Model learning with structured latent representations

Sharad Vikram

July 7, 2017

UCSD

Background

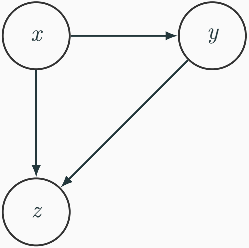
Probabilistic graphical models

A probabilistic graphical model (PGM) is a graphical representation of a joint distribution over several random variables.

Probabilistic graphical models

A probabilistic graphical model (PGM) is a graphical representation of a joint distribution over several random variables.

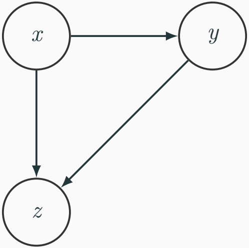
For example, a possible graph for variables x, y and z is



Probabilistic graphical models

A probabilistic graphical model (PGM) is a graphical representation of a joint distribution over several random variables.

For example, a possible graph for variables x, y and z is



The joint distribution factorizes as

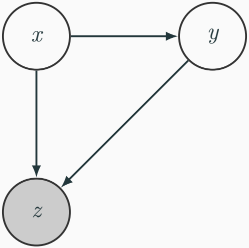
$$p(x, y, z) = p(x)p(y|x)p(z|x, y)$$

Bayesian inference

Bayesian inference allows us to compute probability distributions after observations have been made.

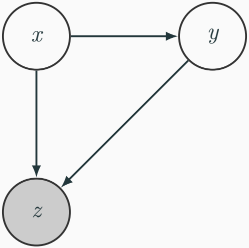
Bayesian inference

Bayesian inference allows us to compute probability distributions after observations have been made.



Bayesian inference

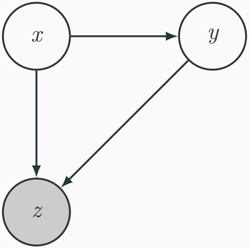
Bayesian inference allows us to compute probability distributions after observations have been made.



We are interested in the posterior distribution $p(x, y|z)$

Bayesian inference

Bayesian inference allows us to compute probability distributions after observations have been made.



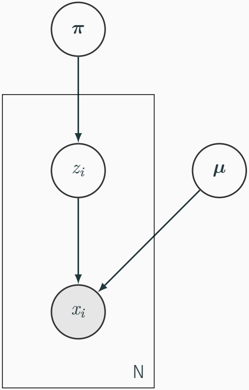
We are interested in the posterior distribution $p(x, y|z)$

This can be computed via Bayes rule:

$$p(x, y|z) = \frac{p(x, y, z)}{p(z)} = \frac{p(x)p(y|x)p(z|x, y)}{\int p(x)p(y|x)p(z|x, y)\,dxdy}$$

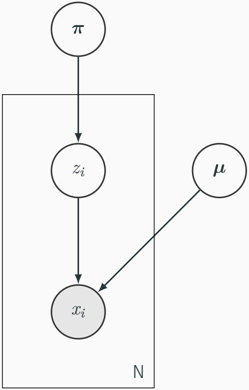
Example PGM

Latent variable model: Gaussian mixture model



Example PGM

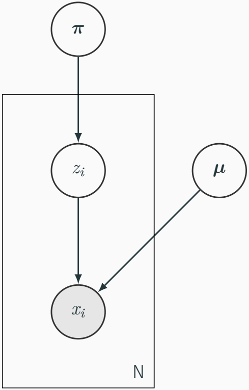
Latent variable model: Gaussian mixture model



Local variables: $\{z_i\}_{i=1}^{N}$

Example PGM

Latent variable model: Gaussian mixture model



Local variables: $\{z_i\}_{i=1}^N$

Global variables: μ, π

Conjugacy

Conjugate

Two random variables x and y whose distribution is

$$p(x, y) = p(x)p(y|x)$$

are said to be *conjugate* if the posterior $p(x|y)$ is in the same family of distributions as $p(x)$.

Conjugacy

Conjugate

Two random variables x and y whose distribution is

$$p(x, y) = p(x)p(y|x)$$

are said to be *conjugate* if the posterior $p(x|y)$ is in the same family of distributions as $p(x)$.

Examples of conjugate distributions:

- Normal/normal

Conjugacy

Conjugate

Two random variables x and y whose distribution is

$$p(x, y) = p(x)p(y|x)$$

are said to be *conjugate* if the posterior $p(x|y)$ is in the same family of distributions as $p(x)$.

Examples of conjugate distributions:

- Normal/normal
- Normal-inverse-wishart(NIW)/normal

Conjugacy

Conjugate

Two random variables x and y whose distribution is

$$p(x, y) = p(x)p(y|x)$$

are said to be *conjugate* if the posterior $p(x|y)$ is in the same family of distributions as $p(x)$.

Examples of conjugate distributions:

- Normal/normal
- Normal-inverse-wishart(NIW)/normal
- Dirichlet/multinomial

Algorithms for inference

Depending on the structure of the graphical model and conjugacy of the variables, computing the posterior distribution can be easy, or very hard!

Algorithms for inference

Depending on the structure of the graphical model and conjugacy of the variables, computing the posterior distribution can be easy, or very hard!

Possible ways:

- Compute posterior analytically

Algorithms for inference

Depending on the structure of the graphical model and conjugacy of the variables, computing the posterior distribution can be easy, or very hard!

Possible ways:

- Compute posterior analytically
- Sampling (MCMC, Gibbs sampling)

Algorithms for inference

Depending on the structure of the graphical model and conjugacy of the variables, computing the posterior distribution can be easy, or very hard!

Possible ways:

- Compute posterior analytically
- Sampling (MCMC, Gibbs sampling)
- Expectation-maximization

Algorithms for inference

Depending on the structure of the graphical model and conjugacy of the variables, computing the posterior distribution can be easy, or very hard!

Possible ways:

- Compute posterior analytically
- Sampling (MCMC, Gibbs sampling)
- Expectation-maximization
- Variational inference

Algorithms for inference

Depending on the structure of the graphical model and conjugacy of the variables, computing the posterior distribution can be easy, or very hard!

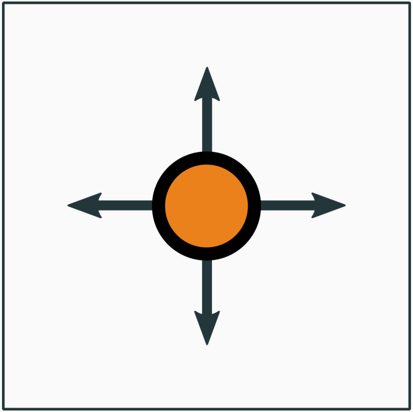
Possible ways:

- Compute posterior analytically
- Sampling (MCMC, Gibbs sampling)
- Expectation-maximization
- Variational inference

PGMs offer interpretability and quantifiable uncertainty, but often don't scale well with data and can be underexpressive.

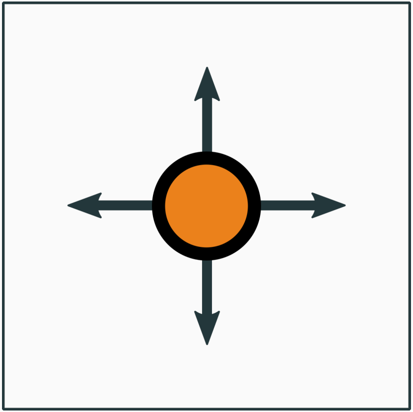
Model learning

Consider an agent in a system, and a set of its possible actions.



Model learning

Consider an agent in a system, and a set of its possible actions.



Model learning is the problem of estimating the dynamics of the system when we don't know it beforehand.

Model learning

Formally, consider an agent in a system with state space $\mathcal{S}$ with action space $\mathcal{A}$ with underlying dynamics function $p(s_{t+1}|s_t, a_t)$.

We are interested in learning an approximate dynamics function

$$\hat{p}(s_{t+1}|s_t, a_t)$$

from a dataset of trajectories

$$\boldsymbol{\tau} = \{(s_0^{(i)}, a_0^{(i)}, s_1^{(i)}, a_1^{(i)}, \dots, s_T^{(i)})\}_{i=1}^N$$

Bayesian linear dynamical system

A simple assumption: **Bayesian linear dynamical system** (LDS)

$$\mu_\rho, \Sigma_\rho \sim \mathcal{NIW}(\Psi, \nu, \mu_0, \kappa)\,, \quad \mathbf{F}, \Sigma \sim \mathcal{MNIW}(\Psi, \nu, \boldsymbol{M}_0, \boldsymbol{V})\,,$$

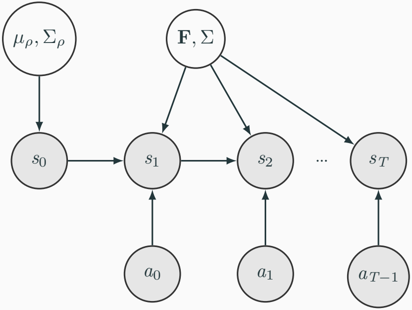
$$\mathbf{s}_0 \mid \mu_\rho, \Sigma_\rho \sim \mathcal{N}(\mu_\rho, \Sigma_\rho)\,, \qquad \mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t \sim \mathcal{N}\left(\mathbf{F}\begin{bmatrix}\mathbf{s}_t \\ \mathbf{a}_t\end{bmatrix}, \Sigma\right) \text{ for } t \in [0, \ldots, T]$$

Bayesian linear dynamical system

A simple assumption: **Bayesian linear dynamical system** (LDS)

$$\mu_\rho, \Sigma_\rho \sim \mathcal{NIW}(\Psi, \nu, \mu_0, \kappa)\,, \quad \mathbf{F}, \Sigma \sim \mathcal{MNIW}(\Psi, \nu, \boldsymbol{M}_0, \boldsymbol{V})\,,$$

$$\mathbf{s}_0 \mid \mu_\rho, \Sigma_\rho \sim \mathcal{N}(\mu_\rho, \Sigma_\rho)\,, \qquad \mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t \sim \mathcal{N}\left(\mathbf{F}\begin{bmatrix}\mathbf{s}_t \\ \mathbf{a}_t\end{bmatrix}, \Sigma\right) \text{ for } t \in [0, \ldots, T]$$

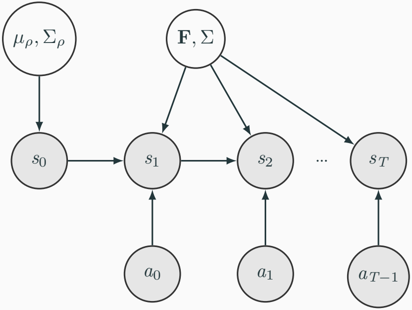


Bayesian linear dynamical system

A simple assumption: **Bayesian linear dynamical system** (LDS)

$$\mu_\rho, \Sigma_\rho \sim \mathcal{NIW}(\Psi, \nu, \mu_0, \kappa)\,, \quad \mathbf{F}, \Sigma \sim \mathcal{MNIW}(\Psi, \nu, \boldsymbol{M}_0, \boldsymbol{V})\,,$$

$$\mathbf{s}_0 \mid \mu_\rho, \Sigma_\rho \sim \mathcal{N}(\mu_\rho, \Sigma_\rho)\,, \qquad \mathbf{s}_{t+1} \mid \mathbf{s}_t, \mathbf{a}_t \sim \mathcal{N}\left(\mathbf{F}\begin{bmatrix}\mathbf{s}_t \\ \mathbf{a}_t\end{bmatrix}, \Sigma\right) \text{ for } t \in [0, \ldots, T]$$



We are interested in the posterior distribution $p(\mu_\rho, \Sigma_\rho, \mathbf{F}, \Sigma | s_0, a_0, \ldots, s_T)$ which can be computed analytically.

Variational inference at a high level

Variational inference at a high level

Consider a latent variable model with global variables θ, local variables z and observations x. Our desired posterior is $p(\theta, z|x)$

Variational inference at a high level

Consider a latent variable model with global variables θ, local variables z and observations x. Our desired posterior is $p(\theta, z|x)$

Strategy: convert inference into optimization

Variational inference at a high level

Consider a latent variable model with global variables θ, local variables z and observations x. Our desired posterior is $p(\theta, z|x)$

Strategy: convert inference into optimization

- Instantiate *variational distribution* $q_\phi(\theta, z)$ where ϕ are free parameters

Variational inference at a high level

Consider a latent variable model with global variables θ, local variables z and observations x. Our desired posterior is $p(\theta, z|x)$

Strategy: convert inference into optimization

- Instantiate *variational distribution* $q_\phi(\theta, z)$ where ϕ are free parameters
- Define loss $\mathrm{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$

Variational inference at a high level

Consider a latent variable model with global variables θ, local variables z and observations x. Our desired posterior is $p(\theta, z|x)$

Strategy: convert inference into optimization

- Instantiate *variational distribution* $q_\phi(\theta, z)$ where ϕ are free parameters
- Define loss $\mathrm{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$
- Minimize loss $\phi^* = \operatorname{argmin}_\phi \mathrm{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$

Variational inference at a high level

Consider a latent variable model with global variables θ, local variables z and observations x. Our desired posterior is $p(\theta, z|x)$

Strategy: convert inference into optimization

- Instantiate *variational distribution* $q_\phi(\theta, z)$ where ϕ are free parameters
- Define loss $\mathrm{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$
- Minimize loss $\phi^* = \operatorname{argmin}_\phi \mathrm{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$

If $q(\theta, z)$ is sufficiently expressive, it can approximate $p(\theta, z|x)$ quite well.

KL-divergence

Kullback-Leibler (KL) divergence is a measure of how far one probability distribution is from another.

KL-divergence

Kullback-Leibler (KL) divergence is a measure of how far one probability distribution is from another.

For distributions $q(x)$ and $p(x)$,

$$\mathrm{KL}(q(x) \| p(x)) = \int q(x) \log \frac{q(x)}{p(x)} \, dx$$

KL-divergence

Kullback-Leibler (KL) divergence is a measure of how far one probability distribution is from another.

For distributions $q(x)$ and $p(x)$,

$$\mathrm{KL}(q(x) \| p(x)) = \int q(x) \log \frac{q(x)}{p(x)} \, dx$$

Properties:

- $\mathrm{KL}(q(x) \| p(x)) = 0$ if $q(x) = p(x)$.
- Asymmetric

Evidence lower bound

In general, we cannot even compute $KL(q(\theta, z) \| p(\theta, z|x))$ because we don't know the posterior $p(\theta, z|x)$.

Evidence lower bound

In general, we cannot even compute $KL(q(\theta, z) \| p(\theta, z|x))$ because we don't know the posterior $p(\theta, z|x)$.

We can rewrite the KL divergence as

$$\begin{aligned} \mathrm{KL}(q(\theta, z) \| p(\theta, z|x)) &= \int q(\theta, z) \log \frac{q(\theta, z)}{p(\theta, z|x)} \, d\theta, z \\ &= \log p(x) - \mathbb{E}_{q(\theta, z)} \left[\log \frac{p(x, \theta, z)}{q(\theta, z)} \right] \end{aligned}$$

Evidence lower bound

In general, we cannot even compute $KL(q(\theta, z) \| p(\theta, z|x))$ because we don't know the posterior $p(\theta, z|x)$.

We can rewrite the KL divergence as

$$\begin{aligned}
\mathrm{KL}(q(\theta, z) \| p(\theta, z|x)) &= \int q(\theta, z) \log \frac{q(\theta, z)}{p(\theta, z|x)} \, d\theta, z \\
&= \log p(x) - \mathbb{E}_{q(\theta, z)} \left[\log \frac{p(x, \theta, z)}{q(\theta, z)} \right]
\end{aligned}$$

and maximize the *evidence lower bound* (ELBO)

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

Variational inference for PGMs

For a general graphical model with variable set $\mathbf{X} = \{x_1, x_2, \ldots\}$ we have joint distribution

$$p(\mathbf{X}) = \prod_i p(x_i|\mathrm{pa}_i)$$

where pa_i are the parents of node x_i in the graph.

Variational inference for PGMs

For a general graphical model with variable set $\mathbf{X} = \{x_1, x_2, \ldots\}$ we have joint distribution

$$p(\mathbf{X}) = \prod_i p(x_i | \mathrm{pa}_i)$$

where pa_i are the parents of node x_i in the graph.

Let the set $\mathbf{H}$ be all unobserved variables and $\mathbf{V}$ be the observed.

Variational inference for PGMs

For a general graphical model with variable set $\mathbf{X} = \{x_1, x_2, \ldots\}$ we have joint distribution

$$p(\mathbf{X}) = \prod_i p(x_i|\mathrm{pa}_i)$$

where pa_i are the parents of node x_i in the graph.

Let the set $\mathbf{H}$ be all unobserved variables and $\mathbf{V}$ be the observed.

We are interested in the posterior $p(\mathbf{H}|\mathbf{V})$ and use variational distribution

$$q(\mathbf{H}) = \prod_i q(\mathbf{H}_i)$$

Variational inference for PGMs

For a general graphical model with variable set $\mathbf{X} = \{x_1, x_2, \ldots\}$ we have joint distribution

$$p(\mathbf{X}) = \prod_i p(x_i|\mathrm{pa}_i)$$

where pa_i are the parents of node x_i in the graph.

Let the set $\mathbf{H}$ be all unobserved variables and $\mathbf{V}$ be the observed.

We are interested in the posterior $p(\mathbf{H}|\mathbf{V})$ and use variational distribution

$$q(\mathbf{H}) = \prod_i q(\mathbf{H}_i)$$

This is called the *mean-field* assumption.

Variational inference for PGMs (cont.)

The ELBO is now

$$
\begin{aligned}
\mathcal{L}[q(\mathbf{H})] &= \mathbb{E}_{q(\mathbf{H})}\left[\log \frac{p(\mathbf{H},\mathbf{V})}{q(\mathbf{H})}\right] \\
&= \int \prod_i q(\mathbf{H}_i)\left(\log p(\mathbf{H},\mathbf{V}) - \log \prod_i q(\mathbf{H}_i)\right) d\mathbf{H} \\
&= \int q(\mathbf{H}_j)\left(\int \log p(\mathbf{H},\mathbf{V}) \prod_{i\neq j} q(\mathbf{H}_i)\, d\mathbf{H}_i\right) d\mathbf{H}_j \\
&\quad - \int q(\mathbf{H}_j) \log q(\mathbf{H}_j)\, d\mathbf{H}_j + \text{const.} \\
&= \int q(\mathbf{H}_j) \log \tilde{q}(\mathbf{H}_j,\mathbf{V})\, d\mathbf{H}_j - \int q(\mathbf{H}_j)\log q(\mathbf{H}_j)\, d\mathbf{H}_j + \text{const.} \\
&= -\mathrm{KL}(q(\mathbf{H}_j) \| \tilde{q}(\mathbf{H}_j,\mathbf{V})) + \text{const.}
\end{aligned}
$$

where

$$\log \tilde{q}(\mathbf{H}_j,\mathbf{V}) = \mathbb{E}_{i\neq j}\left[\log p(\mathbf{H},\mathbf{V})\right] + \text{const.}$$

Optimizing a single factor

We can isolate a single factor for each hidden node in the graph $\mathbf{H}_j$.
This makes optimizing a single variational factor easy!

Optimizing a single factor

We can isolate a single factor for each hidden node in the graph $\mathbf{H}_j$. This makes optimizing a single variational factor easy!

$$\mathcal{L}[q(\mathbf{H})] = \mathrm{KL}(q(\mathbf{H}_j) \| \tilde{q}(\mathbf{H}_j, \mathbf{V})) + \mathrm{const.}$$

Optimizing a single factor

We can isolate a single factor for each hidden node in the graph $\mathbf{H}_j$. This makes optimizing a single variational factor easy!

$$\mathcal{L}[q(\mathbf{H})] = \mathrm{KL}(q(\mathbf{H}_j) \| \tilde{q}(\mathbf{H}_j, \mathbf{V})) + \text{const.}$$

For a single factor $q(\mathbf{H}_j)$, this equation is minimized when $q(\mathbf{H}_j) = \tilde{q}(\mathbf{H}_j, \mathbf{V})$.

Optimizing a single factor

We can isolate a single factor for each hidden node in the graph $\mathbf{H}_j$. This makes optimizing a single variational factor easy!

$$\mathcal{L}[q(\mathbf{H})] = \mathrm{KL}(q(\mathbf{H}_j) \| \tilde{q}(\mathbf{H}_j, \mathbf{V})) + \mathrm{const.}$$

For a single factor $q(\mathbf{H}_j)$, this equation is minimized when $q(\mathbf{H}_j) = \tilde{q}(\mathbf{H}_j, \mathbf{V})$.

Furthermore,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j} [\log p(\mathbf{H}, \mathbf{V})] + \mathrm{const.}$$

is a function of only factors other than $q(\mathbf{H}_j)$ and observed data.

Optimizing a single factor

We can isolate a single factor for each hidden node in the graph $\mathbf{H}_j$. This makes optimizing a single variational factor easy!

$$\mathcal{L}[q(\mathbf{H})] = \mathrm{KL}(q(\mathbf{H}_j) \| \tilde{q}(\mathbf{H}_j, \mathbf{V})) + \text{const.}$$

For a single factor $q(\mathbf{H}_j)$, this equation is minimized when $q(\mathbf{H}_j) = \tilde{q}(\mathbf{H}_j, \mathbf{V})$.

Furthermore,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j} [\log p(\mathbf{H}, \mathbf{V})] + \text{const.}$$

is a function of only factors other than $q(\mathbf{H}_j)$ and observed data.

Mean-field variational inference

Until converged, for each factor $q(\mathbf{H}_j)$, hold factors $q(\mathbf{H}_{i \neq j})$ constant and set $q(\mathbf{H}_j) = \tilde{q}(\mathbf{H}_j, \mathbf{V})$.

Conjugate-exponential graphical models

If our PGM is *conjugate-exponential*, where every node belongs in the exponential family of distributions, and is conjugate w.r.t. its parents,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j} \left[\log p(\mathbf{H}, \mathbf{V})\right] + \mathrm{const.}$$

can be calculated in closed form!

Conjugate-exponential graphical models

If our PGM is *conjugate-exponential*, where every node belongs in the exponential family of distributions, and is conjugate w.r.t. its parents,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j}\left[\log p(\mathbf{H}, \mathbf{V})\right] + \text{const.}$$

can be calculated in closed form!

An exponential family distribution is one that can be written in the following form:

$$p(x|\theta) = h(x) \exp\left\{\langle \eta_x(\theta), t_x(x)\rangle - \log Z(\eta(\theta))\right\}$$

with

Conjugate-exponential graphical models

If our PGM is *conjugate-exponential*, where every node belongs in the exponential family of distributions, and is conjugate w.r.t. its parents,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j} [\log p(\mathbf{H}, \mathbf{V})] + \text{const.}$$

can be calculated in closed form!

An exponential family distribution is one that can be written in the following form:

$$p(x|\theta) = h(x) \exp \{ \langle \eta_x(\theta), t_x(x) \rangle - \log Z(\eta(\theta)) \}$$

with

- $h(x)$: base measure

Conjugate-exponential graphical models

If our PGM is *conjugate-exponential*, where every node belongs in the exponential family of distributions, and is conjugate w.r.t. its parents,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j}\left[\log p(\mathbf{H}, \mathbf{V})\right] + \text{const.}$$

can be calculated in closed form!

An exponential family distribution is one that can be written in the following form:

$$p(x|\theta) = h(x) \exp\left\{\langle \eta_x(\theta), t_x(x) \rangle - \log Z(\eta(\theta))\right\}$$

with

- $h(x)$: base measure
- $\eta_x(\theta)$: natural parameter

Conjugate-exponential graphical models

If our PGM is *conjugate-exponential*, where every node belongs in the exponential family of distributions, and is conjugate w.r.t. its parents,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j}\left[\log p(\mathbf{H}, \mathbf{V})\right] + \text{const.}$$

can be calculated in closed form!

An exponential family distribution is one that can be written in the following form:

$$p(x|\theta) = h(x) \exp\left\{\langle \eta_x(\theta), t_x(x)\rangle - \log Z(\eta(\theta))\right\}$$

with

- $h(x)$: base measure
- $\eta_x(\theta)$: natural parameter
- $t_x(x)$: sufficient statistic

Conjugate-exponential graphical models

If our PGM is *conjugate-exponential*, where every node belongs in the exponential family of distributions, and is conjugate w.r.t. its parents,

$$\log \tilde{q}(\mathbf{H}_j, \mathbf{V}) = \mathbb{E}_{i \neq j}\left[\log p(\mathbf{H}, \mathbf{V})\right] + \text{const.}$$

can be calculated in closed form!

An exponential family distribution is one that can be written in the following form:

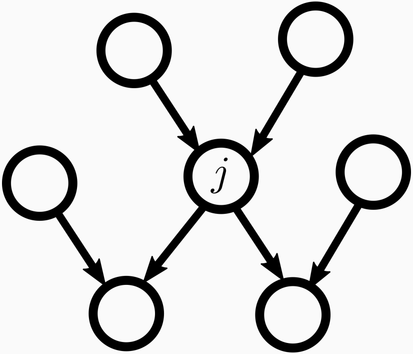
$$p(x|\theta) = h(x) \exp\left\{\langle \eta_x(\theta), t_x(x)\rangle - \log Z(\eta(\theta))\right\}$$

with

- $h(x)$: base measure
- $\eta_x(\theta)$: natural parameter
- $t_x(x)$: sufficient statistic
- $\log Z(\eta_x(\theta))$: log-partition function

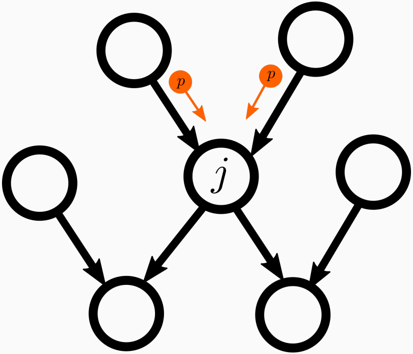
Variational message passing

We now return to graphs! If we assume a mean-field variational distribution and our PGM is conjugate-exponential, we get a very elegant graph algorithm, called *variational message passing* (VMP) [1].



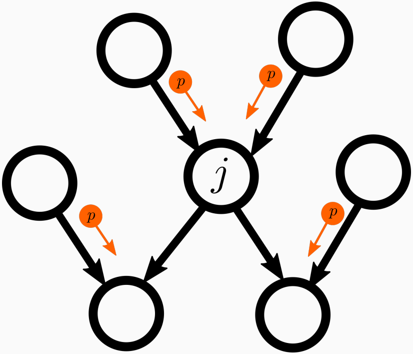
Variational message passing

We now return to graphs! If we assume a mean-field variational distribution and our PGM is conjugate-exponential, we get a very elegant graph algorithm, called *variational message passing* (VMP) [1].



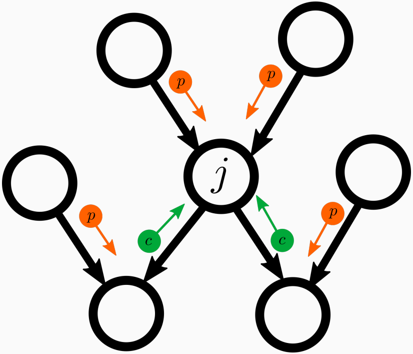
Variational message passing

We now return to graphs! If we assume a mean-field variational distribution and our PGM is conjugate-exponential, we get a very elegant graph algorithm, called *variational message passing* (VMP) [1].



Variational message passing

We now return to graphs! If we assume a mean-field variational distribution and our PGM is conjugate-exponential, we get a very elegant graph algorithm, called *variational message passing* (VMP) [1].



Messages

Remember that η_j is the natural parameter of the distribution $q(\mathbf{H}_j)$.

Messages

Remember that η_j is the natural parameter of the distribution $q(\mathbf{H}_j)$.
The message from a parent Y to a child X is

$$m_{Y \to X} = f(\eta_Y)$$

Messages

Remember that η_j is the natural parameter of the distribution $q(\mathbf{H}_j)$. The message from a parent Y to a child X is

$$m_{Y\to X} = f(\eta_Y)$$

The message from a child X to a parent Y is

$$m_{X\to Y} = g(\eta_X, \{m_{i\to X}\}_{i\in \mathrm{cp}_Y})$$

Messages

Remember that η_j is the natural parameter of the distribution $q(\mathbf{H}_j)$.
The message from a parent Y to a child X is

$$m_{Y \to X} = f(\eta_Y)$$

The message from a child X to a parent Y is

$$m_{X \to Y} = g(\eta_X, \{m_{i \to X}\}_{i \in \mathrm{cp}_Y})$$

In conjugate-exponential PGMs, messages can be computed in closed-form.

Summary of VMP

Setup: Conjugate-exponential graphical model

Summary of VMP

Setup: Conjugate-exponential graphical model

Problem: Compute posterior $p(\mathbf{H}|\mathbf{V})$, approximated with $q(\mathbf{H}) = \prod_j q(\mathbf{H}_j)$

Summary of VMP

Setup: Conjugate-exponential graphical model

Problem: Compute posterior $p(\mathbf{H}|\mathbf{V})$, approximated with $q(\mathbf{H}) = \prod_j q(\mathbf{H}_j)$

Solution: Until converged, for each hidden node $\mathbf{H}_j$:

Summary of VMP

Setup: Conjugate-exponential graphical model

Problem: Compute posterior $p(\mathbf{H}|\mathbf{V})$, approximated with $q(\mathbf{H}) = \prod_j q(\mathbf{H}_j)$

Solution: Until converged, for each hidden node $\mathbf{H}_j$:

1. Collect messages from children and parents

Summary of VMP

Setup: Conjugate-exponential graphical model

Problem: Compute posterior $p(\mathbf{H}|\mathbf{V})$, approximated with $q(\mathbf{H}) = \prod_j q(\mathbf{H}_j)$

Solution: Until converged, for each hidden node $\mathbf{H}_j$:

1. Collect messages from children and parents
2. Compute updated distribution parameters from messages

Summary of VMP

Setup: Conjugate-exponential graphical model

Problem: Compute posterior $p(\mathbf{H}|\mathbf{V})$, approximated with $q(\mathbf{H}) = \prod_j q(\mathbf{H}_j)$

Solution: Until converged, for each hidden node $\mathbf{H}_j$:

1. Collect messages from children and parents
2. Compute updated distribution parameters from messages

Benefits: efficient, simple, can incorporate mini-batches (stochastic variational inference)

Summary of VMP

Setup: Conjugate-exponential graphical model

Problem: Compute posterior $p(\mathbf{H}|\mathbf{V})$, approximated with $q(\mathbf{H}) = \prod_j q(\mathbf{H}_j)$

Solution: Until converged, for each hidden node $\mathbf{H}_j$:

1. Collect messages from children and parents
2. Compute updated distribution parameters from messages

Benefits: efficient, simple, can incorporate mini-batches (stochastic variational inference)

Drawbacks: can be underexpressive (conjugate-exponential requirement)

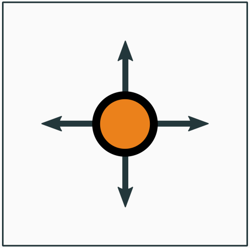
Structured variational autoencoder

Model learning

Recall the model learning problem.

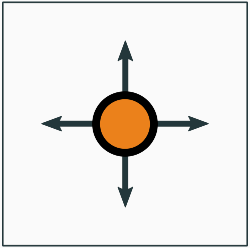
Model learning

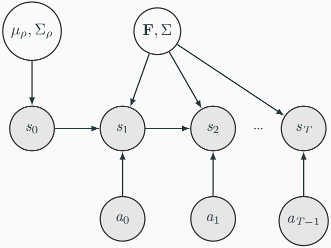
Recall the model learning problem.



Model learning

Recall the model learning problem.



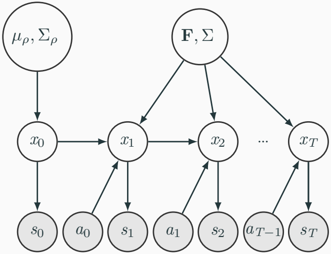


Adding expressivity

One way of making the Bayesian LDS more expressive is to add a observation model.

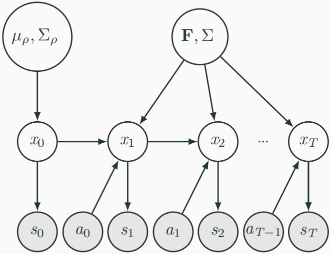
Adding expressivity

One way of making the Bayesian LDS more expressive is to add a observation model.



Adding expressivity

One way of making the Bayesian LDS more expressive is to add a observation model.



What if this observation model was a neural network?

Structured variational autoencoder

We augment the Bayesian LDS with a neural network observation model.

Structured variational autoencoder

We augment the Bayesian LDS with a neural network observation model.

$$\mu_\rho, \Sigma_\rho \sim \mathcal{NIW}(\Psi, \nu, \mu_0, \kappa)\,, \quad \mathbf{F}, \Sigma \ \sim \mathcal{MNIW}(\Psi, \nu, \boldsymbol{M}_0, \boldsymbol{V})\,,$$

$$\mathbf{x}_0 \mid \mu_\rho, \Sigma_\rho \sim \mathcal{N}(\mu_\rho, \Sigma_\rho)\,, \qquad \mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t \sim \mathcal{N}\left(\mathbf{F}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{a}_t\end{bmatrix}, \Sigma\right) \ \text{for} \ t \in [0, \ldots, T]$$

$$\mathbf{s}_t \mid \mathbf{x}_t \sim \mathcal{N}\left(\mu_\gamma(\mathbf{x}_t), \Sigma_\gamma(\mathbf{x}_t)\right) \ \text{for} \ t \in [0, \ldots, T]$$

where $\mu_\gamma(\mathbf{x}_t)$ and $\Sigma_\gamma(\mathbf{x}_t)$ are both neural networks parametrized by γ. This model is called a structured variational autoencoder (SVAE) [2].

Structured variational autoencoder

We augment the Bayesian LDS with a neural network observation model.

$$\mu_\rho, \Sigma_\rho \sim \mathcal{NIW}(\Psi, \nu, \mu_0, \kappa)\,, \quad \mathbf{F}, \Sigma \;\sim \mathcal{MNIW}(\Psi, \nu, \boldsymbol{M}_0, \boldsymbol{V})\,,$$

$$\mathbf{x}_0 \mid \mu_\rho, \Sigma_\rho \sim \mathcal{N}(\mu_\rho, \Sigma_\rho)\,, \qquad \mathbf{x}_{t+1} \mid \mathbf{x}_t, \mathbf{a}_t \sim \mathcal{N}\left(\mathbf{F}\begin{bmatrix}\mathbf{x}_t \\ \mathbf{a}_t\end{bmatrix}, \Sigma\right) \text{ for } t \in [0, \ldots, T]$$

$$\mathbf{s}_t \mid \mathbf{x}_t \sim \mathcal{N}\left(\mu_\gamma(\mathbf{x}_t), \Sigma_\gamma(\mathbf{x}_t)\right) \text{ for } t \in [0, \ldots, T]$$

where $\mu_\gamma(\mathbf{x}_t)$ and $\Sigma_\gamma(\mathbf{x}_t)$ are both neural networks parametrized by γ. This model is called a structured variational autoencoder (SVAE) [2].

But what does it do?

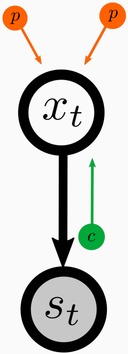
Inference in SVAE

First of all, a neural network is neither conjugate nor exponential.
How do we perform inference?

Inference in SVAE

First of all, a neural network is neither conjugate nor exponential. How do we perform inference?

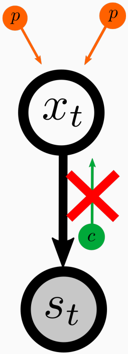
Strategy: Run VMP, but use “fake” messages for the neural network observation model.



Inference in SVAE

First of all, a neural network is neither conjugate nor exponential. How do we perform inference?

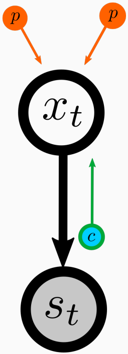
Strategy: Run VMP, but use “fake” messages for the neural network observation model.



Inference in SVAE

First of all, a neural network is neither conjugate nor exponential. How do we perform inference?

Strategy: Run VMP, but use "fake" messages for the neural network observation model.



Messages in SVAE

The message from a **non-conjugate, non-exponential family** observation X to a parent Y is

$$m_{X \to Y} = r_\xi(t_X(X))$$

where r_ξ is a neural network whose output is the same shape of a message from a conjugate-exponential child.

Messages in SVAE

The message from a **non-conjugate, non-exponential family** observation X to a parent Y is

$$m_{X \to Y} = r_\xi(t_X(X))$$

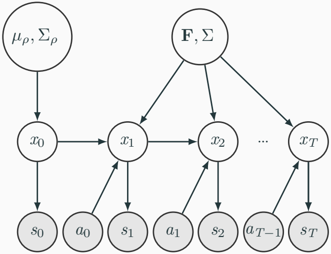
where r_ξ is a neural network whose output is the same shape of a message from a conjugate-exponential child.

Inference in a SVAE

1. For a data $\tau = \{s_0, a_0, s_1, a_1, \ldots, s_T\}$, perform VMP using SVAE messages
2. Compute the ELBO $\mathcal{L}[q(\{x_i\}_{i=1}^T, \mu_\rho, \Sigma_\rho, \mathbf{F}, \Sigma)]$
3. Update neural networks with $\nabla_{\gamma,\xi}\mathcal{L}[q(\{x_i\}_{i=1}^T, \mu_\rho, \Sigma_\rho, \mathbf{F}, \Sigma)]$
4. Update global parameters $(\mu_\rho, \Sigma_\rho, \mathbf{F}, \Sigma)$ with natural gradients

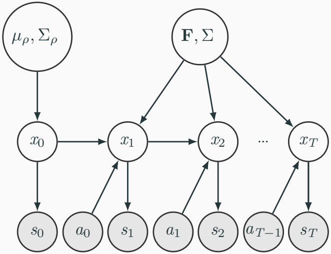
Autoencoding

How is a SVAE an *autoencoder*?



Autoencoding

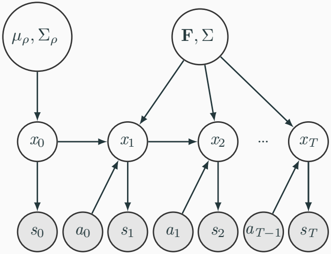
How is a SVAE an *autoencoder*?



We have two neural networks:

Autoencoding

How is a SVAE an *autoencoder*?

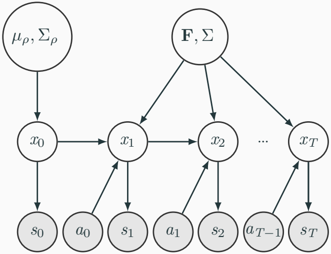


We have two neural networks:

1. $r_\xi(s)$: recognition network, takes real data and "encodes" it

Autoencoding

How is a SVAE an *autoencoder*?



We have two neural networks:

1. $r_\xi(s)$: recognition network, takes real data and "encodes" it
2. $\mu_\gamma(x), \Sigma_\gamma(x)$: observation network, takes latent data and "decodes" it

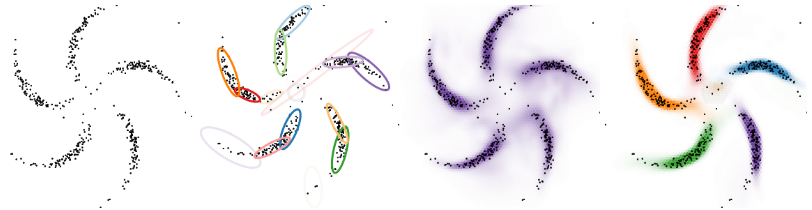
Conclusion

Why SVAE?

The SVAE naturally applies to scenarios where there is already a tractable PGM.

Why SVAE?

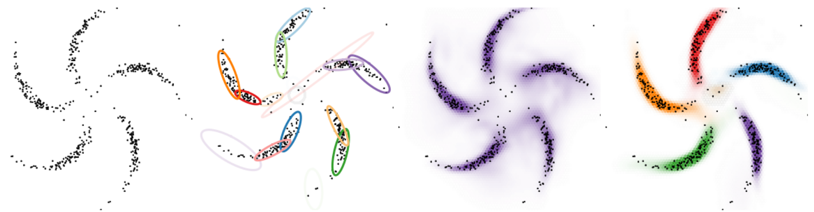
The SVAE naturally applies to scenarios where there is already a tractable PGM.



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

Why SVAE?

The SVAE naturally applies to scenarios where there is already a tractable PGM.



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

In this scenario, the SVAE enables modeling non-Gaussian cluster shapes [2].

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

[1] Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

General approach:

- Learn a latent LDS
- Use MPC to control an agent in the latent space

[1] Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

General approach:

- Learn a latent LDS
- Use MPC to control an agent in the latent space

Benefits of this approach:

[1] Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

General approach:

- Learn a latent LDS
- Use MPC to control an agent in the latent space

Benefits of this approach:

- We can learn simple dynamics even with camera data

[1] Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

General approach:

- Learn a latent LDS
- Use MPC to control an agent in the latent space

Benefits of this approach:

- We can learn simple dynamics even with camera data
- Model-based RL tends to be more sample efficient

[1] Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

General approach:

- Learn a latent LDS
- Use MPC to control an agent in the latent space

Benefits of this approach:

- We can learn simple dynamics even with camera data
- Model-based RL tends to be more sample efficient

[1] Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the SVAE to learn latent models to be used in reinforcement learning[1].

General approach:

- Learn a latent LDS
- Use MPC to control an agent in the latent space

Benefits of this approach:

- We can learn simple dynamics even with camera data
- Model-based RL tends to be more sample efficient

Demo

[1] Joint work with Marvin Zhang from UC Berkeley

Questions?

References

[1] J. Winn and C. Bishop. Variational message passing. *JMLR*, 2005.

[2] M. Johnson, D. Duvenaud, A. Wiltschko, S. Datta, and R. Adams. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2016.