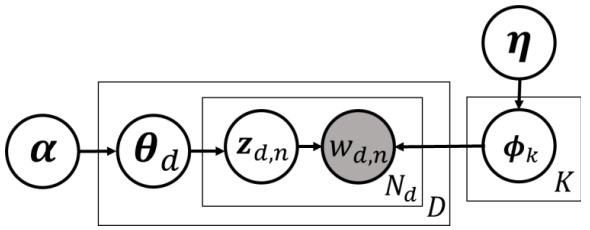
15 most frequent (most probable) words from four most prominent topics in this doc



"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)

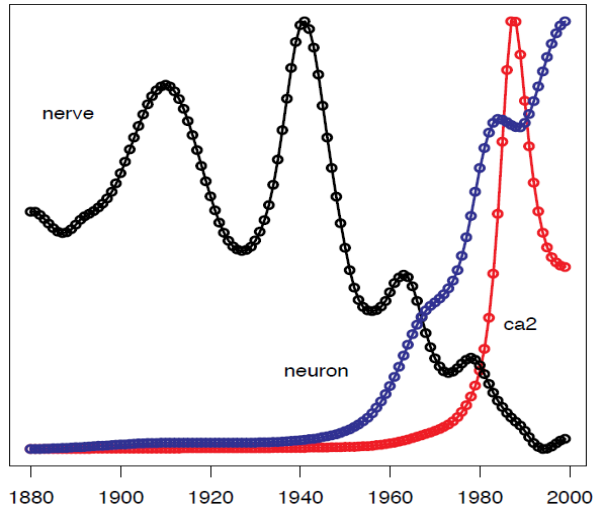
Assume a first-order Markov evolution for each topic w.r.t. time

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

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)



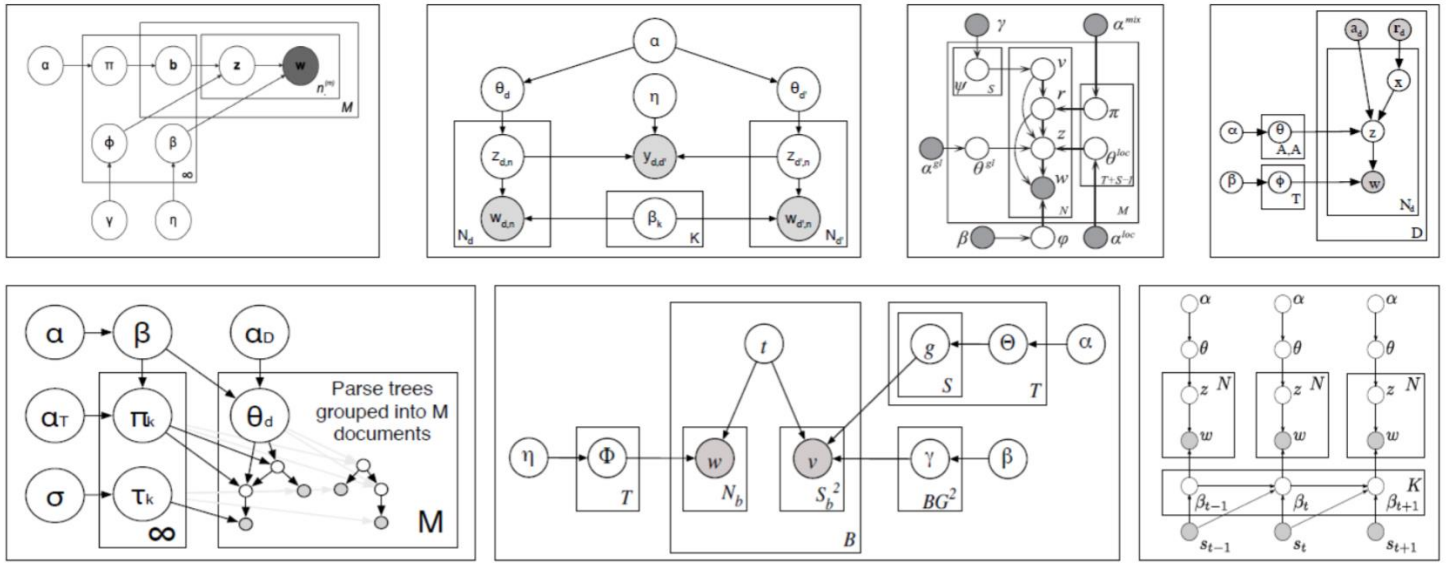
- LDA assumes topics are uncorrelated (improvement: Corr-LDA)
  - Use a logistic normal distribution on θ_d (cov matrix of log-normal makes component correlated)
- LDA ignores the sequential structure in the text (improvement: HMM-LDA)

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

LDA Extensions (Contd)

- 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)

Next Class

- Generative models using deep neural networks
  - Variational Autoencoders
  - Generative Adversarial Networks
  - Denoising Diffusion Models