Group Meeting - October 02, 2020

Paper review & Research progress

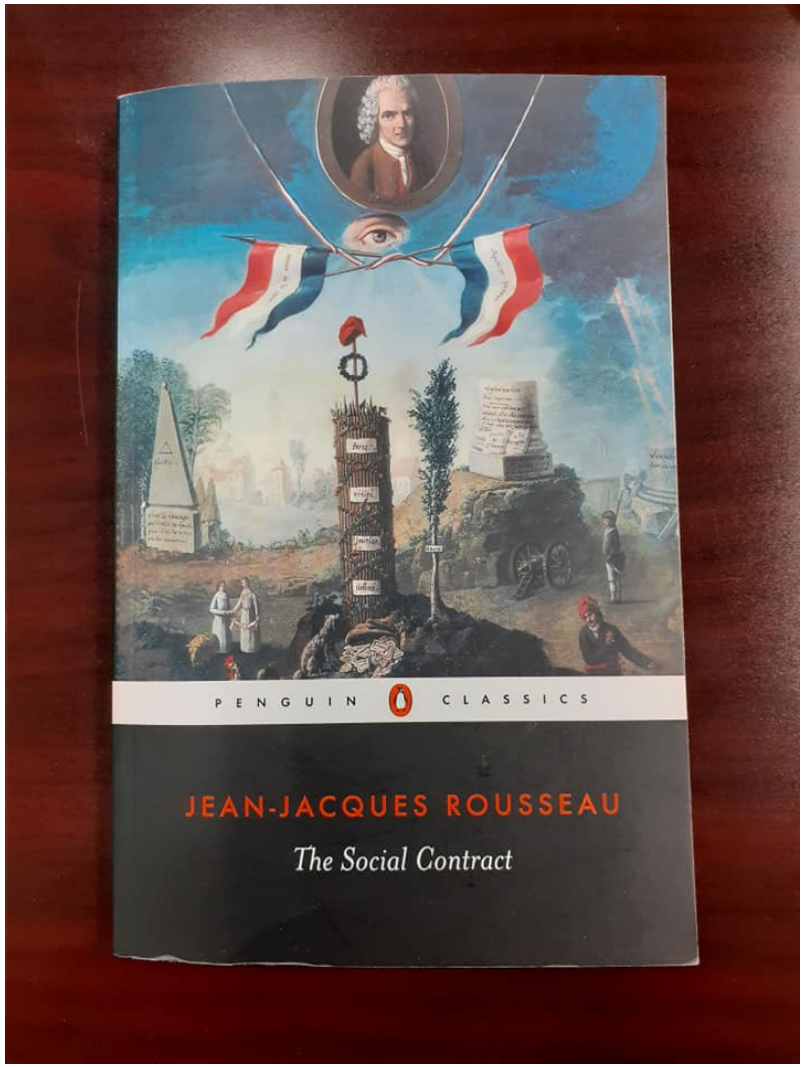
Truong Son Hy *

*Department of Computer Science
The University of Chicago

Ryerson Physical Lab

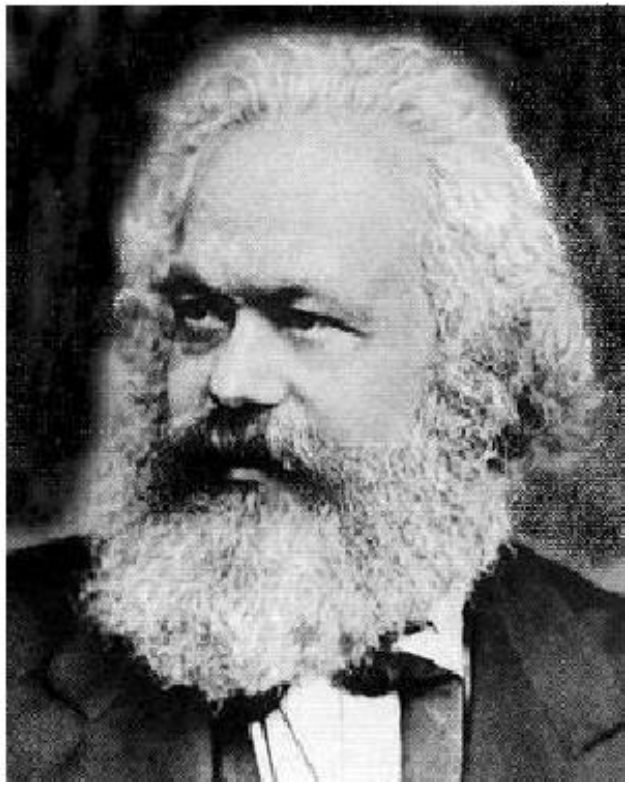
Rousseau – *The Social Contract*

Man was born free, and he is everywhere in chains.



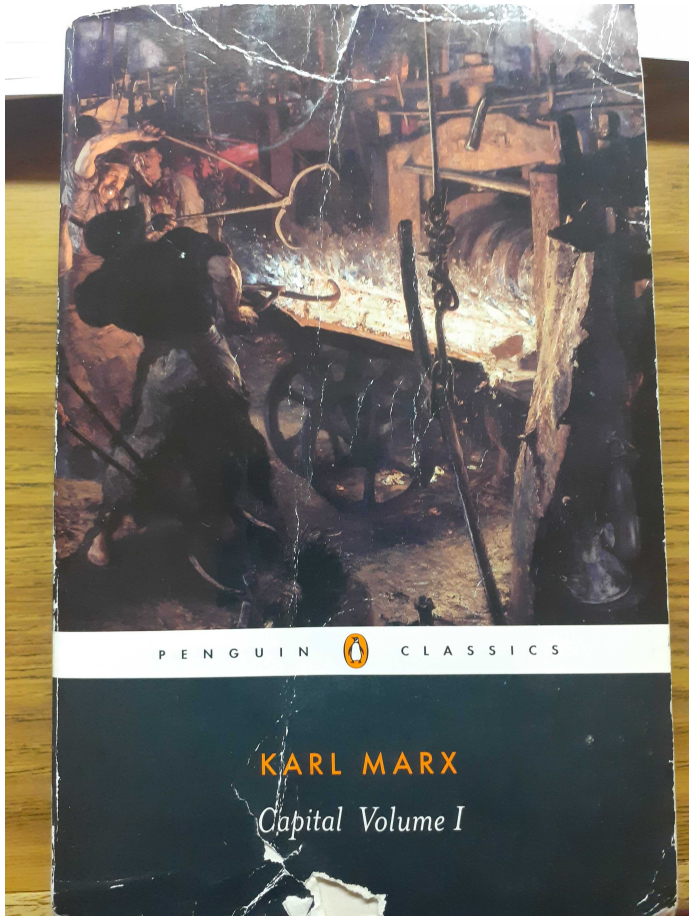
Adam Smith – *The Wealth of Nations*

In general, if any branch of trade, or any division of labour, be advantageous to the public, the freer and more general the competition, it will always be the more so.



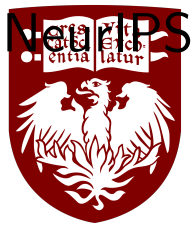
Karl Marx – *Das Kapital*

As exchange-values, all commodities are merely definite quantities of congealed labour-time.



Papers – Stochastic Variational Inference

1. **Composing graphical models with neural networks for structured representations and fast inference** (NIPS 2016)
   https://arxiv.org/pdf/1603.06277.pdf
2. **Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders** (spin-off from paper 1)
   http://datta.hms.harvard.edu/wp-content/uploads/2018/01/pub_24.pdf
3. **Neural Variational Inference for Text Processing** (ICML 2016)
   https://arxiv.org/pdf/1511.06038.pdf
4. **Learning to Generate with Memory** (ICML 2016)
   https://arxiv.org/pdf/1602.07416.pdf
5. **Generated Loss, Augmented Training, and Multiscale VAE**
   https://arxiv.org/pdf/1904.10446.pdf
6. **Learning Correlated Latent Representations with Adaptive Priors** (NeurIPS 2019 workshop)
   http://bayesiandeeplearning.org/2019/papers/47.pdf



3D Learning & Generation (planned for October 9)

1. **PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation** (CVPR 2017)
2. **Learning Representations and Generative Models for 3D Point Clouds** (ICML 2018)
3. **PointFlow: 3D Point Cloud Generation with Continuous Normalizing Flows** (ICCV 2019)
4. **3D Point Cloud Generative Adversarial Network Based on Tree Structured Graph Convolutions** (ICCV 2019)
5. **Learning Localized Generative Models for 3D Point Clouds via Graph Convolution** (ICLR 2019)
6. **Learning Gradient Fields for Shape Generation** (ECCV 2020)
7. **Progressive Point Cloud Deconvolution Generation Network** (ECCV 2020)
8. **Discrete Point Flow Networks for Efficient Point Cloud Generation** (ECCV 2020)

3D Learning & Generation (planned for October 9)

Open question

What are we going to do with the 3D structure?

- For (molecular) Atom3D
- For generic (Computer Vision) point cloud

Partial (under construction) slides for October 09:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___October_09__2020.pdf`

Maybe I will present **Neural Turing Machines** to synchronize with the topic of **differentiable program**(!)

Research – Implementation Progress

Graph generation (as part of LibCCNs):

- Dynamic batching, NO patched/dummy vertices added
- Scale better
- Based on VAE with CCN encoder
- Tested with synthetic graphs (tree, star, ER)
- Tested with graph kernel benchmark datasets (DD, ENZYMES, MUTAG, PTC, etc.)
- `https://github.com/HyTruongSon/LibCCNs/tree/master/pytorch/small_graphs`

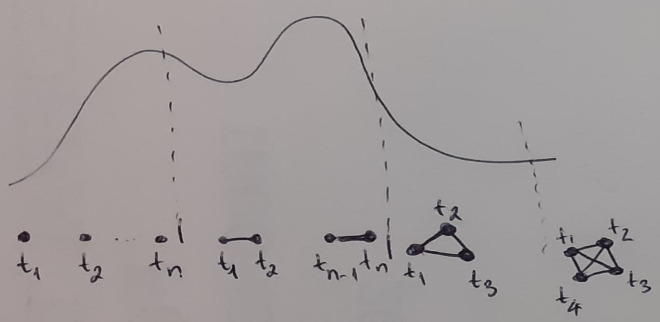
Room for CUDA extension:

- `https://github.com/HyTruongSon/LibCCNs/tree/master/cuda`

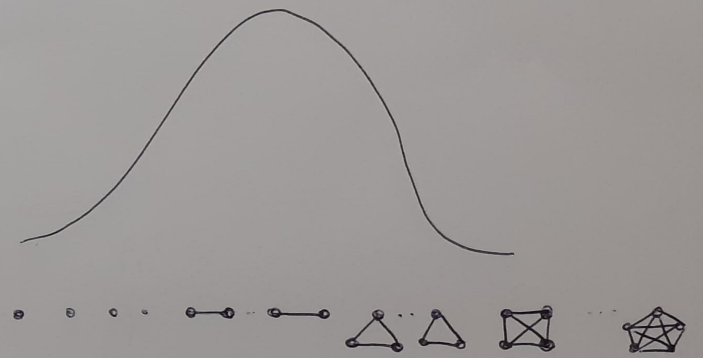
Research – Theory Progress (1)

Research – Theory Progress (2)

Way 2: Distribution over all possible cliques.



or.



Question: Cliques overlapping in the construction.

Papers 1

Composing graphical models with neural networks for structured representations and fast inference (NIPS 2016)

Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, Ryan P. Adams

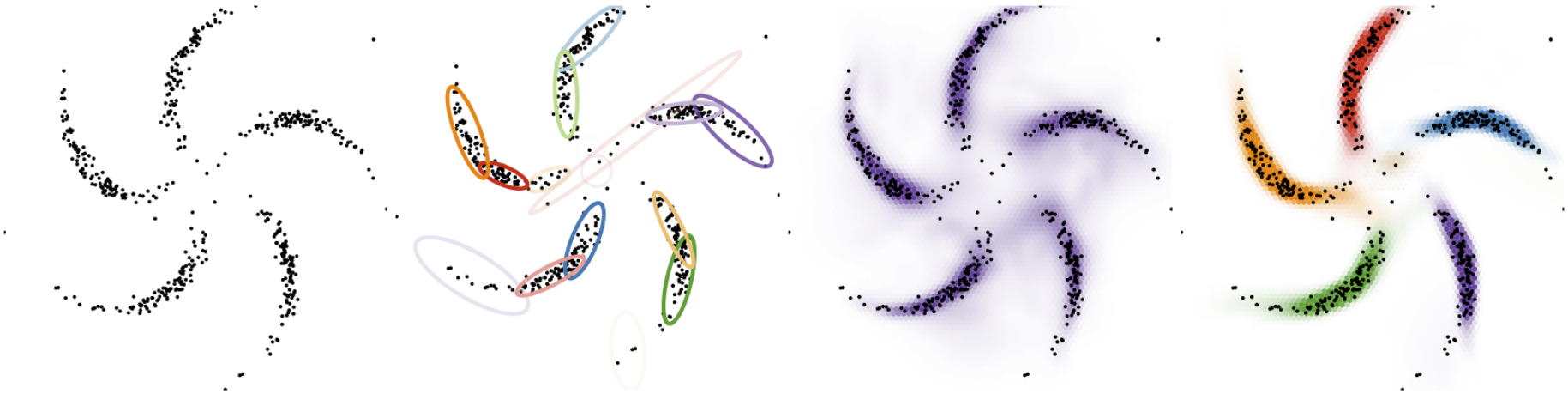
`https://arxiv.org/pdf/1603.06277.pdf`

Composing graphical models – Proposals

Proposals

- A general modeling and inference framework that combines:
  1. **Probabilistic graphical models**: latent graphical models, message-passing algorithms.
  2. **Deep learing methods**: neural network observation likelihoods, recognition networks to produce local evidence potentials.
- Single stochastic variational inference objective.
- Application to automatically segmenting and categorizing mouse behavior from raw depth video.

Warped mixtures for arbitrary cluster shapes (1)



(a) Data (b) GMM (c) Density net (VAE) (d) GMM SVAE

Figure 1: Comparison of generative models fit to spiral cluster data. See Section 2.1.

Note:

- GMM is a generative model, but does not have the representation power of neural networks.
- VAE does not have the power of a mixture model.

Warped mixtures for arbitrary cluster shapes (2)

Consider the problem of modeling the data $y = \{y_n\}_{n=1}^N$. A standard approach to find the clusters in data is to fit a Gaussian Mixture Model (GMM):

$$\pi \sim \text{Dir}(\alpha)$$
$$(\mu_k, \Sigma_k) \sim_{\text{iid}} \text{NIW}(\lambda)$$
$$z_n|\pi \sim_{\text{iid}} \pi$$
$$y_n|z_n, \{(\mu_k, \Sigma_k)\}_{k=1}^K \sim_{\text{iid}} \mathcal{N}(\mu_{z_n}, \Sigma_{z_n})$$

Note

For Dirichlet Process (DP), Dirichlet Process (Gaussian) Mixture Model (DPMM), and Latent Dirichlet Allocation (LDA), please check our past discussion in:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___August_28__2020.pdf`

Warped mixtures for arbitrary cluster shapes (3)

Instead of using a GMM, a more flexible alternative would be a neural network density model:

$$\gamma \sim p(\gamma)$$

$$x_n \sim_{\text{iid}} \mathcal{N}(\mathbf{0}, \mathbf{1})$$

$$y_n | x_n, \gamma \sim_{\text{iid}} \mathcal{N}(\mu(x_n; \gamma), \Sigma(x_n; \gamma))$$

where $\mu(x_n; \gamma)$ and $\Sigma(x_n; \gamma)$ depend on x_n through some smooth (differential) parametric function (e.g. MLP), and $p(\gamma)$ is a Gaussian prior.

Limitation

The neural network density model fits the data well, but does not explicitly represent discrete mixture components.

Warped mixtures for arbitrary cluster shapes (4)

Composing a latent GMM with nonlinear observations:

$$\pi \sim \text{Dir}(\alpha)$$

$$(\mu_k, \Sigma_k) \sim_{\text{iid}} \text{NIW}(\lambda)$$

$$\gamma \sim p(\gamma)$$

$$z_n|\pi \sim_{\text{iid}} \pi$$

$$x_n \sim_{\text{iid}} \mathcal{N}(\mu_{z_n}, \Sigma_{z_n})$$

$$y_n|x_n, \gamma \sim_{\text{iid}} \mathcal{N}(\mu(x_n;\gamma), \Sigma(x_n;\gamma))$$

Latent (switching) linear dynamical systems (1)

Note

For an introduction of linear dynamical systems including Kalman Filter and its variants (exteneded KF and unscented KF), please check the past discussion in:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/EKF___UFK.pdf`

Sequence modeling

Consider the problem of **generatively modeling video** (that is a sequence of image frames):

- High-dimensional images $\{y_n\}_{n=1}^N$.
- Low-dimensional latent variables $\{x_n\}_{n=1}^N$.

Latent (switching) linear dynamical systems (2)

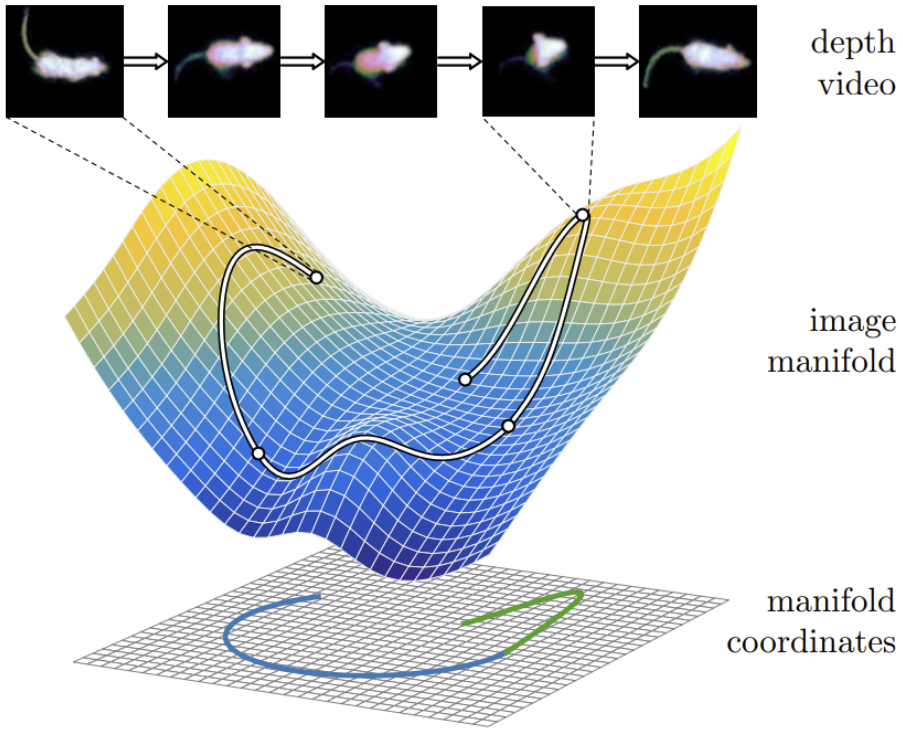


Figure 1. Illustration of unsupervised generative modeling applied to depth video of a mouse. Each video frame lies near a low-dimensional image manifold, and we wish to learn a parameterization of this manifold in which trajectories can be modeled with switching linear dynamics.

Latent (switching) linear dynamical systems (3)

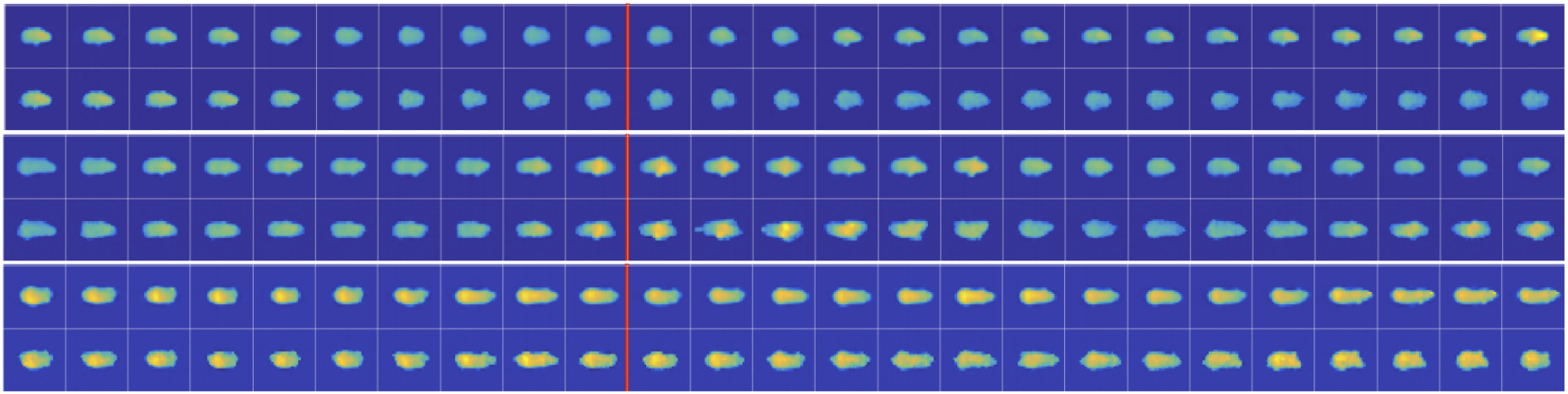


Figure 6: Predictions from an LDS SVAE fit to depth video. In each panel, the top is a sampled prediction and the bottom is real data. The model is conditioned on observations to the left of the line.

Latent (switching) linear dynamical systems (4)

We introduce dependence through time between the latent Gaussian samples $\{x_n\}_{n=1}^N$. Each latent variable x_{n+1} depend on the previous latent variable x_n through a Gaussian linear dynamical system:

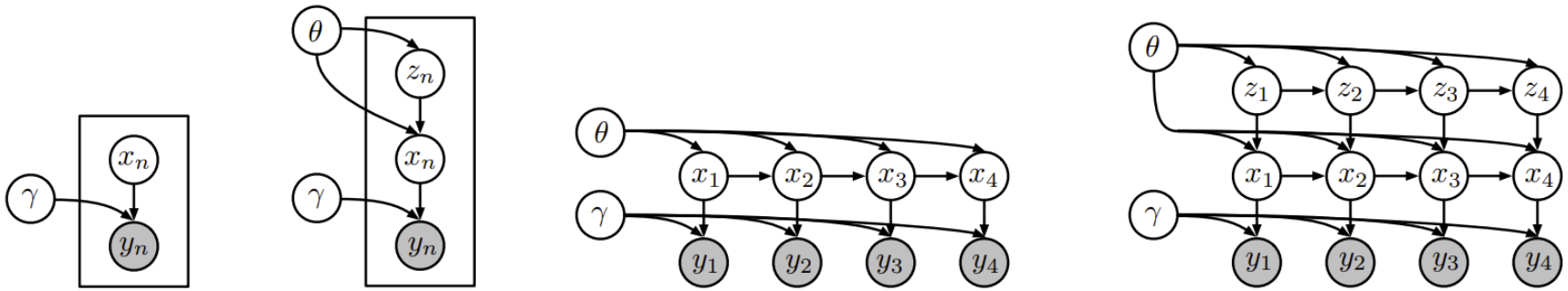
$$x_{n+1} = Ax_n + Bu_n, \qquad u_n \sim_{\text{iid}} \mathcal{N}(\mathbf{0}, \mathbf{1}), \qquad A, B \in \mathbb{R}^{m \times m}$$

We can construct a **latent switching linear dynamical system** (SLDS) to represent the data in terms of continuous latent states that evolve according to a discrete library of Markovian linear dynamics:

$$z_{n+1}|z_n, \pi \sim \pi_{z_n}, \qquad x_{n+1} = A_{z_n}x_n + B_{z_n}u_n, \qquad u_n \sim_{\text{iid}} \mathcal{N}(\mathbf{0}, \mathbf{1})$$

where $n \in \{1, 2, .., N\}$ denotes time, $z_n \in \{1, 2, .., K\}$ denotes discrete-valued latent state, and $\pi = \{\pi_k\}_{k=1}^K$ denotes the Markov transition matrix and $\pi_k \in \mathbb{R}_+^K$ is its k-th row.

Latent (switching) linear dynamical systems (5)



(a) Latent Gaussian (b) Latent GMM (c) Latent LDS (d) Latent SLDS

Figure 2: Generative graphical models discussed in Section 2.

Structured mean field inference (1)

Note

For an overview of variational inference in graphical models, please check our past discussion in:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___August_14__2020.pdf`

Consider learning a Gaussian linear dynamical system model with linear Gaussian observations:

$$x_n = Ax_{n-1} + Bu_{n-1}, \quad u_n \sim_{\text{iid}} \mathcal{N}(\mathbf{0}, \mathbf{1}), \quad y_n = Cx_n + Dv_n, \quad v_n \sim_{\text{iid}} \mathcal{N}(\mathbf{0}, \mathbf{1})$$

where:

- Latent states $x = \{x_n\}_{n=1}^N$ and observations $y = \{y_n\}_{n=1}^N$.
- Parameters $\theta = (A, B, C, D)$ with prior $p(\theta)$.

Structured mean field inference (2)

To approximate the posterior $p(\theta, x|y)$, consider the mean field family $q(\theta)q(x)$ and the variational inference objective:

$$\mathcal{L}[q(\theta)q(x)] = \mathbb{E}_{q(\theta)q(x)}\left[\log \frac{p(\theta)p(x|\theta)p(y|x,\theta)}{q(\theta)q(x)}\right]$$

We can optimize the variational family $q(\theta)q(x)$ to approximate the posterior $p(\theta, x|y)$ by maximizing $\mathcal{L}[q(\theta)q(x)]$. For fixed $q(\theta)$,

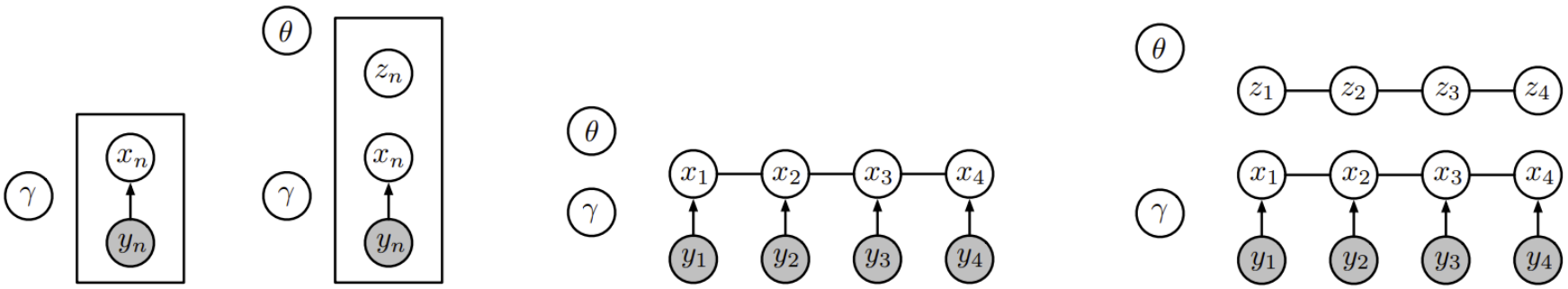
$$q^*(x) \triangleq \arg\max_{q(x)} \mathcal{L}[q(\theta)q(x)]$$

is computed by **exact** inference (message passing algorithms).

Note

Regarding the factorization **restriction** $p(\theta, x|y) \approx q(\theta)q(x)$, please check Chapter 10 (Section 10.1.1) of Bishop's textbook.

Structured variational autoencoders



(a) VAE (b) GMM SVAE (c) LDS SVAE (d) SLDS SVAE

Figure 3: Variational families and recognition networks for the VAE [7] and three SVAE examples.

Note: Paper 1 & 2 overlap a lot, so we continue to discuss Structured VAEs with paper 2.

More reference

[Matthew et al., 2014] **Stochastic Variational Inference for Bayesian Time Series Models** (ICML 2014)

`https://people.csail.mit.edu/mattjj/papers/icml2014svi.pdf`

Papers 2

Structured VAEs: Composing Probabilistic Graphical Models and Variational Autoencoders (spin-off from paper 1)

Matthew J. Johnson, David Duvenaud, Alexander B. Wiltschko, Sandeep R. Datta, Ryan P. Adams

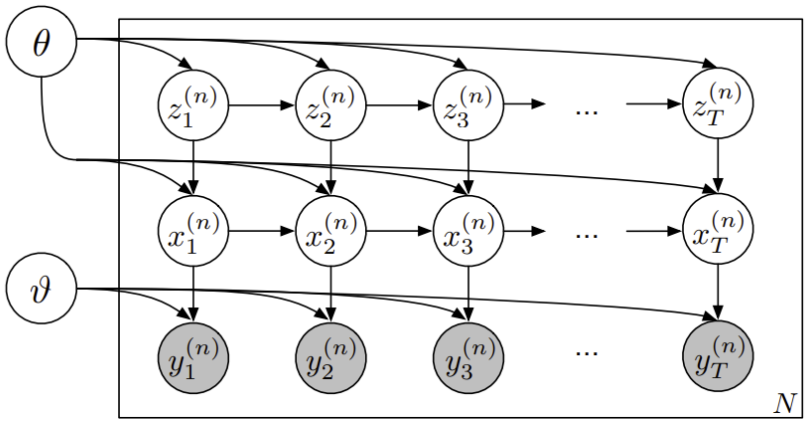
`http://datta.hms.harvard.edu/wp-content/uploads/2018/01/pub_24.pdf`

Structured VAEs – Proposals

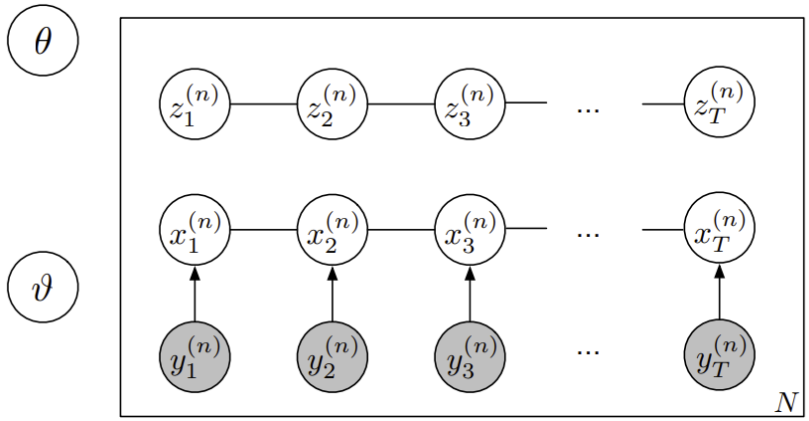
Proposals

- A new framework for unsupervised learning that combines:
  1. **Probabilistic graphical models**: to express structured probability distributions.
  2. **Deep learning methods**: to learn flexible feature models.
- Scalable fitting algorithms for:
  1. Natural gradient stochastic variational inference.
  2. Graphical model message passing.
  3. Backpropagation with the reparameterization trick.
- Application to structured time series model (e.g. mouse behavioral phenotyping).

Conditional Random Field (1)



(a) SLDS generative model with nonlinear observation model parameterized by ϑ.



(b) Structured CRF variational family with node potentials $\{\psi(x_t^{(n)}; y_t^{(n)}, \phi)\}_{t=1}^T$ parameterized by ϕ.

Figure 4. Graphical models for the SLDS generative model and corresponding structured CRF variational family.

Note: Between paper 1 & 2, there is a bit of notation change from γ to ϑ.

Conditional Random Field (2)

[Wainwright & Jordan, 2008] In mean field variational inference, one constructs a tractable variational family by **breaking dependencies** in the posterior. In this case, between the dynamics parameters θ, the observation parameters ϑ, the discrete state sequence $z = z_{1:T}$ and the continuous state sequence $x = x_{1:T}$:

$$q(\theta, \vartheta, z_{1:T}, x_{1:T}) = q(\theta)q(\vartheta)q(z_{1:T})q(x_{1:T})$$

- **Naive** mean field model continues to break the dependencies among $z_{1:T}$ and $x_{1:T}$.
- **Structured** mean field model **preserves** the correlations of these random variables **across time**.

Conditional Random Field (3)

[Murphy, 2012] we parameterize the factor $q(x_{1:T})$ as a Conditional Random Field (CRF) in terms of pairwise (edge) potentials and node potentials:

$$q(x_{1:T}) \propto \left(\prod_{t=1}^{T-1} \psi(x_t, x_{t+1}) \right) \left(\prod_{t=1}^{T} \psi(x_t; y_t, \phi) \right)$$

where $\psi(x_t, x_{t+1})$ is the directed pairwise potential, and the node potential $\psi(x_t; y_t, \phi)$ is a function of the observation y_t. **Remark:** without the direction, this is called as Markov Random Field (MRF).

Conditional Random Field (4)

We choose each node potential to be a Gaussian factor that has (we use this alternative representation instead of mean vector and covariance matrix):

- Precision matrix $J(y_t; \phi)$
- Potential vector $h(y_t; \phi)$

depending on the corresponding observation y_t through a neural network (MLP):

$$\psi(x_t; y_t, \phi) \propto \exp\left\{ -\frac{1}{2} x_t^T J(y_t; \phi) x_t + h(y_t; \phi)^T x_t \right\}$$

$$[h(y_t; \phi), J(y_t; \phi)] \leftarrow \text{MLP}(y_t; \phi)$$

These **local recognition networks** allow us to fit a regression from each observation y_t to a probabilistic guess at the corresponding latent state x_t.

Inference algorithm (1)

The optimization proceeds by alternating between optimizing $q(x_{1:T})$ and optimizing $q(z_{1:T})$. First, fixing the factor on the discrete states $q(z_{1:T})$:

$$q^*(x_{1:T}) \propto \exp\left\{ \log \psi(x_1) + \sum_{t=1}^{T-1} \log \psi(x_t, x_{t+1}) + \sum_{t=1}^{T} \log \psi(x_t; y_t) \right\}$$

where:

$$\psi(x_1) = \mathbb{E}_{q(\theta)q(z)}[\log p(x_1|z_1, \theta)]$$

$$\psi(x_t, x_{t+1}) = \mathbb{E}_{q(\theta)q(z)}[\log p(x_{t+1}|x_t, z_t, \theta)]$$

Inference algorithm (2)

Similarly, fixing $q(x_{1:T})$ the optimal factor $q^*(z_{1:T})$ is proportional to:

$$q^*(z_{1:T}) \propto \exp\left\{ \log \psi(z_1) + \sum_{t=1}^{T-1} \log \psi(z_t, z_{t+1}) + \sum_{t=1}^{T} \log \psi(z_t) \right\}$$

where:

$$\psi(z_1) = \mathbb{E}_{q(\theta)}[\log p(z_1|\theta)] + \mathbb{E}_{q(\theta)q(x)}[\log p(x_1|z_1, \theta)]$$

$$\psi(z_t, z_{t+1}) = \mathbb{E}_{q(\theta)}[\log p(z_{t+1}|z_t)]$$

$$\psi(z_t) = \mathbb{E}_{q(\theta)q(x)}[\log p(x_{t+1}|x_t, z_t, \theta)]$$

Inference algorithm (3)

Algorithm 2 Model inference subroutine for the SLDS

Input: Variational dynamics parameter $\widetilde{\eta}_\theta$ of $q(\theta)$, node potentials $\{\psi(x_t; y_t)\}_{t=1}^T$ from recognition network

function INFERENCE($\widetilde{\eta}_\theta$, $\{\psi(x_t; y_t)\}_{t=1}^T$)

Initialize factor $q(x)$

repeat

$q(z) \propto \exp\{\mathbb{E}_{q(\theta)q(x)} \ln p(z, x \mid \theta)\}$

$q(x) \propto \exp\{\mathbb{E}_{q(\theta)q(z)} \ln p(x \mid z, \theta)\} \prod_t \psi(x_t; y_t)$

until $q(z)$ and $q(x)$ converge

$\hat{x} \leftarrow$ sample $q(x)$

$\bar{t}_{zx} \leftarrow \mathbb{E}_{q(z)q(x)} t_{zx}(z, x)$

$\mathrm{KL} \leftarrow \mathrm{KL}(q(\theta) \,\|\, p(\theta)) + N\mathbb{E}_{q(\theta)} \mathrm{KL}(q(z)q(x) \,\|\, p(z, x \mid \theta))$

returnsample $\hat{x}$, expected stats $\bar{t}_{zx}$, divergence KL

end function

Inference algorithm (4)

To draw samples $\hat{x}_{1:T} \sim q(x_{1:T})$, we perform **message passing**. Given two neighboring nodes i and j, we define the message from i to j:

$$m_{i\to j}(x_j) \triangleq \int \psi(x_i; y_i)\psi(x_i, x_j)m_{k\to i}(x_i)dx_i$$

where k is the other neighbor of node i. Iteratively:

1. Sample $\hat{x}_1 \sim q(x_1)$ from:

$$q(x_1) \propto \psi(x_1)\psi(x_1; y_1)m_{2\to 1}(x_1)$$

2. Given sample $\hat{x}_{t-1}$, we sample $\hat{x}_t \sim q(x_t|\hat{x}_{1:t-1})$ from:

$$q(x_t|\hat{x}_{1:t-1}) \propto \psi(\hat{x}_{t-1}, x_t)\psi(x_t; y_t)m_{t+1\to t}(x_t)$$

3. Finally, we sample $\hat{x}_T \sim q(x_T|\hat{x}_{1:T-1})$ from:

$$q(x_T|\hat{x}_{1:T-1}) \propto \psi(\hat{x}_{T-1}, x_T)\psi(x_T; y_T)$$

To our research: sequence modeling $\rightarrow$ graph modeling

Hammersley-Clifford theorem

A positive distribution $p(\boldsymbol{x}) > 0$ satisfies the CI (conditional independent) properties of an undirected graph G iff p can be represented as a product of factors, one per **maximal clique**, i.e.,

$$p(\boldsymbol{x}|\boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c|\boldsymbol{\theta}_c)$$

where $\mathcal{C}$ is the set of all the (maximal) cliques of G, and $Z(\boldsymbol{\theta})$ is the **partition function** given by:

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{x}} \prod_{c \in \mathcal{C}} \psi_c(\boldsymbol{x}_c|\boldsymbol{\theta}_c)$$

Note that the partition function is to ensure the overall distribution sums to 1.

Paper 3

Neural Variational Inference for Text Processing (ICML 2016)

Yishu Miao, Lei Yu, Phil Blunsom

`https://arxiv.org/pdf/1511.06038.pdf`

Proposals

Proposals

- A generic variational inference framework for **generative** and **conditional** models of text.
- The proposed neural variational document model combines a **continuous stochastic document representation** with a bag-of-words generative model. (unsuperivised document generation)
- The proposed neural answer selection model employs a **stochastic representation layer within an attention mechanism** to extrat the semantics between a question and answer pair. (supervised question answering)

Latent Dirichlet Allocation (LDA)

Last time, we discussed about LDA (can be understood as 2-level Dirichlet Process Mixture Model) but it can only generate bag-of-words (without order).

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___August_28__2020.pdf`

Overview

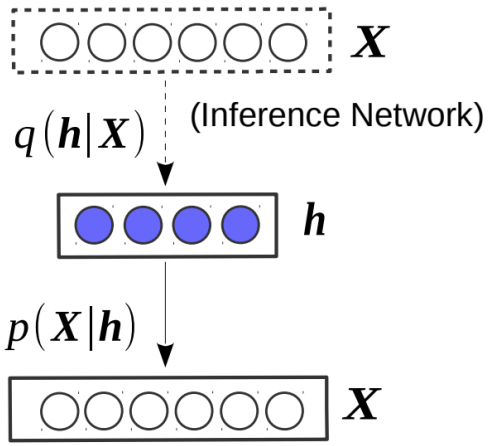


Figure 1. NVDM for document modelling.

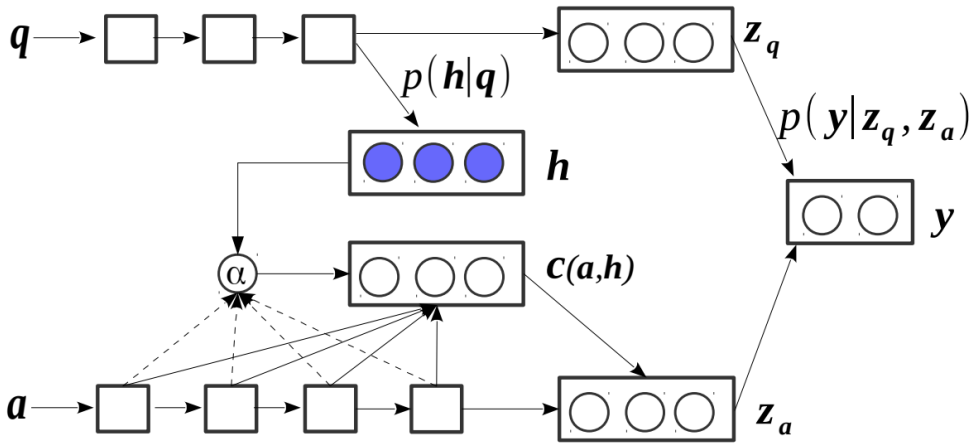


Figure 2. NASM for question answer selection.

NVDM = Neural Variational Document Model (unsupervised)
NASM = Neural Answer Selection Model (supervised)

Note

At first, I was looking for something like **Hierarhical VAEs** (e.g. normalizing flow), or **stochastic-in-time** (e.g. time-series) VAEs as paper 1 & 2. But this work is none of that. Furthermore, it can **only** generate documents as bags-of-words, no better than LDA.

Neural Variational Inference Framework (1)

We introduce the common framework for both **unsupervised** NVDM and **supervised** NASM. Let $\boldsymbol{h}$ denote the latent variable. Let $\boldsymbol{x}$ and $\boldsymbol{y}$ denote the observed parent of $\boldsymbol{h}$, and child nodes of $\boldsymbol{h}$, respectively.

$$p_\theta(\boldsymbol{x}, \boldsymbol{y}) = \sum_{\boldsymbol{h}} p_\theta(\boldsymbol{y}|\boldsymbol{h}) p_\theta(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{x})$$

$$\log p_\theta(\boldsymbol{x}, \boldsymbol{y}) = \log \int \frac{q(\boldsymbol{h})}{q(\boldsymbol{h})} p_\theta(\boldsymbol{y}|\boldsymbol{h}) p_\theta(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{x}) d\boldsymbol{h}$$

We can establish the lower bound:

$$\log p_\theta(\boldsymbol{x}, \boldsymbol{y}) \geq \mathbb{E}_{q(\boldsymbol{h})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{h}) p_\theta(\boldsymbol{h}|\boldsymbol{x}) p(\boldsymbol{x}) - \log q(\boldsymbol{h})]$$

Neural Variational Inference Framework (2)

Parameterized diagonal Gaussian $\mathcal{N}(\boldsymbol{h}|\boldsymbol{\mu}(\boldsymbol{x},\boldsymbol{y}), \text{diag}(\boldsymbol{\sigma}^2(\boldsymbol{x},\boldsymbol{y})))$ as $q_\phi(\boldsymbol{h}|\boldsymbol{x},\boldsymbol{y})$. The inference network is constructed as:

1. Construct vector representations of the observed variables: $\boldsymbol{u} = f_x(\boldsymbol{x})$, $\boldsymbol{v} = f_y(\boldsymbol{y})$. f_x and f_y are any type of neural networks.
2. Assemble a joint representation: $\boldsymbol{\pi} = g(\boldsymbol{u},\boldsymbol{v})$. g is a concatenation and MLP.
3. Parameterize the variational distribution over the latent variable: $\boldsymbol{\mu} = l_1(\boldsymbol{\pi})$, $\log \boldsymbol{\sigma} = l_2(\boldsymbol{\pi})$. l_1 and l_2 are linear transformations.
4. Reparameterization $\boldsymbol{h} = \boldsymbol{\mu} + \boldsymbol{\sigma} \cdot \boldsymbol{\epsilon}$, $\boldsymbol{\epsilon}^{(\ell)} \sim \mathcal{N}(\boldsymbol{0}, \boldsymbol{1})$.

Neural Variational Document Model

Let $\boldsymbol{X} \in \mathbb{R}^{|V|}$ be the bag-of-words representation of a document, and $\boldsymbol{x}_i \in \mathbb{R}^{|V|}$ be the one-hot representation of the i-th word. The lower bound:

$$\mathcal{L} = \mathbb{E}_{q_\phi(\boldsymbol{h}|\boldsymbol{X})}\left[\sum_{i=1}^{N} \log p_\theta(\boldsymbol{x}_i|\boldsymbol{h})\right] - \mathcal{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{h}|\boldsymbol{X})||p(\boldsymbol{h}))$$

where N is the number of words in the document. The decoder $p_\theta(\boldsymbol{x}_i|\boldsymbol{h})$ is modeled by multinomial logistic regression and shared across documents:

$$p_\theta(\boldsymbol{x}_i|\boldsymbol{h}) = \frac{\exp\{-E(\boldsymbol{x}_i; \boldsymbol{h}, \theta)\}}{\sum_{j=1}^{|V|} \exp\{-E(\boldsymbol{x}_j; \boldsymbol{h}, \theta)\}}$$

$$E(\boldsymbol{x}_i; \boldsymbol{h}, \theta) = -\boldsymbol{h}^T \boldsymbol{R}\boldsymbol{x}_i - \boldsymbol{b}_{x_i}$$

where $\boldsymbol{R} \in \mathbb{R}^{K \times |V|}$ learns the semantic word embeddings and $\boldsymbol{b}_{x_i}$ represents the bias term.

Neural Answer Selection Model (1)

- **Problem setting:** A question $\boldsymbol{q}$ is associated with a set of answer sentences $\{\boldsymbol{a}_1, .., \boldsymbol{a}_n\}$ together with their judgments $\{\boldsymbol{y}_1, .., \boldsymbol{y}_n\}$, where $\boldsymbol{y}_m = 1$ if the answer $\boldsymbol{a}_m$ is correct, and $\boldsymbol{y}_m = 0$ otherwise.
- **Task:** Predict $\boldsymbol{y}$ for the question-answer pair $(\boldsymbol{q}, \boldsymbol{a})$.
- **Model:** In short, we employ 2 LSTMs (one for the question sentence, and the another one for the answer sentence) along with the attention mechanism.
- Why don't we use a **deterministic** model instead? The authors claimed this variational approach is more robust to noises(!?).

Neural Answer Selection Model (2)

Variational lower bound:

$$\log p(\boldsymbol{y}|\boldsymbol{q},\boldsymbol{a}) = \log \int p_\theta(\boldsymbol{y}|z_q(\boldsymbol{q}), z_a(\boldsymbol{a},\boldsymbol{h}))p_\theta(\boldsymbol{h}|\boldsymbol{q})d\boldsymbol{h} \geq$$

$$\mathbb{E}_{q_\phi(\boldsymbol{h})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{z}_q(\boldsymbol{q}), \boldsymbol{z}_a(\boldsymbol{a},\boldsymbol{h}))] - \mathcal{D}_{\mathrm{KL}}(q_\phi(\boldsymbol{h})||p_\theta(\boldsymbol{h}|\boldsymbol{q}))$$

where $\boldsymbol{z}_q(\boldsymbol{q})$ and $\boldsymbol{z}_a(\boldsymbol{a},\boldsymbol{h})$ are the question and answer representations for predicting $\boldsymbol{y}$:

$$p_\theta(\boldsymbol{y}=1|\boldsymbol{z}_q,\boldsymbol{z}_a) = \sigma(\boldsymbol{z}_q^T\boldsymbol{M}\boldsymbol{z}_a + b)$$

Note: $\boldsymbol{z}_q(\boldsymbol{q})$ is **deterministic**, while $\boldsymbol{z}_a(\boldsymbol{a},\boldsymbol{h})$ is non-deterministic.

Paper 4

Learning to Generate with Memory (ICML 2016)

Chongxuan Li, Jun Zhu, Bo Zhang

`https://arxiv.org/pdf/1602.07416.pdf`

Proposals

Proposals

- A **deep generative model** with a possibly large **external memory** and an **attention mechanism** to capture the local detail information (that is often lost in the bottom-up abstraction process).
- By adopting a **smooth** attention model, the whole network is trained end-to-end.
- Applications:
  1. Density estimation
  2. Image generation
  3. Missing value imputation

Overall architecture

Overall architecture

- Combination between the **stochastic layers** and **deterministic layers**.
- Each deterministic layer is associated with an **external memory** to capture local variant information.
- An attention mechanism is used to record information in the memory during learning, and retrieve information from the memory during data generation.

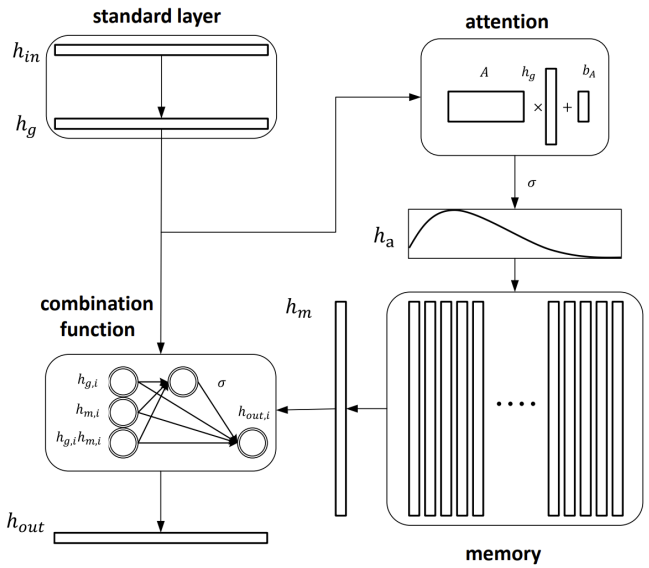


Figure 1. Architecture comparison between a standard layer (top-left part) and a layer with memory (the whole figure).

Probabilistic DGMs with Memory (1)

Note

This work is an extension to the **Hierarchial VAEs** to address its loss of detail information in the abstraction process, that we discussed in one of our meetings:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___September_11__2020.pdf`

Given a set of training data $\mathcal{D}$. The latent factors $\boldsymbol{z}_L, .., \boldsymbol{z}_1$ are organized in a hierarchy:

- Draw the top-layer $\boldsymbol{z}_L \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$.
- For $\ell = L-1, .., 0$, calculate the mean parameters $\boldsymbol{\mu}_\ell = g_\ell(\boldsymbol{z}_{\ell+1}; M_\ell)$ and draw the factors $\boldsymbol{z}_\ell \sim P_\ell(\boldsymbol{\mu}_\ell)$.
- $\boldsymbol{z}_0 = \boldsymbol{x} \leftarrow$ This is data.

Probabilistic DGMs with Memory (2)

$$\boldsymbol{\mu}_\ell = g_\ell(\boldsymbol{z}_{\ell+1}; M_\ell) \qquad \boldsymbol{z}_\ell \sim P_\ell(\boldsymbol{\mu}_\ell)$$

- P_ℓ is assumed to be of an exponential family form, with mean parameters $\boldsymbol{\mu}_\ell$.
- g_ℓ is a feed-forward deep neural network with l_ℓ deterministic layers and a set of associated memories $\{M_\ell^{(i)}\}_{i=0}^{l_\ell-1}$, one per layer.
- Each memory is a trainable matrix with dimension $d_s \times n_s$, where d_s is the number of slots in the memory, and n_s is the dimension of each slot.
- The network g_ℓ is parameterized as:
  - $\boldsymbol{h}_\ell^{l_\ell} = \boldsymbol{z}_{\ell+1}$ $\leftarrow$ The top layer of g_ℓ.
  - $\boldsymbol{h}_\ell^{(i)} = \phi(\boldsymbol{h}_\ell^{(i+1)}; M_\ell^{(i)})$ $\leftarrow$ The i-th layer of g_ℓ.

Inference (1)

Let $\boldsymbol{\theta}_g$ be the collection of parameters in the DGM.

$$p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}_g) = p(\boldsymbol{z}; \boldsymbol{\theta}_g) p(\boldsymbol{x}|\boldsymbol{z}; \boldsymbol{\theta}_g)$$

- Q-**net (encoder)**: To model the approximate posterior distribution $q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}_r)$, where $\boldsymbol{\theta}_r$ is the collection of parameters in the recognition model. No external memory.

$$\hat{\boldsymbol{h}}^{(i+1)} = \phi(V^{(i)} \hat{\boldsymbol{h}}^{(i)} + b_r^{(i)})$$

- P-**net (decoder)**: With external memories. The overall architecture is asymmetric.

Inference (2)

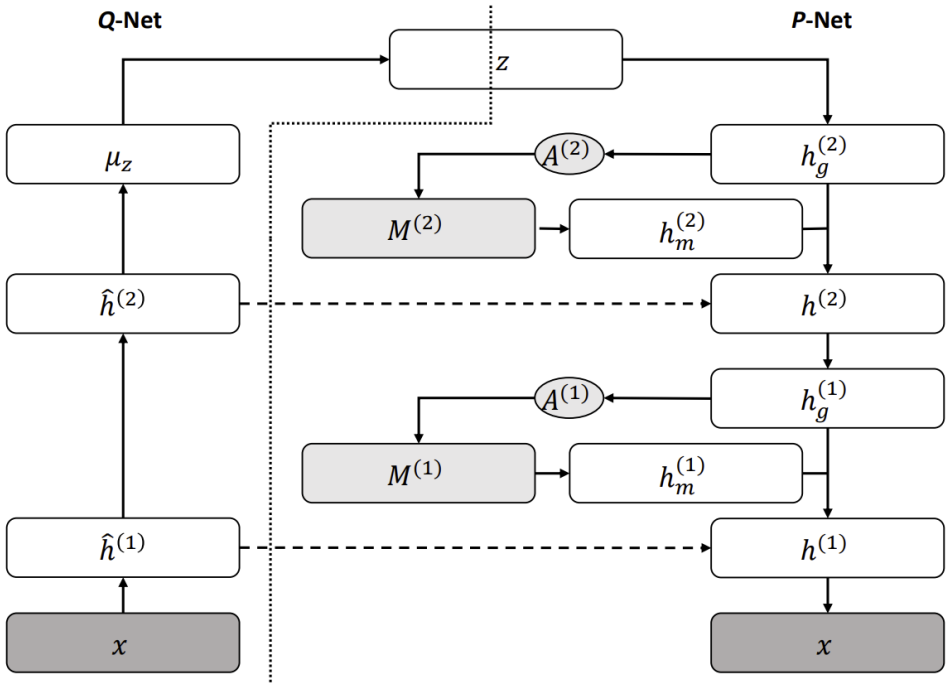


Figure 2. A model with one stochastic layer and two deterministic layers, where $\mathbf{z}$ is shared by the P-net and Q-net.

Note: The authors only used 1 stochastic layer, while the Hierarchical VAEs employs **several** stochastic layers. So they cannot claim it addresses the loss of information in Hierarchical VAEs.

Inference (3)

A variational lower bound of log-likelihood for per data $\boldsymbol{x}$ can be formulated as:

$$\mathcal{L}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_r; \boldsymbol{x}) \triangleq \mathbb{E}_{q(\boldsymbol{z}|\boldsymbol{x};\boldsymbol{\theta}_r)}[\log p(\boldsymbol{x}, \boldsymbol{z}; \boldsymbol{\theta}_g) - \log q(\boldsymbol{z}|\boldsymbol{x}; \boldsymbol{\theta}_r)]$$

We add local reconstruction error terms as an optional regularizer, and jointly optimize the parameters in the generative model and the recognition model:

$$\min_{\boldsymbol{\theta}_g, \boldsymbol{\theta}_r} \frac{1}{|\mathcal{D}|} \sum_{\boldsymbol{x} \in \mathcal{D}} \left\{ \mathcal{L}(\boldsymbol{\theta}_g, \boldsymbol{\theta}_r; \boldsymbol{x}) + \sum_{i=1}^{I} \lambda^{(i)} ||\boldsymbol{h}^{(i)} - \hat{\boldsymbol{h}}^{(i)}||_2^2 \right\}$$

where $\lambda^{(i)}$ are the prefixed hyperparameters.

How does this relate to our research?

Some idea we can use for **multiscale** graph generation:

1. The recognition (encoder) model can apply **differential pooling** to contruct a coarse-grained, bottom-up abstraction:
   - $\hat{\boldsymbol{h}}^{(1)}$ is a complete graph pooled from input graph $\boldsymbol{x}$: $\hat{\boldsymbol{h}}^{(1)} = \text{POOL}(\boldsymbol{x})$
   - $\hat{\boldsymbol{h}}^{(i+1)} = \text{POOL}(\hat{\boldsymbol{h}}^{(i)})$
2. The generative (decoder) model can reconstruct the graph in a reverse coarse-grained manner, from the high/general level into the low/detail level, along with a reconstruction loss:

$$\sum_{i=1}^{I} \lambda^{(i)} ||\boldsymbol{h}^{(i)} - \hat{\boldsymbol{h}}^{(i)}||_2^2$$

Paper 5

Generated Loss, Augmented Training, and Multiscale VAE

(only a technical report with VAE training tricks)

Jason Chou, Gautam Hathi

`https://arxiv.org/pdf/1904.10446.pdf`

Proposals

Proposals

- **Discrete** VAE with tree recursive architecture shows limited capability of capturing field correlations within structured data.
- **Augmented training**: augmenting training data with generated variants.
- **Multiscale VAE**: training a VAE with multiple values of β simultaneously.

Some insights

- **Mode collapse:** VAE is directly trained to encode all training examples and therefore is less prone to the failure mode of generating a **few memorized training examples** as in GAN.
- **Sajjadi et al., 2018:** VAE tends to have lower precision (e.g. blurry images for visual problems). **Rezende & Viola, 2018:** Holes in the latent space. (This comes from the information bottleneck of VAE.)
- **Higgins et al., 2017:** Fine-tuning the strength of the information bottleneck by the introduction of the KL-divergence weight β.
- **Manual annealing:**
  1. KL-divergence warm-up
  2. Controlled capacity increase

Generalized β-VAE

The evidence lower bound (ELBO):

$$\log p(x) \geq \mathbb{E}[\log p_\theta(x|z)] - \mathcal{D}_{\mathrm{KL}}(q_\lambda(z|x)||p(z))$$

Modified ELBO:

$$\mathbb{E}[\log p_\theta(x|z)] - \beta\mathcal{D}_{\mathrm{KL}}(q_\lambda(z|x)||p(z))$$

where β controls the strength of the information bottleneck on the latent vector. For higher values of β, we accept lossier reconstruction.

Higgins et al., 2017

This hyperparameter β is (theoretically justified) a Karush-Kuhn-Tucker (KKT) multiplier for maximizing $\mathbb{E}[\log p_\theta(x|z)]$ under the inequality constraint $\mathcal{D}_{\mathrm{KL}}(q_\lambda(z|x)||p(z)) \leq c$ where c is a constant.

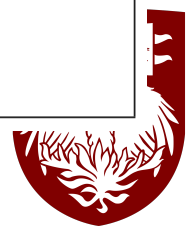
Augmented training (1)

Intuition

Augmented training extends the standard VAE training scheme. Instead of just taking one hop from the sampled latent vector and then minimizing the reconstruction loss, we take the second hop by taking the reconstructed sample as the new input to the encoder, and continue, and so on.

After `gen_start_step` steps:

1. Initialize $n_{\text{augmented}}$ augmented latent vectors with sampled latent vectors of the current training batch.
2. Augment next training batch with variants generated from the augmented latent vectors.
3. After a training step, each augmented latent vector is replaced with either:
   (a) The sampled latent vector of an example from the current training batch, selected without replacement, with probability p_{sampled}.
   (b) The sampled latent vector of the variant generated from it with probability $(1 - p_{\text{sampled}})$.
4. Repeat from 2.



Augmented training (2)

Mathematically saying:

$$x^{(1)} = g(z), \quad z \sim \mathcal{N}(\mu_\lambda(x), \sigma_\lambda(x))$$

$$x^{(2)} = g(z^{(1)}), \quad z^{(1)} \sim \mathcal{N}(\mu_\lambda(x^{(1)}), \sigma_\lambda(x^{(1)}))$$

$$\dots$$

$$x^{(n)} = g(z^{(n-1)}), \quad z^{(n-1)} \sim \mathcal{N}(\mu_\lambda(x^{(n-1)}), \sigma_\lambda(x^{(n-1)}))$$

The new objective:

$$\sum_{n=0}^{\infty} p_{\text{sampled}}^{\min(n,1)} (1 - p_{\text{sampled}})^{\max(n-1,0)} \Big\{ \mathbb{E}[\log p_\theta(x^{(n)}|z^{(n)})] - \beta \mathcal{D}_{\text{KL}}(q_\lambda(z^{(n)}|x^{(n)})||p(z^{(n)})) \Big\}$$

Multiscale VAE (1)

Observations:

- We can improve the model with KL-divergence warm-up (high β) and cool-down (low β).
- The objective functions with different β are **not** necessarily in conflict with each other.
- Training with β_{max}, the highest possible β before $q_\lambda(z|x)$ collapes into multivariate unit Gaussian, optimizes the global structure of the latent space.
- Training with smaller β optimizes the local structure.
- Training with $\beta = 0$ optimizes autoencoding.

Multiscale VAE (2)

New training scheme with multiple values of β, each trained on one worker:

1. Initialize $n_{\text{KL\_weight}}$ standard deviation networks $\sigma_{\lambda,i}(\mu)$, where each standard deviation network is associated with a distinct but constant β value β_i. Without loss of generality, we assume $\beta_i < \beta_{i+1}$.
2. Assign $(\beta_i, \sigma_{\lambda,i}(\mu))$ pairs evenly to workers, which otherwise share the same encoder / decoder.
3. Train the model with these workers.

In terms of objective function, we have

$$\sum_{i=0}^{\text{kl\_weight\_levels}-1} \mathbb{E}[\log p_\theta(x|z^{(i)})] - \beta_i KL(q_{\lambda,i}(z^{(i)}|x)||p(z^{(i)})) \tag{5}$$

$$\text{where } z^{(i)} \sim \mathcal{N}(\mu_\lambda(x), \sigma_{\lambda,i}(\mu_\lambda(x)))$$

Note: This kind of **asynchronous** training is actually used in the industry to boost up the efficiency.

Paper 6

Learning Correlated Latent Representations with Adaptive Priors

(NeurIPS 2019 workshop)

Da Tang, Dawen Liang, Nicholas Ruozzi, Tony Jebara

http://bayesiandeeplearning.org/2019/papers/47.pdf

Proposals

Proposals

- Based on CVAEs, the authors proposed Adaptive Correlated Variational Auto-Encoders (ACVAEs): an **adaptive prior** distribution that can be **adjusted during training**.
- Applications: link prediction and hierarchical clustering.

Correlated Variational Auto-Encoders (CVAEs)

For a presentation about the previous work (Tony Jebara's group) of this one, please check our previous discussion in:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___August_07__2020.pdf`

Standard VAEs (1)

Input data $\boldsymbol{x} = \{\boldsymbol{x}_1, .., \boldsymbol{x}_n\} \subseteq \mathrm{R}^D$. Standard VAEs assume that each data point $\boldsymbol{x}_i$ is generated independently by the following process:

1. Genearte the latent variables $\boldsymbol{z} = \{\boldsymbol{z}_1, .., \boldsymbol{z}_n\} \subseteq \mathrm{R}^d$ $(d << D)$ by drawing i.i.d from the prior distribution (e.g. standard Gaussian distribution) $\boldsymbol{z}_i \sim p_0(\boldsymbol{z}_i)$, for each i.
2. Generate the data points $\boldsymbol{x}_i \sim p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}_i)$ from the model conditional distribution $p_{\boldsymbol{\theta}}$ independently.

Standard VAEs (2)

Optimizing θ to maximize the likelihood p_θ requires computing **intractable** posterior distribution

$$p_\theta(z|x) = \prod_{i=1}^{n} p_\theta(z_i|x_i)$$

VAEs approximates this posterior distribution as $q_\lambda(z|x) = \prod_{i=1}^{n} q_\lambda(z_i|x_i)$ via amortized inference and maximize the evidence lower bound (ELBO):

$$L(\lambda, \theta) = \mathrm{E}_{q_\lambda(z|x)}[\log p_\theta(x|z)] - \mathcal{D}_{\mathrm{KL}}(q_\lambda(z|x)||p_0(z))$$

$$= \sum_{i=1}^{n} \left[\mathrm{E}_{q_\lambda(z_i|x_i)}[\log p_\theta(x_i|z_i)] - \mathcal{D}_{\mathrm{KL}}(q_\lambda(z_i|x_i)||p_0(z_i)) \right]$$

Correlated prior (1)

Given an undirected correlation graph $G = (V, E)$, where $(v_i, v_j) \in E$ represents that the data point x_i and x_j are correlated:

$$p_0^{corr}(z_i) = p_0(z_i) \qquad \forall v_i \in V$$

$$p_0^{corr}(z_i, z_j) = p_0(z_i, z_j) \qquad \forall (v_i, v_j) \in E$$

Wainwright & Jordan, 2008

For acyclic graph, the prior is factorized as singletons and pairwise marginal distributions:

$$p_0^{corr}(z) = \prod_{i=1}^{n} p_0(z_i) \prod_{(v_i, v_j) \in E} \frac{p_0(z_i, z_j)}{p_0(z_i) p_0(z_j)}$$

Our previous discussion about **[Wainwright & Jordan, 2008]**:

`http://people.cs.uchicago.edu/~hytruongson/Discussions-2020/Group_meeting___August_14__2020.pdf`

Correlated prior (2)

Maximal acyclic subgraph

For an undirected graph $G = (V, E)$, a subgraph $G' = (V', E')$ is a maximal acyclic subgraph of G if:

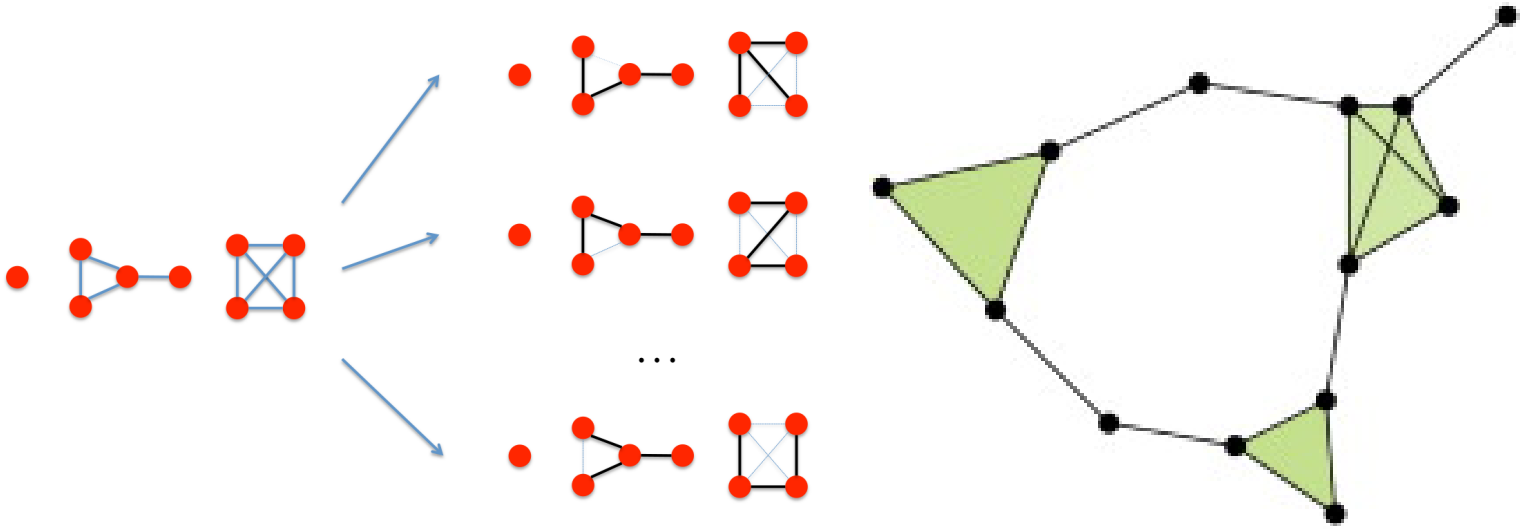
1. G' is acylic.
2. $V' = V$, i.e., G' contains all vertices of G.
3. Adding any edge from E/E' to E' will create a cycle in G'.

CVAEs optimize a different ELBO:

$$\log p_\theta(\boldsymbol{x}) = \mathrm{E}_{p_0^{corrg}(\boldsymbol{z})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] = \frac{1}{|\mathcal{A}_G|} \sum_{G' \in \mathcal{A}_G} \mathrm{E}_{p_0^{G'}(\boldsymbol{z})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})]$$

$$\geq \frac{1}{|\mathcal{A}_G|} \sum_{G' \in \mathcal{A}_G} \left(\mathrm{E}_{q_\lambda^{G'}(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{x}|\boldsymbol{z})] - \mathcal{D}(q_\lambda^{G'}(\boldsymbol{z}|\boldsymbol{x}) || p_0^{G'}(\boldsymbol{z})) \right)$$

A different type of decomposition: maximal clique



The left is maximal acyclic graph (tree) decomposition.
The right is maximal clique decomposition.

Note: The right picture is taken from Wikipedia.

Non-Uniform Mixture Prior (1)

Motivated by the ELBO:

$$\frac{1}{|\mathcal{A}_G|}\sum_{G'\in\mathcal{A}_G}\left(\mathbb{E}_{q_\lambda^{G'}(z|x)}[\log p_\theta(x|z)] - \mathcal{D}(q_\lambda^{G'}(z|x)||p_0^{G'}(z))\right)$$

Rather than using a uniform average, we employ a categorical distribution $\pi \in \triangle^{|\mathcal{A}_G|-1}$ representing the normalized weights over all maximal acyclic subgraphs $G' \in \mathcal{A}_G$ of G. With the non-uniform distribution π, we get a new ELBO:

$$\log p_{\pi,\theta}(x) \geq \mathbb{E}_{p_0^\pi(z)}[\log p_\theta(x|z)] = \mathbb{E}_{G'\sim\pi}\left[\mathbb{E}_{p_0^\pi(z)}[\log p_\theta(x|z)]\right] \geq$$

$$\geq \mathbb{E}_{G'\sim\pi}\left[\mathbb{E}_{q_\lambda^{G'}(z|x)}[\log p_\theta(x|z)] - \mathcal{D}(q_\lambda^{G'}(z|x)||p_0^{G'}(z))\right]$$

with the non-uniform prior $p_0^\pi(z) = \mathbb{E}_{G'\sim\pi}[p_0^{G'}(z)]$.

Non-Uniform Mixture Prior (2)

If we optimize $\boldsymbol{\pi}$ together with all other parameters, we are optimizing with an adaptive prior:

$$\mathcal{L}^{\text{ACVAE}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta}) := \sum_{i=1} \Big(\mathbb{E}_{q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i|\boldsymbol{x}_i)} \left[\log p_{\boldsymbol{\theta}}(\boldsymbol{x}_i|\boldsymbol{z}_i) \right] - \text{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i|\boldsymbol{x}_i)||p_0(\boldsymbol{z}_i)) \Big) - \sum_{(v_i,v_j)\in E} w^{\text{MAS}}_{G,\boldsymbol{\pi},(v_i,v_j)} \cdot$$
$$\Big(\text{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i, \boldsymbol{z}_j|\boldsymbol{x}_i, \boldsymbol{x}_j)||p_0(\boldsymbol{z}_i, \boldsymbol{z}_j)) - \text{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i|\boldsymbol{x}_i)||p_0(\boldsymbol{z}_i)) - \text{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}_j|\boldsymbol{x}_j)||p_0(\boldsymbol{z}_j)) \Big).$$

where:

$$w^{\text{MAS}}_{G,\boldsymbol{\pi},e} = \sum_{G'\in\mathcal{A}_G, e\in E'} \boldsymbol{\pi}(G')$$

ELBO with regularization:

$$\mathcal{L}^{\text{ACVAE-NS}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta}) := \mathcal{L}^{\text{ACVAE}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta}) - \gamma \cdot \Big(\sum_{i=1}^{n} \text{KL}(q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i|\boldsymbol{x}_i)||p_0(\boldsymbol{z}_i))$$
$$+ \frac{2}{n} \sum_{1\le i<j\le n} \mathbb{E}_{q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i,\boldsymbol{z}_j|\boldsymbol{x}_i,\boldsymbol{x}_j)} \log \frac{q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i, \boldsymbol{z}_j|\boldsymbol{x}_i, \boldsymbol{x}_j)}{q_{\boldsymbol{\lambda}}(\boldsymbol{z}_i|\boldsymbol{x}_i) q_{\boldsymbol{\lambda}}(\boldsymbol{z}_j|\boldsymbol{x}_j)} \Big).$$

Learning the Non-Uniform Mixture

Directly maximize $\mathcal{L}^{\text{ACVAE}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta})$ with respect to $\boldsymbol{\pi}$, $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$:

$$\max_{\boldsymbol{\lambda},\boldsymbol{\theta}} \max_{\boldsymbol{\pi}} \mathcal{L}^{\text{ACVAE}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta})$$

Alternatively, we can consider a minimax saddle-point optimization (which may lead to more robust inference):

$$\max_{\boldsymbol{\lambda},\boldsymbol{\theta}} \min_{\boldsymbol{\pi}} \mathcal{L}^{\text{ACVAE}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta})$$

Proposition 1

If the optimization has global optima, then at least one optimum $(\boldsymbol{\pi}^*, \boldsymbol{\lambda}^*, \boldsymbol{\theta}^*)$ will have a $\boldsymbol{\pi}^*$ that places all of its probability mass on a single maximal acyclic subgraph $G'^* \in \mathcal{A}_G$.

That means at this optimum, the ELBO becomes the ELBO on a single spanning forest G'^*.

ACVAEs Learning

Algorithm 1 ACVAEs learning

Input: data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n \in \mathbb{R}^D$, undirected graph $G = (V = \{v_1, \ldots, v_n\}, E)$, parameter $\gamma > 0$.
Initialize the parameters $\boldsymbol{\lambda}$, $\boldsymbol{\theta}$. Initialize the weights $w^{\text{MAS}}_{G,\boldsymbol{\pi},e}$ for each $e \in E$.
while not converged **do**
  Optimize the parameters $(\boldsymbol{\lambda}, \boldsymbol{\theta})$ using the gradients $\nabla_{\boldsymbol{\lambda},\boldsymbol{\theta}} \mathcal{L}^{\text{ACVAE}}(\boldsymbol{\pi}, \boldsymbol{\lambda}, \boldsymbol{\theta})$.
  Compute the mass $m_{(v_i,v_j)}$ for each edge $e \in E$ with Eq. 11.
  Compute a minimum (if applying the empirical Bayes option in Eq. 9) or maximum (if applying the saddle-point option in Eq. 10) spanning tree of the graph with the masses $m_{(v_i,v_j)}$'s as the edge weights and update the weights $w^{\text{MAS}}_{G,\boldsymbol{\pi},e}$ for each $e \in E$ according to Eq. 12.
end while
Return: The parameters $\boldsymbol{\lambda}$, $\boldsymbol{\theta}$, the weights $w^{\text{MAS}}_{G,\boldsymbol{\pi},e}$ for each $e \in E$.

The algorithm can be understood as:

while not converged **do**

1. Update for $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$
2. Update for $\boldsymbol{\pi}$

end while

Metrics

Given learned latent embeddings $\boldsymbol{z}_1, .., \boldsymbol{z}_n$. We compute the pairwise distance $d_{i,j}$ between each pair $(\boldsymbol{z}_i, \boldsymbol{z}_j)$ as $||\boldsymbol{z}_i - \boldsymbol{z}_j||_2^2$. For the link prediction experiments, for each node v_i, we define the **Cumulative Reciprocal Rank** (CRR) as follows:

$$\text{CRR}_i = \sum_{(v_i, v_j) \in E_{\text{test}}} \frac{1}{|\{k : (v_i, v_k) \notin E_{\text{train}}, d_{i,k} \leq d_{i,j}\}|}$$

We further normalize CRR values to be in $[0, 1]$ as normalized CRR (NCRR). The larger CRR, the better.