Machine Learning Reading Group

Presenter: Daniel Iong

Paper #1: Auto-encoding Variational Bayes (Kingma et al. (2014))
Paper #2: Variational Inference with Normalizing Flows (Rezende et al. (2015))

October 1, 2021

Overview

1. Background
   - Bayesian Inference/Latent variable modeling
   - Variational Inference
2. Overview of contributions
3. Paper #1
   - Reparameterization trick
   - Stochastic Gradient VB Estimators
   - Auto-encoding VB Algorithm
   - Variational Auto-Encoder
4. Paper #2
   - Normalizing flows
   - VI algorithm
5. Comparison of papers
6. Related work

Overview

Background: Bayesian Inference/Latent Variable Modeling

Consider a latent variable model of data $\mathbf{x} = \{x_i\}$ and latent variables $\mathbf{z} = \{z_i\}$ with joint density

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}).$$

Background: Bayesian Inference/Latent Variable Modeling

Consider a latent variable model of data $\mathbf{x} = \{x_i\}$ and latent variables $\mathbf{z} = \{z_i\}$ with joint density

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}).$$

Example: Gaussian Mixture model

$$p(z_i = k) = \pi_k, \;\; k = 1, \ldots, K$$
$$x_i | z_i = k \sim N(\mu_k, \sigma_k^2)$$

Background: Bayesian Inference/Latent Variable Modeling

Consider a latent variable model of data $\mathbf{x} = \{x_i\}$ and latent variables $\mathbf{z} = \{z_i\}$ with joint density

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}).$$

Example: Gaussian Mixture model

$$p(z_i = k) = \pi_k, \ \ k = 1, \ldots, K$$
$$x_i | z_i = k \sim N(\mu_k, \sigma_k^2)$$

Objective: Compute/sample from the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$.

- $p(\mathbf{x})$ is usually difficult/impossible to compute.

Background: Bayesian Inference/Latent Variable Modeling

Consider a latent variable model of data $\mathbf{x} = \{x_i\}$ and latent variables $\mathbf{z} = \{z_i\}$ with joint density

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}).$$

Example: Gaussian Mixture model

$$p(z_i = k) = \pi_k, \ \ k = 1, \ldots, K$$
$$x_i | z_i = k \sim N(\mu_k, \sigma_k^2)$$

Objective: Compute/sample from the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$.

- $p(\mathbf{x})$ is usually difficult/impossible to compute.

Two solutions

1. MCMC: Construct an ergodic Markov chain whose stationary distribution is $p_\theta(\mathbf{z}|\mathbf{x})$.

Background: Bayesian Inference/Latent Variable Modeling

Consider a latent variable model of data $\mathbf{x} = \{x_i\}$ and latent variables $\mathbf{z} = \{z_i\}$ with joint density

$$p(\mathbf{z}, \mathbf{x}) = p(\mathbf{z})p(\mathbf{x}|\mathbf{z}).$$

Example: Gaussian Mixture model

$$p(z_i = k) = \pi_k, \;\; k = 1, \ldots, K$$
$$x_i | z_i = k \sim N(\mu_k, \sigma_k^2)$$

Objective: Compute/sample from the posterior distribution $p_\theta(\mathbf{z}|\mathbf{x}) = \frac{p(\mathbf{x}|\mathbf{z})p(\mathbf{z})}{p(\mathbf{x})}$.

- $p(\mathbf{x})$ is usually difficult/impossible to compute.

Two solutions

1. MCMC: Construct an ergodic Markov chain whose stationary distribution is $p_\theta(\mathbf{z}|\mathbf{x})$.
2. **Variational Inference (VI)**: Compute an approximation of $p_\theta(\mathbf{z}|\mathbf{x})$ by optimizing over a family of approx. distributions.

Background: Variational Inference

Objective: Obtain approximate posterior (or **recognition model**)

$$q_\phi^*(\mathbf{z}|\mathbf{x}) = \arg\min_{q \in \mathcal{Q}} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})),$$

- $\mathcal{Q}$ is a (specified) **variational family** (e.g. mean-field)
- ϕ contain the **variational parameters**; θ contain the **model parameters**.

Background: Variational Inference

Objective: Obtain approximate posterior (or **recognition model**)

$$q_\phi^*(\mathbf{z}|\mathbf{x}) = \arg\min_{q \in \mathcal{Q}} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})),$$

- $\mathcal{Q}$ is a (specified) **variational family** (e.g. mean-field)
- ϕ contain the **variational parameters**; θ contain the **model parameters**.

KL Divergence:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) := \int \log\left(\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right) q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} = E_q\left[\log \frac{q_\phi(\mathbf{Z}|\mathbf{x})}{p_\theta(\mathbf{Z}|\mathbf{x})}\right]$$

Background: Variational Inference

Objective: Obtain approximate posterior (or **recognition model**)

$$q_\phi^*(\mathbf{z}|\mathbf{x}) = \arg\min_{q \in \mathcal{Q}} D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})),$$

- $\mathcal{Q}$ is a (specified) **variational family** (e.g. mean-field)
- ϕ contain the **variational parameters**; θ contain the **model parameters**.

KL Divergence:

$$D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) := \int \log\left(\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})}\right) q_\phi(\mathbf{z}|\mathbf{x}) d\mathbf{z} = E_q\left[\log \frac{q_\phi(\mathbf{Z}|\mathbf{x})}{p_\theta(\mathbf{Z}|\mathbf{x})}\right]$$

- Asymmetric: $D_{KL}(q||p)$ (*exclusive KL*) $\neq D_{KL}(p||q)$ (*inclusive KL*).
  - VI chooses to optimize $D_{KL}(q||p)$ b/c it's easier to take expectations w.r.t. q.
  - There's existing work about optimizing $D_{KL}(p||q)$ (Markovian score climbing).
- Exclusive KL: $q(x) > 0 \Rightarrow p(x) > 0$

Background: Variational Inference

- The marginal likelihood (**evidence**) $p(\mathbf{x})$ can be expressed as

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \underbrace{E_q\big[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\theta(\mathbf{x},\mathbf{z})\big]}_{\mathcal{L}(\theta,\phi;\mathbf{x})} \geq \mathcal{L}(\theta,\phi;\mathbf{x}),$$

where $\mathcal{L}(\theta,\phi;\mathbf{x})$ is the **evidence lower bound** (ELBO) which can also be written as

$$\mathcal{L}(\theta,\phi;\mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + E_q\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big].$$

Background: Variational Inference

- The marginal likelihood (**evidence**) $p(\mathbf{x})$ can be expressed as

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \underbrace{E_q\big[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\theta(\mathbf{x},\mathbf{z})\big]}_{\mathcal{L}(\theta,\phi;\mathbf{x})} \geq \mathcal{L}(\theta,\phi;\mathbf{x}),$$

  where $\mathcal{L}(\theta,\phi;\mathbf{x})$ is the **evidence lower bound** (ELBO) which can also be written as

$$\mathcal{L}(\theta,\phi;\mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + E_q\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big].$$

- **Goal**: Optimize $\mathcal{L}(\theta,\phi;\mathbf{x})$ w.r.t. both ϕ and θ.

Background: Variational Inference

- The marginal likelihood (**evidence**) $p(\mathbf{x})$ can be expressed as

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \underbrace{E_q\big[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\theta(\mathbf{x},\mathbf{z})\big]}_{\mathcal{L}(\theta,\phi;\mathbf{x})} \geq \mathcal{L}(\theta,\phi;\mathbf{x}),$$

  where $\mathcal{L}(\theta,\phi;\mathbf{x})$ is the **evidence lower bound** (ELBO) which can also be written as

$$\mathcal{L}(\theta,\phi;\mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + E_q\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big].$$

- **Goal**: Optimize $\mathcal{L}(\theta,\phi;\mathbf{x})$ w.r.t. both ϕ and θ.
  - Optimizing w.r.t. θ is easy
    - Unbiased gradient estimates are straightforward to obtain.

Background: Variational Inference

- The marginal likelihood (**evidence**) $p(\mathbf{x})$ can be expressed as

$$\log p_\theta(\mathbf{x}) = D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}|\mathbf{x})) + \underbrace{E_q\big[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p_\theta(\mathbf{x},\mathbf{z})\big]}_{\mathcal{L}(\theta,\phi;\mathbf{x})} \geq \mathcal{L}(\theta,\phi;\mathbf{x}),$$

  where $\mathcal{L}(\theta,\phi;\mathbf{x})$ is the **evidence lower bound** (ELBO) which can also be written as

$$\mathcal{L}(\theta,\phi;\mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + E_q\big[\log p_\theta(\mathbf{x}|\mathbf{z})\big].$$

- **Goal**: Optimize $\mathcal{L}(\theta,\phi;\mathbf{x})$ w.r.t. both ϕ and θ.
  - Optimizing w.r.t. θ is easy
    - Unbiased gradient estimates are straightforward to obtain.
  - Optimizing w.r.t. ϕ is harder.
    - Expectations in the ELBO are taken w.r.t. $q_\phi(\mathbf{z}|\mathbf{x})$, which is a function of ϕ.
    - Reparameterization trick is useful here.

Overview

Overview of contributions

Paper #1: Auto-encoding Variational Bayes (*Kingma et al. 2014*)

- Two significant contributions
  1. SGVB estimator + AEVB algorithm
  2. Variational Auto-encoders (VAEs)

Overview of contributions

Paper #1: Auto-encoding Variational Bayes (*Kingma et al. 2014*)

- Two significant contributions
  1. SGVB estimator + AEVB algorithm
  2. Variational Auto-encoders (VAEs)

Paper #2: Variational Inference with Normalizing Flows (*Rezende et al. 2015*)

- Provides a rich, computationally-feasible approximate posterior using normalizing flows.
  - In the asymptotic regime, the space of solutions is rich enough to contain the true posterior distribution.
- Uses gradient estimator from Paper #1 (amortized VI).

Overview of contributions

Paper #1: Auto-encoding Variational Bayes (*Kingma et al. 2014*)

- Two significant contributions
  1. SGVB estimator + AEVB algorithm
  2. Variational Auto-encoders (VAEs)

Paper #2: Variational Inference with Normalizing Flows (*Rezende et al. 2015*)

- Provides a rich, computationally-feasible approximate posterior using normalizing flows.
  - In the asymptotic regime, the space of solutions is rich enough to contain the true posterior distribution.
- Uses gradient estimator from Paper #1 (amortized VI).

Topics I will not cover:

- Experiments/empirical results
- infinitesimal flows

Overview

Paper #1: Reparameterization Trick

Suppose $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. Then, it is often possible to express $\mathbf{z}$ as

$$\mathbf{z} = g_\phi(\epsilon, \mathbf{x}),$$

where $\epsilon \sim p(\epsilon)$ (a distribution that doesn't depend on ϕ).

Example: Suppose $z \sim N(\mu, \sigma^2)$. Then $z = \mu + \sigma\epsilon$, where $\epsilon \sim N(0, 1)$.

Paper #1: Reparameterization Trick

Suppose $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. Then, it is often possible to express $\mathbf{z}$ as

$$\mathbf{z} = g_\phi(\epsilon, \mathbf{x}),$$

where $\epsilon \sim p(\epsilon)$ (a distribution that doesn't depend on ϕ).

Example: Suppose $z \sim N(\mu, \sigma^2)$. Then $z = \mu + \sigma\epsilon$, where $\epsilon \sim N(0, 1)$.

Why is it useful?

- In general, for a differentiable function f,

$$\nabla_\phi E_{q_\phi}[f(\mathbf{z})] \neq E_{q_\phi}[\nabla_\phi f(\mathbf{z})]$$

Paper #1: Reparameterization Trick

Suppose $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$. Then, it is often possible to express $\mathbf{z}$ as

$$\mathbf{z} = g_\phi(\epsilon, \mathbf{x}),$$

where $\epsilon \sim p(\epsilon)$ (a distribution that doesn't depend on ϕ).

Example: Suppose $z \sim N(\mu, \sigma^2)$. Then $z = \mu + \sigma\epsilon$, where $\epsilon \sim N(0, 1)$.

Why is it useful?

- In general, for a differentiable function f,

$$\nabla_\phi E_{q_\phi}[f(\mathbf{z})] \neq E_{q_\phi}[\nabla_\phi f(\mathbf{z})]$$

- However, with the reparameterization trick,

$$\begin{aligned}\nabla_\phi E_{q_\phi}[f(\mathbf{z})] &= \nabla_\phi E_{p(\epsilon)}[f(g_\phi(\epsilon, \mathbf{x}))] \\ &= E_{p(\epsilon)}[\nabla_\phi f(g_\phi(\epsilon, \mathbf{x}))] \\ &\approx \frac{1}{L}\sum_{l=1}^{L} f(g_\phi(\epsilon^{(l)}, \mathbf{x})), \text{ where } \epsilon^{(l)} \sim p(\epsilon)\end{aligned}$$

Paper #1: Reparameterization Trick

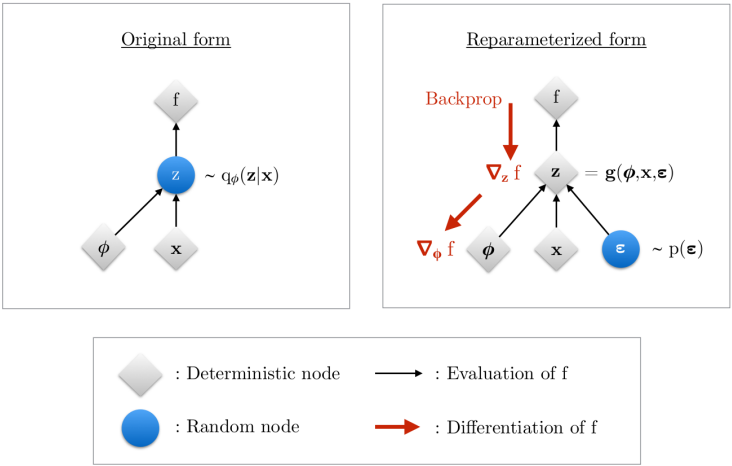

Illustration of reparameterization trick from Kingma (2017).

Paper #1: Stochastic Gradient Variational Bayes Estimators

Using the reparameterization trick, a generic SGVB estimator of the ELBO is given by

$$\tilde{\mathcal{L}}^A(\theta, \phi; \mathbf{x}) = \frac{1}{L} \sum_{l=1}^{L} \Big[\log p_\theta(\mathbf{x}, \mathbf{z}^{(l)}) - \log q_\phi(\mathbf{z}^{(l)}|\mathbf{x}) \Big],$$

where $\mathbf{z}^{(l)} = g_\phi(\epsilon^{(l)}, \mathbf{x})$ and $\epsilon^{(l)} \sim p(\epsilon)$, for $l = 1, \ldots, L$.

Paper #1: Stochastic Gradient Variational Bayes Estimators

Using the reparameterization trick, a generic SGVB estimator of the ELBO is given by

$$\tilde{\mathcal{L}}^A(\theta, \phi; \mathbf{x}) = \frac{1}{L} \sum_{l=1}^{L} \Big[\log p_\theta(\mathbf{x}, \mathbf{z}^{(l)}) - \log q_\phi(\mathbf{z}^{(l)}|\mathbf{x}) \Big],$$

where $\mathbf{z}^{(l)} = g_\phi(\epsilon^{(l)}, \mathbf{x})$ and $\epsilon^{(l)} \sim p(\epsilon)$, for $l = 1, \ldots, L$.

When $D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))$ can be computed analytically (e.g. Gaussian case),

$$\tilde{\mathcal{L}}^B(\theta, \phi; \mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z})) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}^{(l)})$$

- $\tilde{\mathcal{L}}^B$ typically has less variance than $\tilde{\mathcal{L}}^A$.

Paper #1: Auto-Encoding Variational Bayes Algorithm

Algorithm 1 Minibatch version of the Auto-Encoding VB (AEVB) algorithm. Either of the two SGVB estimators in section 2.3 can be used. We use settings $M = 100$ and $L = 1$ in experiments.

$\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Initialize parameters
repeat
  $\mathbf{X}^M \leftarrow$ Random minibatch of M datapoints (drawn from full dataset)
  $\boldsymbol{\epsilon} \leftarrow$ Random samples from noise distribution $p(\boldsymbol{\epsilon})$
  $\mathbf{g} \leftarrow \nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}^M(\boldsymbol{\theta}, \boldsymbol{\phi}; \mathbf{X}^M, \boldsymbol{\epsilon})$ (Gradients of minibatch estimator (8))
  $\boldsymbol{\theta}, \boldsymbol{\phi} \leftarrow$ Update parameters using gradients $\mathbf{g}$ (e.g. SGD or Adagrad [DHS10])
until convergence of parameters $(\boldsymbol{\theta}, \boldsymbol{\phi})$
return $\boldsymbol{\theta}, \boldsymbol{\phi}$

The mini-batch estimator $\tilde{\mathcal{L}}^M$ is given by

$$\tilde{\mathcal{L}}^M(\theta, \phi; X^M) = \frac{N}{M} \sum_{i=1}^{M} \tilde{\mathcal{L}}(\theta, \phi; \mathbf{x})$$

Paper #1: Auto-Encoding Variational Bayes Algorithm

How is it related to auto-encoders?

An **autoencoder** is a neural network used for unsupervised learning that minimizes an objective function with the form: reconstruction error + regularizer.

Recall:

$$\tilde{\mathcal{L}}^B(\theta, \phi; \mathbf{x}) = \underbrace{-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_\theta(\mathbf{z}))}_{\text{regularizer}} + \underbrace{\frac{1}{L}\sum_{l=1}^{L} \log p_\theta(\mathbf{x}|\mathbf{z}^{(l)})}_{\text{Negative reconstruction error}}$$

Paper #1: Variational Auto-Encoder

VAEs are given as an example but AFAIK, this is the original paper that introduced them.

Assume $\mathbf{x}$, $\mathbf{z}$ are continuous.

Decoder (generative model)

$$
\begin{aligned}
p(\mathbf{z}) &= N(\mathbf{z}; 0, I) \\
p(\mathbf{x}|\mathbf{z}) &= N(\mathbf{x}; \mu, \sigma^2 I) \\
\mu &= W_2 h + b_2 \\
\log \sigma^2 &= W_3 h + b_3 \\
h &= \tanh(W_1 \mathbf{z} + b_1)
\end{aligned}
$$

Encoder (inference model)

$$
\begin{aligned}
q_\phi(\mathbf{z}|\mathbf{x}) &= N(\mathbf{z}; \tilde{\mu}, \tilde{\sigma}^2 I) \\
\tilde{h} &= \tanh(\tilde{W}_1 \mathbf{x} + \tilde{b}_1) \\
\tilde{\mu} &= \tilde{W}_1 \tilde{h} + \tilde{b}_2 \\
\log \tilde{\sigma}^2 &= \tilde{W}_3 \tilde{h} + \tilde{b}_3
\end{aligned}
$$

$\theta = \{W_j, b_j : j = 1, 2, 3\}, \phi = \{\tilde{W}_j, \tilde{b}_j : j = 1, 2, 3\}$

Paper #1: Variational Auto-Encoder

VAEs are given as an example but AFAIK, this is the original paper that introduced them.

Assume $\mathbf{x}$, $\mathbf{z}$ are continuous.

Decoder (generative model)

$$
\begin{aligned}
p(\mathbf{z}) &= N(\mathbf{z}; 0, I) \\
p(\mathbf{x}|\mathbf{z}) &= N(\mathbf{x}; \mu, \sigma^2 I) \\
\mu &= W_2 h + b_2 \\
\log \sigma^2 &= W_3 h + b_3 \\
h &= \tanh(W_1 \mathbf{z} + b_1)
\end{aligned}
$$

Encoder (inference model)

$$
\begin{aligned}
q_\phi(\mathbf{z}|\mathbf{x}) &= N(\mathbf{z}; \tilde{\mu}, \tilde{\sigma}^2 I) \\
\tilde{h} &= \tanh(\tilde{W}_1 \mathbf{x} + \tilde{b}_1) \\
\tilde{\mu} &= \tilde{W}_1 \tilde{h} + \tilde{b}_2 \\
\log \tilde{\sigma}^2 &= \tilde{W}_3 \tilde{h} + \tilde{b}_3
\end{aligned}
$$

$\theta = \{W_j, b_j : j = 1, 2, 3\}, \phi = \{\tilde{W}_j, \tilde{b}_j : j = 1, 2, 3\}$

- VAEs are a non-linear generalization of probabilistic PCA, where $\mu = \mathbf{Wz}$.
  - In this case, the evidence has an analytical form: $p(\mathbf{x}) = N(0, \mathbf{WW}' + \sigma^2 I)$.

Paper #1: Variational Auto-Encoder

VAEs are given as an example but AFAIK, this is the original paper that introduced them.

Assume $\mathbf{x}$, $\mathbf{z}$ are continuous.

Decoder (generative model)

$$
\begin{aligned}
p(\mathbf{z}) &= N(\mathbf{z}; 0, I) \\
p(\mathbf{x}|\mathbf{z}) &= N(\mathbf{x}; \mu, \sigma^2 I) \\
\mu &= W_2 h + b_2 \\
\log \sigma^2 &= W_3 h + b_3 \\
h &= \tanh(W_1 \mathbf{z} + b_1)
\end{aligned}
$$

Encoder (inference model)

$$
\begin{aligned}
q_\phi(\mathbf{z}|\mathbf{x}) &= N(\mathbf{z}; \tilde{\mu}, \tilde{\sigma}^2 I) \\
\tilde{h} &= \tanh(\tilde{W}_1 \mathbf{x} + \tilde{b}_1) \\
\tilde{\mu} &= \tilde{W}_1 \tilde{h} + \tilde{b}_2 \\
\log \tilde{\sigma}^2 &= \tilde{W}_3 \tilde{h} + \tilde{b}_3
\end{aligned}
$$

$\theta = \{W_j, b_j : j = 1, 2, 3\}, \phi = \{\tilde{W}_j, \tilde{b}_j : j = 1, 2, 3\}$

- VAEs are a non-linear generalization of probabilistic PCA, where $\mu = \mathbf{W}\mathbf{z}$.
  - In this case, the evidence has an analytical form: $p(\mathbf{x}) = N(0, \mathbf{W}\mathbf{W}' + \sigma^2 I)$.

To sample from $q_\phi(\mathbf{z}|\mathbf{x})$,

1. Sample $\epsilon^{(\ell)} \sim N(0, I)$, $\ell = 1, \ldots, L$
2. Compute $\mathbf{z}^{(\ell)} = \tilde{\mu} + \tilde{\sigma} \odot \epsilon^{(\ell)}$, where $\odot$ denotes element-wise product.

Paper #1: Variational Auto-Encoder

In the case where $p(\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are Gaussian, $-D_{KL}(q_\phi(\mathbf{z}||\mathbf{x})||p(\mathbf{z}))$ can be computed analytically.

$$-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left[1 + \log(\tilde{\sigma}_j^2) - \tilde{\mu}_j^2 - \tilde{\sigma}_j^2\right]$$

Paper #1: Variational Auto-Encoder

In the case where $p(\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are Gaussian, $-D_{KL}(q_\phi(\mathbf{z}||\mathbf{x})||p(\mathbf{z}))$ can be computed analytically.

$$-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left[1 + \log(\tilde{\sigma}_j^2) - \tilde{\mu}_j^2 - \tilde{\sigma}_j^2\right]$$

The SGVB estimator for VAEs is given by,

$$\tilde{\mathcal{L}}^B(\theta, \phi; \mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + \frac{1}{L}\sum_{\ell=1}^{L}\log p_\theta(\mathbf{x}|\mathbf{z}^{(\ell)}) \tag{1}$$

where $\mathbf{z}^{(\ell)} = \tilde{\mu} + \tilde{\sigma} \odot \epsilon^{(\ell)}$ and $\epsilon^{(\ell)} \sim N(0, I)$.

Paper #1: Variational Auto-Encoder

In the case where $p(\mathbf{z})$ and $q_\phi(\mathbf{z}|\mathbf{x})$ are Gaussian, $-D_{KL}(q_\phi(\mathbf{z}||\mathbf{x})||p(\mathbf{z}))$ can be computed analytically.

$$-D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) = \frac{1}{2}\sum_{j=1}^{J}\left[1 + \log(\tilde{\sigma}_j^2) - \tilde{\mu}_j^2 - \tilde{\sigma}_j^2\right]$$

The SGVB estimator for VAEs is given by,

$$\tilde{\mathcal{L}}^B(\theta, \phi; \mathbf{x}) = -D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p(\mathbf{z})) + \frac{1}{L}\sum_{\ell=1}^{L}\log p_\theta(\mathbf{x}|\mathbf{z}^{(\ell)}) \tag{1}$$

where $\mathbf{z}^{(\ell)} = \tilde{\mu} + \tilde{\sigma} \odot \epsilon^{(\ell)}$ and $\epsilon^{(\ell)} \sim N(0, I)$.

- We can optimize (1) using the AEVB algorithm to obtain estimates of θ and ϕ.

Overview

Paper #2: Normalizing flows

- In paper #1, a Gaussian distribution with diagonal covariance was used as the approximate posterior.
- How can we obtain a more flexible family of variational distributions?
  - **Solution**: Normalizing flows
- **Normalizing flows** transform simple distributions (e.g. Gaussian) through a sequence of invertible mappings into rich complex distributions.

Paper #2: Normalizing flows

- In paper #1, a Gaussian distribution with diagonal covariance was used as the approximate posterior.
- How can we obtain a more flexible family of variational distributions?
  - **Solution**: Normalizing flows
- **Normalizing flows** transform simple distributions (e.g. Gaussian) through a sequence of invertible mappings into rich complex distributions.

Review: Change of variables

Let $f : \mathbb{R}^d \to \mathbb{R}^d$ be an invertible, smooth function with inverse f^{-1} and $\mathbf{z} \sim q(\mathbf{z})$.
Then $\mathbf{z}' = f(\mathbf{z})$ has the distribution:

$$q(\mathbf{z}') = q(\mathbf{z}) \left| \det \frac{\partial f^{-1}}{\partial \mathbf{z}'} \right| = q(\mathbf{z}) \left| \det \frac{\partial f}{\partial \mathbf{z}} \right|^{-1}$$

Paper #2: Normalizing flows

Suppose $\mathbf{z}_0 \sim q_0(\mathbf{z}_0)$ and $\mathbf{z}_K = f_K \circ \dots f_2 \circ f_1(\mathbf{z}_0)$, where $f_1, \dots, f_K$ is a sequence of mappings. Then the log density of $\mathbf{z}_K$ is given by

$$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$$

Paper #2: Normalizing flows

Suppose $\mathbf{z}_0 \sim q_0(\mathbf{z}_0)$ and $\mathbf{z}_K = f_K \circ \dots f_2 \circ f_1(\mathbf{z}_0)$, where $f_1, \dots, f_K$ is a sequence of mappings. Then the log density of $\mathbf{z}_K$ is given by

$$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}_0) - \sum_{k=1}^{K} \log \det \left| \frac{\partial f_k}{\partial \mathbf{z}_k} \right|$$

By the law of the unconscious statistician, for any function h,

$$E_{q_K}\big[h(\mathbf{z})\big] = E_{q_0}\big[h(f_K \circ \dots \circ f_1(\mathbf{z}_0))\big]$$

- There are many good choices for the sequence of invertible transformations. Paper #2 uses planar and radial flows.

Paper #2: Normalizing flows

Planar flows are transformations of the form

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}'\mathbf{z} + b),$$

where $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$ are free parameters and h is a smooth function with derivative h'.

Paper #2: Normalizing flows

Planar flows are transformations of the form

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}'\mathbf{z} + b),$$

where $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$ are free parameters and h is a smooth function with derivative h'.

- The logdet-Jacobian term can computed in $O(D)$ time:

$$\det\left|\frac{\partial f}{\partial \mathbf{z}}\right| = \left|\det(\mathbf{I} + \mathbf{u}\psi(\mathbf{z})')\right| = |1 + \mathbf{u}'\psi(\mathbf{z})|,$$
$$\psi(\mathbf{z}) = h'(\mathbf{w}'\mathbf{z} + b)\mathbf{w}.$$

Paper #2: Normalizing flows

Planar flows are transformations of the form

$$f(\mathbf{z}) = \mathbf{z} + \mathbf{u}h(\mathbf{w}'\mathbf{z} + b),$$

where $\lambda = \{\mathbf{w} \in \mathbb{R}^D, \mathbf{u} \in \mathbb{R}^D, b \in \mathbb{R}\}$ are free parameters and h is a smooth function with derivative h'.

- The logdet-Jacobian term can computed in $O(D)$ time:

$$\det\left|\frac{\partial f}{\partial \mathbf{z}}\right| = \left|\det(\mathbf{I} + \mathbf{u}\psi(\mathbf{z})')\right| = |1 + \mathbf{u}'\psi(\mathbf{z})|,$$
$$\psi(\mathbf{z}) = h'(\mathbf{w}'\mathbf{z} + b)\mathbf{w}.$$

- The log-density of $z_K = f_K \circ \cdots \circ f_1(\mathbf{z})$, where $f_k = \mathbf{z} + \mathbf{u}_k h(\mathbf{w}_k'\mathbf{z} + b_k)$:

$$\log q_K(\mathbf{z}_K) = \log q_0(\mathbf{z}) - \sum_{k=1}^{K} \log|1 + \mathbf{u}_k'\psi_k(\mathbf{z}_k)|,$$

Paper #2: Normalizing flows

Radial flows are transformations of the form:

$$f(\mathbf{z}) = \mathbf{z} + \beta h(\alpha, r)(\mathbf{z} - \mathbf{z}_0),$$

where $r = ||\mathbf{z} - \mathbf{z}_0||$; $h(\alpha, r) = 1/(\alpha + r)$; and $\lambda = \{\mathbf{z}_0 \in \mathbb{R}^D, \alpha \in \mathbb{R}, \beta \in \mathbb{R}\}$ are parameters.

- The logdet-Jacobian can also be computed in linear time:

$$\det\left|\frac{\partial f}{\partial \mathbf{z}}\right| = \left[1 + \beta h(\alpha, r)\right]^{d-1}\left[1 + \beta h(\alpha, r) + h'(\alpha, r)r\right]$$

Paper #2: Normalizing flows

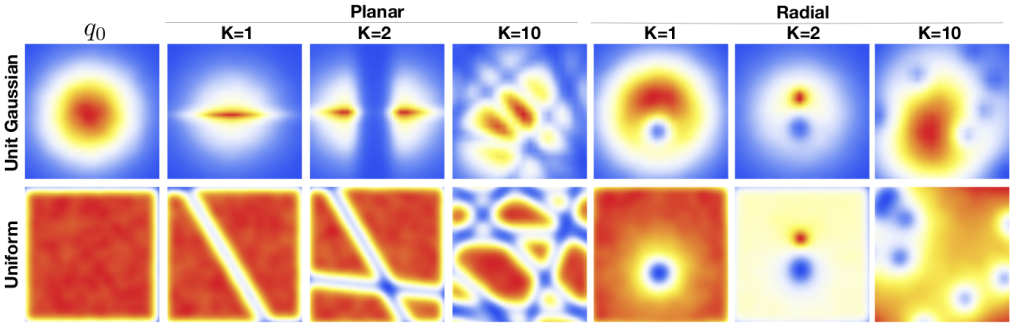

Figure 1 in Rezende et al. (2015) showing effects planar/radial flows on two distributions.

Paper #2: Normalizing flows

Flow-based ELBO (negative free energy bound)

Suppose the approximate posterior is parameterized with a (planar) flow of length K i.e. $q_\phi(\mathbf{z}|\mathbf{x}) := q_K(\mathbf{z}_K)$.

Then the ELBO is given by

$$\begin{aligned}
\mathcal{L}(\theta, \phi; \mathbf{x}) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}\Big[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{x}, \mathbf{z}) \Big] \\
&= E_{q_0(z_0)}\Big[-\log q_K(\mathbf{z}_K) + \log p(\mathbf{x}, \mathbf{z}_K) \Big] \\
&= -E_{q_0(z_0)}\big[\log q_0(z_0) \big] + E_{q_0(z_0)}\big[\log p(\mathbf{x}, \mathbf{z}_K) \big] + E_{q_0(z_0)}\Big[\sum_{k=1}^{K} \log |1 + \mathbf{u}_k' \psi_k(\mathbf{z}_k) \Big]
\end{aligned}$$

Paper #2: Normalizing flows

Flow-based ELBO (negative free energy bound)

Suppose the approximate posterior is parameterized with a (planar) flow of length K i.e. $q_\phi(\mathbf{z}|\mathbf{x}) := q_K(\mathbf{z}_K)$.

Then the ELBO is given by

$$\begin{aligned}
\mathcal{L}(\theta, \phi; \mathbf{x}) &= E_{q_\phi(\mathbf{z}|\mathbf{x})}\Big[-\log q_\phi(\mathbf{z}|\mathbf{x}) + \log p(\mathbf{x}, \mathbf{z}) \Big] \\
&= E_{q_0(z_0)}\Big[-\log q_K(\mathbf{z}_K) + \log p(\mathbf{x}, \mathbf{z}_K) \Big] \\
&= -E_{q_0(z_0)}\big[\log q_0(z_0) \big] + E_{q_0(z_0)}\big[\log p(\mathbf{x}, \mathbf{z}_K) \big] + E_{q_0(z_0)}\Big[\sum_{k=1}^{K} \log |1 + \mathbf{u}_k' \psi_k(\mathbf{z}_k) \Big]
\end{aligned}$$

- Paper #2 also constructs a recognition model (inference network) using a deep neural network that maps observations x to the parameters of $q_0 = N(\mu, \sigma^2)$ and flow parameters λ.
  - They don't give explicit details about how they do this like paper #1 does.
  - It could be interesting to work out the explicit form of the ELBO here (*if time permits*).

Paper #2: VI algorithm

Algorithm 1 Variational Inf. with Normalizing Flows

Parameters: ϕ variational, $\boldsymbol{\theta}$ generative
while not converged **do**
  $\mathbf{x} \leftarrow$ {Get mini-batch}
  $\mathbf{z}_0 \sim q_0(\bullet|\mathbf{x})$
  $\mathbf{z}_K \leftarrow f_K \circ f_{K-1} \circ \ldots \circ f_1(\mathbf{z}_0)$
  $\mathcal{F}(\mathbf{x}) \approx \mathcal{F}(\mathbf{x}, \mathbf{z}_K)$
  $\Delta\boldsymbol{\theta} \propto -\nabla_\theta \mathcal{F}(\mathbf{x})$
  $\Delta\boldsymbol{\phi} \propto -\nabla_\phi \mathcal{F}(\mathbf{x})$
end while

- Similiar in nature to the AEVB algorithm
- The authors don't make this explicit in the paper MC estimates are still necessary (I think?)

Overview

Comparison of both papers

- Both papers focus on different problems in VI.
  - Paper #1 focuses on estimating the ELBO and its gradient.
  - Paper #2 focuses on on coming up with a flexible variational family.
- In paper #2, the initial density is $q_0 = N(\mu, \sigma^2)$, where μ and σ are parameterized with neural networks. This is similiar to paper #1 but they use this as the approximate posterior, whereas this is transformed through normalizing flows in paper #2.
- Paper #1 gives a specific example where AEVB is applied (VAEs) whereas paper #2 talks about their algorithm more generally.
- Your thoughts?

Overview

Related work

- Wake-sleep algorithm (Hinton et al. (1995))
  - Applicable to same general class of continuous latent variable models as AEVB and also discrete latent variables.
  - Drawback: concurrent optimization of two objective functions $\not\Rightarrow$ optimization of (a bound of) the evidence.
- Non-linear Independent Components Estimation (NICE) (Dinh et al. (2014))
  - Transformations are neural networks with easy to compute inverses
- Hamiltonian variational approx. (HVI) (Salimans et al. (2015))
  - infinitesimal volume-preserving flow
  - Elegant but not as computationally efficient