

Structured variational autoencoders (SVAE) and beyond

Sharad Vikram

October 20, 2017

UCSD

Variational inference

Variational inference

- Variational message passing

- Gradient-based variational inference

Structured variational autoencoder

Conclusion

- Applications

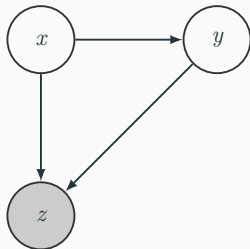
- Current work

Bayesian inference

In Bayesian inference, we compute the posterior distribution of observations given data.

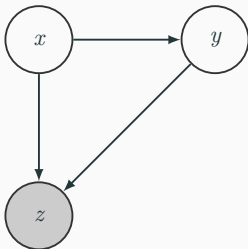
Bayesian inference

In Bayesian inference, we compute the posterior distribution of observations given data.



Bayesian inference

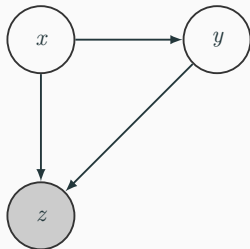
In Bayesian inference, we compute the posterior distribution of observations given data.



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

Bayesian inference

In Bayesian inference, we compute the posterior distribution of observations given data.



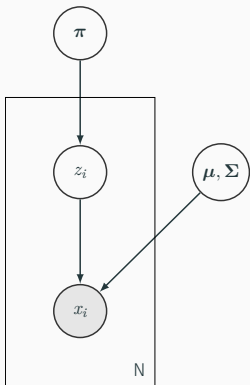
We are interested in the posterior distribution $p(x, y|z)$

This can be computed via Bayes rule:

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

Example PGM

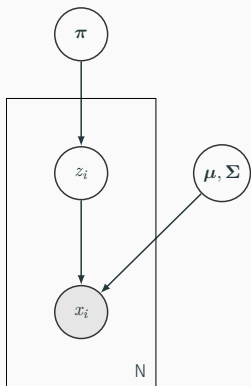
Latent variable model: Gaussian mixture model



$$\begin{aligned}\pi &\sim \text{Dir}(\alpha) \\ \mu_k, \Sigma_k &\sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu) \\ z_i | \pi &\sim \text{Cat}(\pi) \\ x_i | z_i, \mu, \Sigma &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})\end{aligned}$$

Example PGM

Latent variable model: Gaussian mixture model

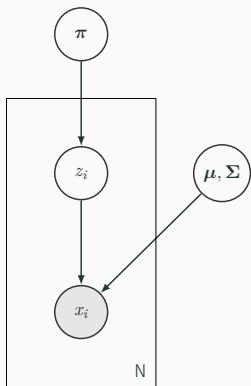


$$\begin{aligned}\pi &\sim \text{Dir}(\alpha) \\ \mu_k, \Sigma_k &\sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu) \\ z_i | \pi &\sim \text{Cat}(\pi) \\ x_i | z_i, \boldsymbol{\mu}, \boldsymbol{\Sigma} &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})\end{aligned}$$

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

Example PGM

Latent variable model: Gaussian mixture model



$$\begin{aligned}\pi &\sim \text{Dir}(\alpha) \\ \mu_k, \Sigma_k &\sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu) \\ z_i | \pi &\sim \text{Cat}(\pi) \\ x_i | z_i, \mu, \Sigma &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})\end{aligned}$$

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

Global variables: π, μ, Σ

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

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

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

where

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

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

where

- $h(x)$ - base measure

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

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

where

- $h(x)$ - base measure
- $\eta(\theta)$ - natural parameter function

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

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

where

- $h(x)$ - base measure
- $\eta(\theta)$ - natural parameter function
- $t_x(x)$ - sufficient statistic function

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

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

where

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

Exponential family

A probability distribution is in the *exponential family* if it can be parametrized in the following way.

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

where

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

Examples: Gaussian, Categorical, Dirichlet, inverse-Wishart

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 methods:

- 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 methods:

- 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 methods:

- 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 methods:

- 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 methods:

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

Variational inference at a high level

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

Variational inference at a high level

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

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

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

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

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

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

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

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 $\text{KL}(q_{\phi}(\theta, z) || p(\theta, z|x))$

Variational inference at a high level

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

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 $\text{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$
- Minimize loss $\phi^* = \operatorname{argmin}_\phi \text{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$

Variational inference at a high level

For graphical models where we can't compute the posterior analytically, **variational inference** is a viable approach.

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 $\text{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$
- Minimize loss $\phi^* = \text{argmin}_\phi \text{KL}(q_\phi(\theta, z) \| p(\theta, z|x))$

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 $p(x)$,

$$\text{KL}(q(x)\|p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{p(x)} \right]$$

KL-divergence

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

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

$$\text{KL}(q(x)\|p(x)) = \int q(x) \log \frac{q(x)}{p(x)} dx = \mathbb{E}_{q(x)} \left[\log \frac{q(x)}{p(x)} \right]$$

Properties:

- $\text{KL}(q(x)\|p(x)) = 0$ 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

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

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

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

and maximize the *evidence lower bound* (ELBO)

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

Picking $q(\theta, z)$

How do we pick a $q(\theta, z)$?

Picking $q(\theta, z)$

How do we pick a $q(\theta, z)$?

In general, a broader $q(\theta, z)$ is harder to optimize.

Options:

Picking $q(\theta, z)$

How do we pick a $q(\theta, z)$?

In general, a broader $q(\theta, z)$ is harder to optimize.

Options:

- Mean-field ($q(\theta, z) = q(\theta) \prod_i q(z_i)$)

Picking $q(\theta, z)$

How do we pick a $q(\theta, z)$?

In general, a broader $q(\theta, z)$ is harder to optimize.

Options:

- Mean-field ($q(\theta, z) = q(\theta) \prod_i q(z_i)$)
- Structured mean-field ($q(\theta, z) = q(\theta)q(z)$)

Picking $q(\theta, z)$

How do we pick a $q(\theta, z)$?

In general, a broader $q(\theta, z)$ is harder to optimize.

Options:

- Mean-field ($q(\theta, z) = q(\theta) \prod_i q(z_i)$)
- Structured mean-field ($q(\theta, z) = q(\theta)q(z)$)
- Differentiable q

Mean-field variational inference

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

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

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

Mean-field variational inference

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

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

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.

Mean-field variational inference

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

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

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

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

Mean-field variational inference (cont.)

The ELBO is now

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

where

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

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!

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

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!

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

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

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!

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

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

Furthermore,

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

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

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!

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

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

Furthermore,

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

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 $q(\mathbf{H}_j) = \tilde{f}(\mathbf{H}_j, \mathbf{V})$.

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

There is a deep relationship between conjugacy and linearity.

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

There is a deep relationship between conjugacy and linearity. For a conjugate-exponential model you can

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

There is a deep relationship between conjugacy and linearity. For a conjugate-exponential model you can

- Take a model $p(\mathbf{X})$ and pick a variable X_j .

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

There is a deep relationship between conjugacy and linearity. For a conjugate-exponential model you can

- Take a model $p(\mathbf{X})$ and pick a variable X_j .
- Isolate all occurrences of X_j in the log density $\log p(\mathbf{X})$.

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

There is a deep relationship between conjugacy and linearity. For a conjugate-exponential model you can

- Take a model $p(\mathbf{X})$ and pick a variable X_j .
- Isolate all occurrences of X_j in the log density $\log p(\mathbf{X})$.
- Rewrite $\log p(\mathbf{X}) = \langle t_{X_j}(X_j), \eta_{X_j} \rangle + \text{const.}$.

Conjugate-exponential graphical models

A model is *conjugate-exponential* if every node is exponential family and conjugate w.r.t. its parents.

There is a deep relationship between conjugacy and linearity. For a conjugate-exponential model you can

- Take a model $p(\mathbf{X})$ and pick a variable X_j .
- Isolate all occurrences of X_j in the log density $\log p(\mathbf{X})$.
- Rewrite $\log p(\mathbf{X}) = \langle t_{X_j}(X_j), \eta_{X_j} \rangle + \text{const.}$.

Furthermore, the complete-conditional $p(X_j | X_{i \neq j})$ will be in the same (exponential) family as $p(X_j | \boldsymbol{\pi}_{X_j})$ ($\boldsymbol{\pi}_{X_j}$ are the parents of X_j).

Model assumptions

If our model is *conjugate-exponential*,

$$\begin{aligned}\log \tilde{f}(\mathbf{H}_j, \mathbf{V}) &= \mathbb{E}_{i \neq j} [\log p(\mathbf{H}, \mathbf{V})] + \text{const.} \\ &= \langle t_{\mathbf{H}_j}(\mathbf{H}_j), \mathbb{E}_{i \neq j} [\eta_{\mathbf{H}_j}] \rangle + \text{const.}\end{aligned}$$

Model assumptions

If our model is *conjugate-exponential*,

$$\begin{aligned}\log \tilde{f}(\mathbf{H}_j, \mathbf{V}) &= \mathbb{E}_{i \neq j} [\log p(\mathbf{H}, \mathbf{V})] + \text{const.} \\ &= \langle t_{\mathbf{H}_j}(\mathbf{H}_j), \mathbb{E}_{i \neq j} [\eta_{\mathbf{H}_j}] \rangle + \text{const.}\end{aligned}$$

We can then calculate the optimal parameter for the variational distribution $q(\mathbf{H}_j)^*$.

$$\eta_{\mathbf{H}_j}^* = \mathbb{E}_{i \neq j} [\eta_{\mathbf{H}_j}]$$

Model assumptions

If our model is *conjugate-exponential*,

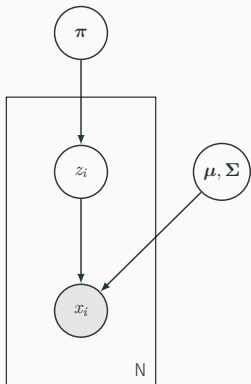
$$\begin{aligned}\log \tilde{f}(\mathbf{H}_j, \mathbf{V}) &= \mathbb{E}_{i \neq j} [\log p(\mathbf{H}, \mathbf{V})] + \text{const.} \\ &= \langle t_{\mathbf{H}_j}(\mathbf{H}_j), \mathbb{E}_{i \neq j} [\eta_{\mathbf{H}_j}] \rangle + \text{const.}\end{aligned}$$

We can then calculate the optimal parameter for the variational distribution $q(\mathbf{H}_j)^*$.

$$\eta_{\mathbf{H}_j}^* = \mathbb{E}_{i \neq j} [\eta_{\mathbf{H}_j}]$$

Computing $\mathbb{E}_{i \neq j} [\eta_{\mathbf{H}_j}]$ involves a computation of expectations from \mathbf{H}_j 's Markov blanket.

Mean-field for GMM



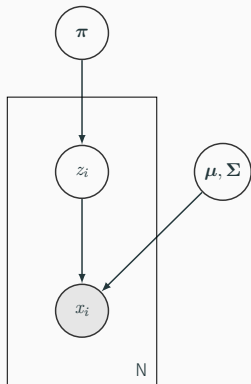
$$\pi \sim \text{Dir}(\alpha)$$

$$\mu_k, \Sigma_k \sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu)$$

$$z_i | \pi \sim \text{Cat}(\pi)$$

$$x_i | z_i, \mu, \Sigma \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

Mean-field for GMM



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

$$\mu_k, \Sigma_k \sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu)$$

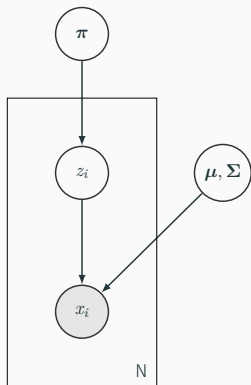
$$z_i | \pi \sim \text{Cat}(\pi)$$

$$x_i | z_i, \mu, \Sigma \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

Variational posterior:

$$q(\theta, z) = q(\pi) \left(\prod_{k=1}^K q(\mu_k, \Sigma_k) \right) \left(\prod_i q(z_i) \right)$$

Mean-field for GMM



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

$$\mu_k, \Sigma_k \sim \mathcal{N}\mathcal{IW}(\psi, \mu_0, \kappa, \nu)$$

$$z_i | \pi \sim \text{Cat}(\pi)$$

$$x_i | z_i, \mu, \Sigma \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

Variational posterior:

$$q(\theta, z) = q(\pi) \left(\prod_{k=1}^K q(\mu_k, \Sigma_k) \right) \left(\prod_i q(z_i) \right)$$

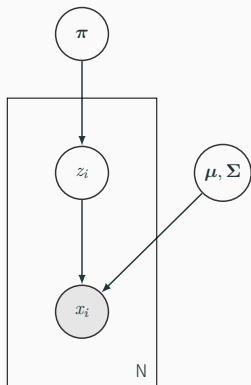
Variational distributions:

$$q(\pi) = \text{Dir}(\alpha')$$

$$q(\mu_k, \Sigma_k) = \mathcal{N}\mathcal{IW}(\psi'_k, \mu'_{0k}, \kappa'_k, \nu'_k)$$

$$q(z_i) = \text{Cat}(p'_i)$$

Mean-field for GMM



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

$$\mu_k, \Sigma_k \sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu)$$

$$z_i | \pi \sim \text{Cat}(\pi)$$

$$x_i | z_i, \mu, \Sigma \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

Variational posterior:

$$q(\theta, z) = q(\pi) \left(\prod_{k=1}^K q(\mu_k, \Sigma_k) \right) \left(\prod_i q(z_i) \right)$$

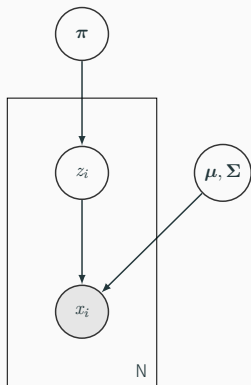
Variational distributions:

$$q(\pi) = \text{Dir}(\alpha')$$

$$q(\mu_k, \Sigma_k) = \mathcal{NIW}(\psi'_k, \mu'_{0k}, \kappa'_k, \nu'_k)$$

$$q(z_i) = \text{Cat}(p'_i)$$

Mean-field for GMM



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

$$\mu_k, \Sigma_k \sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu)$$

$$z_i | \pi \sim \text{Cat}(\pi)$$

$$x_i | z_i, \mu, \Sigma \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

Variational posterior:

$$q(\theta, z) = q(\pi) \left(\prod_{k=1}^K q(\mu_k, \Sigma_k) \right) \left(\prod_i q(z_i) \right)$$

Variational distributions:

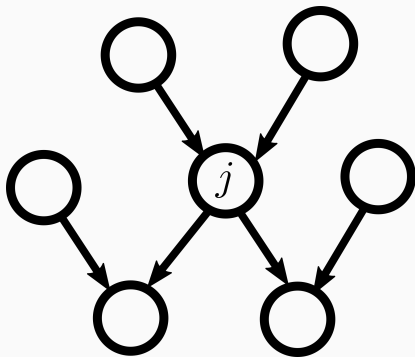
$$q(\pi) = \text{Dir}(\alpha')$$

$$q(\mu_k, \Sigma_k) = \mathcal{NIW}(\psi'_k, \mu'_{0k}, \kappa'_k, \nu'_k)$$

$$q(z_i) = \text{Cat}(p'_i)$$

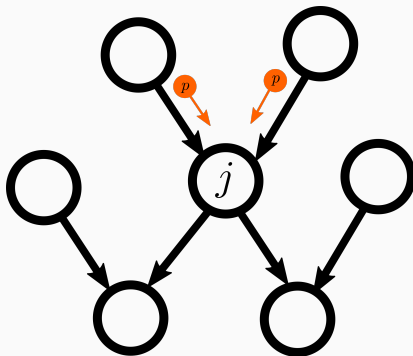
Variational message passing

An computation for variable X_j relies on only its Markov blanket.



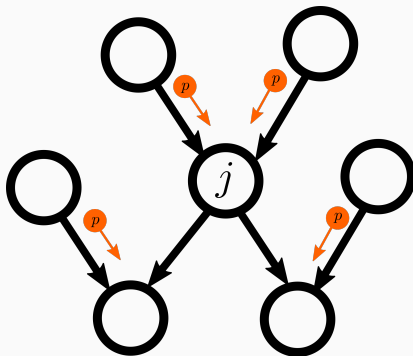
Variational message passing

An computation for variable X_j relies on only its Markov blanket.



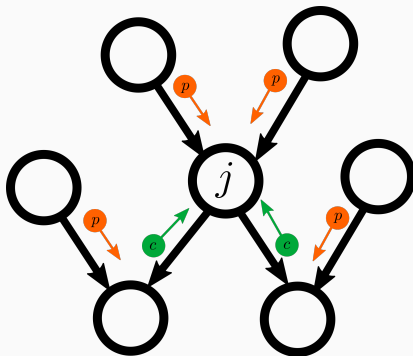
Variational message passing

An computation for variable X_j relies on only its Markov blanket.



Variational message passing

An computation for variable X_j relies on only its Markov blanket.



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

Messages

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

$$m_{Y \rightarrow X} = \mathbb{E}_{q(Y)} [t_Y(Y)]$$

Messages

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

$$m_{Y \rightarrow X} = \mathbb{E}_{q(Y)} [t_Y(Y)]$$

The message from a child X to a parent Y is

$$m_{X \rightarrow Y} = g(\mathbb{E}_{q(X)} [t_X(X)], \{m_{i \rightarrow X}\}_{i \in \text{cp}_Y})$$

Messages

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

$$m_{Y \rightarrow X} = \mathbb{E}_{q(Y)} [t_Y(Y)]$$

The message from a child X to a parent Y is

$$m_{X \rightarrow Y} = g(\mathbb{E}_{q(X)} [t_X(X)], \{m_{i \rightarrow X}\}_{i \in \text{cp}_Y})$$

The optimal parameter is

$$\eta_{\mathbf{H}_j}^* = f(\{m_{p \rightarrow \mathbf{H}_j}\}_{p \in \pi(\mathbf{H}_j)}) + \sum_{c \in \text{ch}(\mathbf{H}_j)} m_{c \rightarrow \mathbf{H}_j}$$

Messages

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

$$m_{Y \rightarrow X} = \mathbb{E}_{q(Y)} [t_Y(Y)]$$

The message from a child X to a parent Y is

$$m_{X \rightarrow Y} = g(\mathbb{E}_{q(X)} [t_X(X)], \{m_{i \rightarrow X}\}_{i \in \text{cp}_Y})$$

The optimal parameter is

$$\eta_{\mathbf{H}_j}^* = f(\{m_{p \rightarrow \mathbf{H}_j}\}_{p \in \pi(\mathbf{H}_j)}) + \sum_{c \in \text{ch}(\mathbf{H}_j)} m_{c \rightarrow \mathbf{H}_j}$$

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

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)

Extension: stochastic VMP

Goal: incorporate mini-batches into VMP to scale to large datasets

Extension: stochastic VMP

Goal: incorporate mini-batches into VMP to scale to large datasets

Ideas:

Extension: stochastic VMP

Goal: incorporate mini-batches into VMP to scale to large datasets
Ideas:

1. Mini-batches correspond to local latent variables. We can do inference in these using VMP.

Extension: stochastic VMP

Goal: incorporate mini-batches into VMP to scale to large datasets

Ideas:

1. Mini-batches correspond to local latent variables. We can do inference in these using VMP.
2. Harder task: global variables. Solution: natural gradient updates

Extension: stochastic VMP

Goal: incorporate mini-batches into VMP to scale to large datasets

Ideas:

1. Mini-batches correspond to local latent variables. We can do inference in these using VMP.
2. Harder task: global variables. Solution: natural gradient updates

A natural gradient update is

$$\tilde{\nabla}_{\theta} \mathcal{L} = \eta_{\theta}^0 + \left(N \sum_{c \in \text{ch}(\theta)} m_{c \rightarrow \theta} \right) - \eta_{\theta}$$

and only works for conjugate-exponential models.

Demo

Monte-Carlo approach

The previous approach was *restrictive*! Coordinate ascent only works on conjugate-exponential models with the mean-field assumption.

Monte-Carlo approach

The previous approach was *restrictive*! Coordinate ascent only works on conjugate-exponential models with the mean-field assumption.

Assume a non-conjugate, non-exponential model and a differentiable, sampleable $q(\theta, z)$.

Monte-Carlo approach

The previous approach was *restrictive*! Coordinate ascent only works on conjugate-exponential models with the mean-field assumption.

Assume a non-conjugate, non-exponential model and a differentiable, sampleable $q(\theta, z)$.

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

Monte-Carlo approach

The previous approach was *restrictive*! Coordinate ascent only works on conjugate-exponential models with the mean-field assumption.

Assume a non-conjugate, non-exponential model and a differentiable, sampleable $q(\theta, z)$.

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

Monte-Carlo ELBO with l samples from $q(\theta, z)$:

$$\hat{\mathcal{L}}[q(\theta, z)] = \sum_{l=1}^L q(\theta^{(l)}, z^{(l)}) \left[\log \frac{p(x, \theta^{(l)}, z^{(l)})}{q(\theta^{(l)}, z^{(l)})} \right]$$

Gradient-based approaches

1. Start with Monte-Carlo loss

$$\hat{\mathcal{L}}[q(\theta, z)] = \frac{1}{S} \sum_{s=1}^S q(\theta^{(s)}, z^{(s)}) \left[\log \frac{p(x, \theta^{(s)}, z^{(s)})}{q(\theta^{(s)}, z^{(s)})} \right]$$

2. Compute gradient

$$\nabla_{\phi} \hat{\mathcal{L}}[q_{\phi}(\theta, z)] = \frac{1}{S} \sum_{s=1}^S \nabla_{\phi} q(\theta^{(s)}, z^{(s)}) \left[\log \frac{p(x, \theta^{(s)}, z^{(s)})}{q(\theta^{(s)}, z^{(s)})} \right]$$

3. Perform gradient ascent

Issues with gradient-based approaches

Core problem: high-variance gradients

Core problem: high-variance gradients

How do we address this?

- Rao-Blackwellization (replace $\mathbb{E}[f(X, Y)]$ with $\mathbb{E}[f(X, Y)|X]$)[1]
- Reparametrization trick [2]

Variational autoencoder

The VAE is a generative model for data.

$$z_i \sim \mathcal{N}(0, I)$$

$$x_i \sim \mathcal{N}(\mu_\gamma(z_i), \Sigma_\gamma(z_i))$$

where μ_γ and Σ_γ are neural networks parameterized by γ .

Variational autoencoder

The VAE is a generative model for data.

$$z_i \sim \mathcal{N}(0, I)$$

$$x_i \sim \mathcal{N}(\mu_\gamma(z_i), \Sigma_\gamma(z_i))$$

where μ_γ and Σ_γ are neural networks parameterized by γ .



Variational autoencoder

The VAE is a generative model for data.

$$z_i \sim \mathcal{N}(0, I)$$

$$x_i \sim \mathcal{N}(\mu_\gamma(z_i), \Sigma_\gamma(z_i))$$

where μ_γ and Σ_γ are neural networks parameterized by γ .



How do we do inference?

Variational autoencoder

Basic strategy: gradient-based variational inference

Variational autoencoder

Basic strategy: gradient-based variational inference

- Pick $q_{\phi}(z|x)$ to be a *neural network* parameterized by ϕ (both differentiable and sampleable)

Variational autoencoder

Basic strategy: gradient-based variational inference

- Pick $q_\phi(z|x)$ to be a *neural network* parameterized by ϕ (both differentiable and sampleable)

Let $r_\phi(x)$ be a neural network (with weights ϕ) that outputs the parameters to a distribution, for example Gaussian.

$$q_\phi(z|x) = \mathcal{N}(r_\phi(x))$$

Variational autoencoder

Basic strategy: gradient-based variational inference

- Pick $q_\phi(z|x)$ to be a *neural network* parameterized by ϕ (both differentiable and sampleable)

Let $r_\phi(x)$ be a neural network (with weights ϕ) that outputs the parameters to a distribution, for example Gaussian.

$$q_\phi(z|x) = \mathcal{N}(r_\phi(x))$$

- Use the reparametrization trick to lower variance of gradients

Variational autoencoder

Basic strategy: gradient-based variational inference

- Pick $q_\phi(z|x)$ to be a *neural network* parameterized by ϕ (both differentiable and sampleable)

Let $r_\phi(x)$ be a neural network (with weights ϕ) that outputs the parameters to a distribution, for example Gaussian.

$$q_\phi(z|x) = \mathcal{N}(r_\phi(x))$$

- Use the reparametrization trick to lower variance of gradients

We learn the weights for the two neural networks $(\mu_\gamma(z), \Sigma_\gamma(z))$ and $r_\phi(x)$.

Variational autoencoder

Basic strategy: gradient-based variational inference

- Pick $q_\phi(z|x)$ to be a *neural network* parameterized by ϕ (both differentiable and sampleable)

Let $r_\phi(x)$ be a neural network (with weights ϕ) that outputs the parameters to a distribution, for example Gaussian.

$$q_\phi(z|x) = \mathcal{N}(r_\phi(x))$$

- Use the reparametrization trick to lower variance of gradients

We learn the weights for the two neural networks $(\mu_\gamma(z), \Sigma_\gamma(z))$ and $r_\phi(x)$.

$$\mathcal{L}[q(z|x)] = \mathbb{E}_{q(z|x)} [\log p(x, z) - \log q(z|x)]$$

Structured variational autoencoder

Variational inference

- Variational message passing

- Gradient-based variational inference

Structured variational autoencoder

Conclusion

- Applications

- Current work

What is an SVAE?

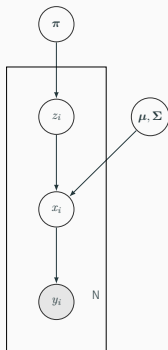
There are two ways of thinking about it

- A conjugate-exponential graphical model augmented with a neural network observation model
- A VAE augmented with a graphical model

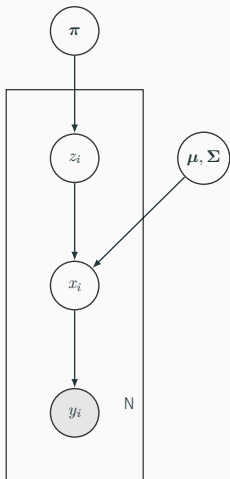
What is an SVAE?

There are two ways of thinking about it

- A conjugate-exponential graphical model augmented with a neural network observation model
- A VAE augmented with a graphical model



SVAE GMM



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

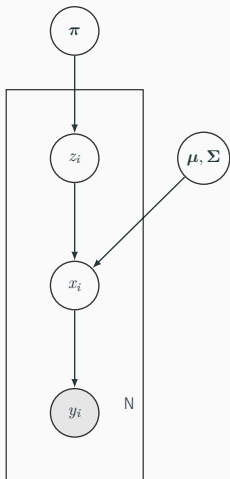
$$\mu_k, \Sigma_k \sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu)$$

$$z_i | \pi \sim \text{Cat}(\pi)$$

$$x_i | z_i, \mu, \Sigma \sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i})$$

$$y_i | x_i \sim \mathcal{N}(\mu_\gamma(x_i), \Sigma(x_i))$$

SVAE GMM



$$\begin{aligned}\pi &\sim \text{Dir}(\alpha) \\ \mu_k, \Sigma_k &\sim \mathcal{NIW}(\psi, \mu_0, \kappa, \nu) \\ z_i | \pi &\sim \text{Cat}(\pi) \\ x_i | z_i, \mu, \Sigma &\sim \mathcal{N}(\mu_{z_i}, \Sigma_{z_i}) \\ y_i | x_i &\sim \mathcal{N}(\mu_\gamma(x_i), \Sigma(x_i))\end{aligned}$$

How do we do inference?

Inference in SVAE

Inference in SVAE is a hybrid of gradient-based methods and coordinate-ascent.

Inference in SVAE

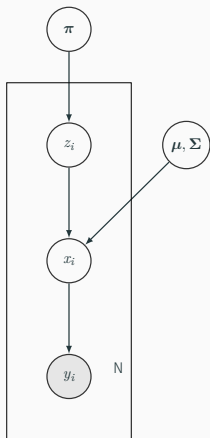
Inference in SVAE is a hybrid of gradient-based methods and coordinate-ascent.

How do we merge these two different approaches?

Inference in SVAE

Inference in SVAE is a hybrid of gradient-based methods and coordinate-ascent.

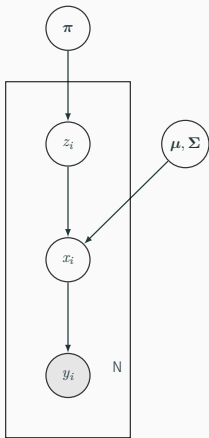
How do we merge these two different approaches?



Inference in SVAE

Inference in SVAE is a hybrid of gradient-based methods and coordinate-ascent.

How do we merge these two different approaches?

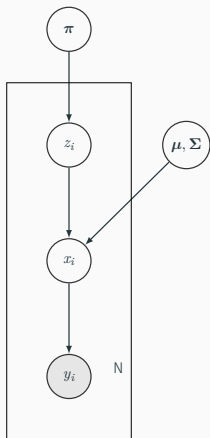


- Learn $q(z_i)$, $q(x_i)$, $q(\pi)$, $q(\mu, \Sigma)$ with stochastic VMP using a neural network $r_\phi(y) = m_{y_i \rightarrow x_i}$

Inference in SVAE

Inference in SVAE is a hybrid of gradient-based methods and coordinate-ascent.

How do we merge these two different approaches?



- Learn $q(z_i), q(x_i), q(\pi), q(\mu, \Sigma)$ with stochastic VMP using a neural network $r_\phi(y) = m_{y_i \rightarrow x_i}$
- Learn weights of neural networks $(\mu_\gamma, \Sigma_\gamma), r_\phi$ using the gradient of the Monte Carlo ELBO

Conclusion

Variational inference

- Variational message passing

- Gradient-based variational inference

Structured variational autoencoder

Conclusion

- Applications

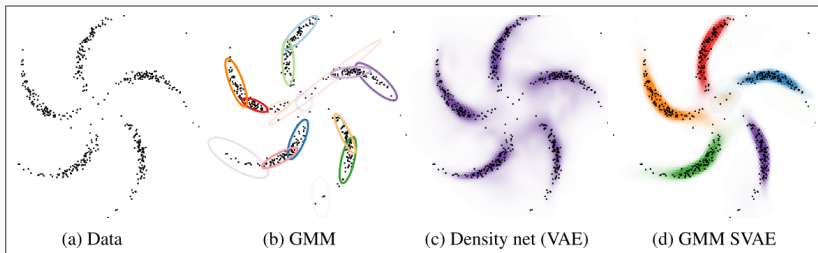
- Current work

Why SVAE?

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

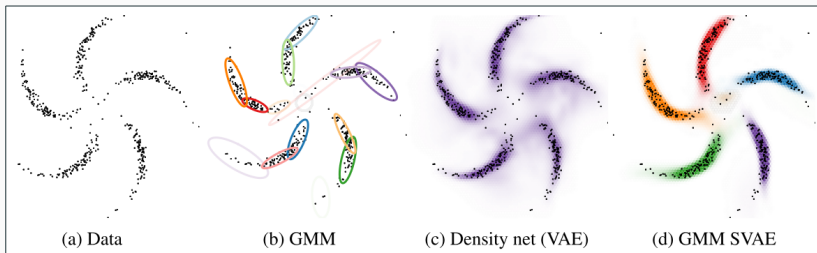
Why SVAE?

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



Why SVAE?

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



In this scenario, the SVAE enables modeling non-Gaussian cluster shapes [3].

Two main projects:

Two main projects:

- Neural variational message passing (generalization of SVAE/VAE/VMP)

Two main projects:

- Neural variational message passing (generalization of SVAE/VAE/VMP)
- Model-based reinforcement learning (application of SVAE)

Questions?

References

- [1] Rajesh Ranganath, Sean Gerrish, and David M. Blei. Black Box Variational Inference. dec 2013.
- [2] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. dec 2013.
- [3] M. Johnson, D. Duvenaud, A. Wiltchko, S. Datta, and R. Adams. Composing graphical models with neural networks for structured representations and fast inference. In *NIPS*, 2016.

Reparametrization trick

What is the reparametrization trick?

Reparametrization trick

What is the reparametrization trick? We want to sample $z \sim q(z|x)$ and do so it in a roundabout way.

$$\epsilon \sim p(\epsilon)$$

$$z = f(r_\phi(x), \epsilon)$$

where ϵ is noise sampled from a simple distribution (e.g. Gaussian) and f is a function that depends on the type of distribution.

Reparametrization trick

What is the reparametrization trick? We want to sample $z \sim q(z|x)$ and do so it in a roundabout way.

$$\epsilon \sim p(\epsilon)$$

$$z = f(r_\phi(x), \epsilon)$$

where ϵ is noise sampled from a simple distribution (e.g. Gaussian) and f is a function that depends on the type of distribution.

Example: sampling from a univariate Gaussian $q_\phi(z|x)$

Reparametrization trick

What is the reparametrization trick? We want to sample $z \sim q(z|x)$ and do so it in a roundabout way.

$$\epsilon \sim p(\epsilon)$$

$$z = f(r_\phi(x), \epsilon)$$

where ϵ is noise sampled from a simple distribution (e.g. Gaussian) and f is a function that depends on the type of distribution.

Example: sampling from a univariate Gaussian $q_\phi(z|x)$

- Compute $(\mu, \sigma^2) = r(x)$

Reparametrization trick

What is the reparametrization trick? We want to sample $z \sim q(z|x)$ and do so it in a roundabout way.

$$\epsilon \sim p(\epsilon)$$

$$z = f(r_\phi(x), \epsilon)$$

where ϵ is noise sampled from a simple distribution (e.g. Gaussian) and f is a function that depends on the type of distribution.

Example: sampling from a univariate Gaussian $q_\phi(z|x)$

- Compute $(\mu, \sigma^2) = r(x)$
- Sample $\epsilon \sim N(0, 1)$

Reparametrization trick

What is the reparametrization trick? We want to sample $z \sim q(z|x)$ and do so it in a roundabout way.

$$\epsilon \sim p(\epsilon)$$

$$z = f(r_\phi(x), \epsilon)$$

where ϵ is noise sampled from a simple distribution (e.g. Gaussian) and f is a function that depends on the type of distribution.

Example: sampling from a univariate Gaussian $q_\phi(z|x)$

- Compute $(\mu, \sigma^2) = r(x)$
- Sample $\epsilon \sim N(0, 1)$
- Return $z = \mu + \epsilon\sigma$

Reparametrization trick

What is the reparametrization trick? We want to sample $z \sim q(z|x)$ and do so it in a roundabout way.

$$\epsilon \sim p(\epsilon)$$

$$z = f(r_\phi(x), \epsilon)$$

where ϵ is noise sampled from a simple distribution (e.g. Gaussian) and f is a function that depends on the type of distribution.

Example: sampling from a univariate Gaussian $q_\phi(z|x)$

- Compute $(\mu, \sigma^2) = r(x)$
- Sample $\epsilon \sim N(0, 1)$
- Return $z = \mu + \epsilon\sigma$

This is very effective at decreasing the variance of the gradient.

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

¹Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

General approach:

- Learn a latent LDS
- Use a linear controller to optimize trajectories in the latent space

¹Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

General approach:

- Learn a latent LDS
- Use a linear controller to optimize trajectories in the latent space

Benefits of this approach:

¹Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

General approach:

- Learn a latent LDS
- Use a linear controller to optimize trajectories in the latent space

Benefits of this approach:

- We can learn simple dynamics even with camera data

¹Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

General approach:

- Learn a latent LDS
- Use a linear controller to optimize trajectories 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

¹Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

General approach:

- Learn a latent LDS
- Use a linear controller to optimize trajectories 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

¹Joint work with Marvin Zhang from UC Berkeley

Current work

One idea I am currently working on is using the VAE to learn latent models to be used in reinforcement learning¹.

General approach:

- Learn a latent LDS
- Use a linear controller to optimize trajectories 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

¹Joint work with Marvin Zhang from UC Berkeley