10-708 PGM (Spring 2020): Homework 4

Andrew ID: changshi
Name: Chang Shi
Collaborators:

1 Variational Autoencoders (Yiwen) [65 pts]

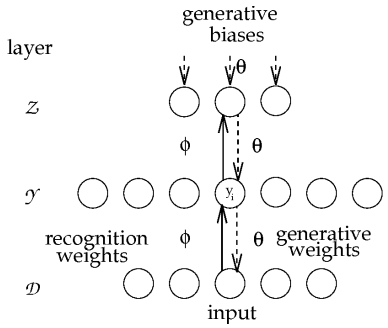


Figure 1: A **Helmholtz machine** [3, 1] contains two networks: (1) bottom-up "recognition" connections ϕ that convert the input data into representations in successive hidden layers, and (2) top-down "generative" connections θ that reconstruct the data from representation in one layer from the representation in the layer above.

The Helmholtz machine (Figure 1) is an architecture that can find hidden structure in data by learning a generative model of the data. Helmholtz machines are usually trained using unsupervised learning algorithms such as the classical **Wake-Sleep** algorithm [3] or the modern **Auto-Encoding Variational Bayes (AEVB)** [4], also known as variational autoencoder.

In this problem, you will (re-)derive and implement the Wake-Sleep and AEVB algorithms. The sections are organized as follows:

(3 pts) Section 1.1: Derivation of the **evidence lower bound objective (ELBO)**, which lowerbounds the data log-likelihood $\log p_\theta(\mathbf{x})$.

(6 pts) Section 1.2: Derivation of the **Wake-Sleep** algorithm, which alternates between the Wake phase and Sleep phase to optimize an estimate of ELBO.

(10 pts) Section 1.3: Derivation of **AEVB**, which optimizes a stochastic estimate of ELBO.

(3 pts) Section 1.4: Short-answer questions on Wake-Sleep and AEVB.

(8 pts) Section 1.5: Derivation of an **alternate lower bound** $\mathcal{L}_k(\mathbf{x})$ for the data log-likelihood, which will be used to evaluate trained models in the next section.

(35 pts) Section 1.6: Implementations and experiments on the MNIST handwritten digits dataset.

For all parts, assume that latent variables $\mathbf{z}$ are distributed according to a prior $p(\mathbf{z}) = N(0, I)$. The Helmholtz machine tries to learn the **recognition** parameters ϕ and **generative** parameters θ such that

$$q_\phi(\mathbf{z} \mid \mathbf{x}) \approx p_\theta(\mathbf{z} \mid x) \propto p_\theta(\mathbf{x}, \mathbf{z})$$

where:

- $q_\phi(\mathbf{z} \mid \mathbf{x})$ is the variational distribution approximating the posterior distribution $p_\theta(\mathbf{z} \mid \mathbf{x})$ for $\mathbf{z}$ given the evidence $\mathbf{x}$. Assume that q_ϕ is parameterized by a Gaussian, i.e., $q_\phi(\mathbf{z} \mid \mathbf{x}) = N(\mathbf{z}; \mu_\phi(\mathbf{x}), \Sigma_\phi(\mathbf{x}))$.
- $p_\theta(\mathbf{x}, \mathbf{z}) = p(\mathbf{z})p_\theta(\mathbf{x} \mid \mathbf{z})$ is the joint probability of $(\mathbf{x}, \mathbf{z})$, where $\mathbf{z} \sim p(\mathbf{z}) = N(0, I)$, and $\mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{z})$ is the likelihood.

Assume $\mathbf{x}$ are binary vectors. In other words, $p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})$ can be modeled with a sigmoid belief net, so the likelihood is of the form $p_\theta(\mathbf{x}|\mathbf{z}) = \text{Bernoulli}(f_\theta(\mathbf{z}))$. Actually, the data points $\mathbf{x}$ in MNIST take values in $[0, 1]$ rather than $\{0, 1\}$, but the loss term $\mathbb{E}_q[p_\theta(\mathbf{x} \mid \mathbf{z})]$ still uses sigmoid cross-entropy loss, which is a common practice [2].

1.1 Evidence Lower Bound Objective (ELBO)

Suppose we want to learn a directed latent variable model (Figure 2) that is able represent a complex distribution $p(\mathbf{x})$ over the data in the following form:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x} \mid \mathbf{z})p(\mathbf{z})d\mathbf{z} \tag{1}$$

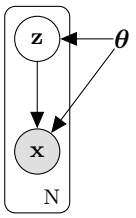


Figure 2: The latent variable model in Problem 1.1.

Suppose we want to approximate the posterior distribution $p_{\boldsymbol{\theta}}(\mathbf{z} \mid \mathbf{x})$ using some variational distribution $q_{\boldsymbol{\phi}}(\mathbf{z} \mid \mathbf{x})$. A tractable way to learn this model is to optimize the **evidence lower bound objective (ELBO)**, also known as the variational lower bound, defined as follows:

$$\begin{aligned}\mathcal{L}(\mathbf{x}) &= \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})}[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x})] \\ &= \int_z q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{p_\theta(\mathbf{x}, \mathbf{z})}{q_\phi(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} \ .\end{aligned}$$

(3 pts) For a single data point $\mathbf{x}^{(i)}$, prove that

$$\log p_\theta(\mathbf{x}^{(i)}) \geq \mathcal{L}(\mathbf{x}^{(i)}).$$

Solution

$$
\begin{aligned}
\log p_\theta(\mathbf{x}^{(i)}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log\left(\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x}^{(i)})}\frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})p_\theta(\mathbf{x}^{(i)})}{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}\right)\right] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log\left(\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}\mid\mathbf{x}^{(i)})}\right)\right] + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log\left(\frac{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}{p_\theta(\mathbf{z}|\mathbf{x}^{(i)})}\right)\right] \\
&= \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\log\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}d\mathbf{z} + D_{\mathrm{KL}}\left[q_\phi(\mathbf{z}\mid\mathbf{x}^{(i)})\parallel p_\theta(\mathbf{z}\mid\mathbf{x}^{(i)})\right] \\
&\geq \int_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}^{(i)})\log\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}d\mathbf{z} \quad \text{(KL divergence is always nonnegative)} \\
&:= \mathcal{L}(\mathbf{x}^{(i)};\theta,\phi)
\end{aligned}
$$

Or we can prove by Jensen's inequality:

$$
\log p_\theta(\mathbf{x}^{(i)}) = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\left[\log\frac{p_\theta(\mathbf{x}^{(i)},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\right] = \mathcal{L}(\mathbf{x}^{(i)})
$$

The above result shows that, for iid data points $\mathbf{x} = \{\mathbf{x}^{(i)}\}_{i=1}^N$,

$$
\log p_\theta(\mathbf{x}) \overset{\text{iid}}{=} \sum_{i=1}^N \log p_\theta(\mathbf{x}^{(i)}) \geq \sum_{i=1}^N \mathcal{L}(\mathbf{x}^{(i)}) = \mathcal{L}(\mathbf{x})
$$

which gives the ELBO $\mathcal{L}(\mathbf{x})$ on the data log-likelihood $\log p_\theta(\mathbf{x})$.

1.2 Wake-Sleep Algorithm

In this section, we will derive the optimization objectives for the Wake-Sleep algorithm, which decomposes the optimization procedure into two phases:

- **Wake**-phase: Given recognition weights ϕ, we activate the recognition process and update the generative weights θ to increase the probability that they would reconstruct the correct activity vector in the layer below.
- **Sleep**-phase: Given generative weights θ, we activate the generative process and update the recognition weights ϕ to increase the probability that they would produce the correct activity vector in the layer above. Since it has generated the instance, it knows the true underlying causes, and therefore has available the target values for the hidden units that are required to train the bottom-up weights ϕ.

1.2.1 Wake-phase

The Wake-phase fixes the recognition weights ϕ and optimizes a Monte Carlo estimate of ELBO w.r.t. the generative weights θ.

(3 pts) Given N iid data points $\mathbf{x} = \{\mathbf{x}^{(i)}\}_{i=1}^N$, show that

$$
\theta^* := \arg\max_\theta \mathcal{L}(\mathbf{x}) = \arg\max_\theta \sum_{i=1}^N \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}\log p_\theta(\mathbf{x}^{(i)}\mid\mathbf{z}) \tag{2}
$$

which gives the Wake-phase objective.

Solution

$$
\begin{aligned}
\theta^* &:= \arg\max_\theta \mathcal{L}(\mathbf{x}) \\
&= \arg\max_\theta \sum_{i=1}^N \mathcal{L}(\mathbf{x}^{(i)}) \\
&= \arg\max_\theta \sum_{i=1}^N \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})] \\
&= \arg\max_\theta \sum_{i=1}^N \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}) p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})] \\
&= \arg\max_\theta \sum_{i=1}^N \left(\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}) + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})] \right) \\
&= \arg\max_\theta \sum_{i=1}^N \left(\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}) + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})] \right) \\
&= \arg\max_\theta \sum_{i=1}^N \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})} \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z})
\end{aligned}
$$

Wake-phase Pseudocode: Given N iid data points $\{\mathbf{x}^{(i)}\}_{i=1}^N$, do the following for each $i \in [N]$:

1. Feed $\mathbf{x}^{(i)}$ into the recognition network to get $\mu_\phi(\mathbf{x}^{(i)})$ and $\Sigma_\phi(\mathbf{x}^{(i)})$.
2. Draw L samples $\mathbf{z}_1^{(i)}, \ldots, \mathbf{z}_L^{(i)} \sim q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) = N(\mathbf{z}; \mu_\phi(\mathbf{x}^{(i)}), \Sigma_\phi(\mathbf{x}^{(i)}))$.
3. For each $l \in [L]$, feed $\mathbf{z}_l^{(i)}$ into the generative network to get $f_\theta(\mathbf{z}_l^{(i)})$ for the likelihood $p_\theta(\mathbf{x} \mid \mathbf{z}_l^{(i)}) = \text{Bernoulli}(\mathbf{x}; f_\theta(\mathbf{z}_l^{(i)}))$.

Finally, use SGD to maximize

$$\max_\theta \sum_{i=1}^N \frac{1}{L} \sum_{l=1}^L \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}_l^{(i)}) \tag{3}$$

This gives a Monte Carlo estimate of the Wake-phase objective in Eq. (2).

1.2.2 Sleep-phase

The Sleep phase fixes the generative weights θ and updates the recognition weights ϕ. It is generally intractable to directly minimize the KL-divergence term in $\mathcal{L}(\mathbf{x})$ w.r.t. ϕ:

$$\arg\min_\phi D_{\text{KL}}\left[q_\phi(\mathbf{z} \mid \mathbf{x}) \parallel p_\theta(\mathbf{z} \mid \mathbf{x})\right] = \arg\min_\phi \int_{\mathbf{z}} q_\phi(\mathbf{z} \mid \mathbf{x}) \log \frac{q_\phi(\mathbf{z} \mid \mathbf{x})}{p_\theta(\mathbf{z} \mid \mathbf{x})} \, d\mathbf{z}$$

So instead, the Sleep phase minimizes the KL divergence the wrong way round,

$$\arg\min_\phi D_{\text{KL}}\left[p_\theta(\mathbf{z} \mid \mathbf{x}) \parallel q_\phi(\mathbf{z} \mid \mathbf{x})\right].$$

(3 pts) Suppose we sample $\mathbf{z} \sim p(\mathbf{z}) = N(0, I)$, then sample $\mathbf{x} \sim p_\theta(\mathbf{x} \mid \mathbf{z})$. Show that

$$\phi^* := \arg\min_\phi \; D_{\text{KL}}\left[p_\theta(\mathbf{z} \mid \mathbf{x}) \parallel q_\phi(\mathbf{z} \mid \mathbf{x})\right] = \arg\max_\phi \mathbb{E}_{p_\theta(\mathbf{x},\mathbf{z})}\left[\log q_\phi(\mathbf{z} \mid \mathbf{x})\right] \tag{4}$$

which gives the Sleep-phase objective.

Solution

$$
\begin{aligned}
&\arg\min_{\phi} D_{\mathrm{KL}}\left[p_\theta(\mathbf{z} \mid \mathbf{x}) \parallel q_\phi(\mathbf{z} \mid \mathbf{x})\right] \\
&= \arg\min_{\phi} \int_z p_\theta(\mathbf{z} \mid \mathbf{x}) \log \frac{p_\theta(\mathbf{z} \mid \mathbf{x})}{q_\phi(\mathbf{z} \mid \mathbf{x})} d\mathbf{z} \\
&= \arg\min_{\phi} \left[\int_z p_\theta(\mathbf{z} \mid \mathbf{x}) \log p_\theta(\mathbf{z} \mid \mathbf{x}) d\mathbf{z} - \int_z p_\theta(\mathbf{z} \mid \mathbf{x}) \log q_\phi(\mathbf{z} \mid \mathbf{x}) d\mathbf{z}\right] \\
&= \arg\min_{\phi} -\int_z p_\theta(\mathbf{z} \mid \mathbf{x}) \log q_\phi(\mathbf{z} \mid \mathbf{x}) d\mathbf{z} \quad \text{(when } \theta \text{ is fixed, } p_\theta(\mathbf{z} \mid \mathbf{x}) \text{ is also fixed)} \\
&= \arg\max_{\phi} \int_z p_\theta(\mathbf{z} \mid \mathbf{x}) \log q_\phi(\mathbf{z} \mid \mathbf{x}) d\mathbf{z} \\
&= \arg\max_{\phi} \mathbb{E}_{p_\theta(\mathbf{x},\mathbf{z})}\left[\log q_\phi(\mathbf{z} \mid \mathbf{x})\right]
\end{aligned}
$$

Sleep-phase Pseudocode: Let $L \in \mathbb{N}$ be a sample size hyperparameter. For each $l \in [L]$, do the following:

1. Draw $\mathbf{z}^l \sim N(0, I)$.
2. Sample $\mathbf{x}^l$ from the generative network $p_\theta(\mathbf{x} \mid \mathbf{z}^l) = \text{Bernoulli}(f_\theta(\mathbf{z}^l))$.
3. Feed $\mathbf{x}^l$ into the recognition network to get $\mu(\mathbf{x}^l)$ and $\Sigma(\mathbf{x}^l)$.
4. Compute $q_\phi(\mathbf{z}^l \mid \mathbf{x}^l) = N(\mathbf{z}^l; \mu(\mathbf{x}^l), \Sigma(\mathbf{x}^l))$.

Finally, do SGD to maximize

$$\max_{\phi} \frac{1}{L} \sum_{l=1}^{L} \log q_\phi(\mathbf{z}^l \mid \mathbf{x}^l) \tag{5}$$

This gives a Monte Carlo estimate of the Sleep-phase objective in Eq. (4).

1.3 Autoencoding Variational Bayes (AEVB)

In this section, you will derive the optimization procedure for Auto-Encoding Variational Bayes (AEVB). Unlike Wake-Sleep, AEVB avoids the two-stage optimization procedure and instead optimizes a stochastic estimate of ELBO directly w.r.t. to parameters θ of the generative model (generation network) and parameters ϕ of the variational distribution (recognition network).

(3 pts) For a given data point $\mathbf{x}^{(i)}$, show that ELBO can be rewritten as

$$\mathcal{L}(\mathbf{x}^{(i)}) = -D_{\mathrm{KL}}\left[q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \parallel p(\mathbf{z})\right] + \mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z})]. \tag{6}$$

Solution

$$
\begin{aligned}
\mathcal{L}(\mathbf{x}^{(i)}) &= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})] \\
&= \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z})p(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})] \\
&= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) - \log p(\mathbf{z})] + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z})] \\
&= -\int_z q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \log \frac{q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})}{p(\mathbf{z})} d\mathbf{z} + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z})] \\
&= -D_{\mathrm{KL}}\left[q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \parallel p(\mathbf{z})\right] + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x}^{(i)})}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z})]
\end{aligned}
$$

Equation (6) gives a stochastic estimator for ELBO:

$$
\tilde{\mathcal{L}}(\mathbf{x}^{(i)}) = -D_{\mathrm{KL}}\left[q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \parallel p(\mathbf{z})\right] + \frac{1}{L}\sum_{l=1}^{L}[\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)})] \tag{7}
$$

where $\{\mathbf{z}^{(i,l)}\}_{l=1}^{L}$ are sampled from $q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})$. The AEVB algorithm optimizes this stochastic estimate of ELBO using a Monte Carlo gradient estimate.

In order to optimize the AEVB objective in Eq. (7) efficiently, we use a **reparameterization trick** to rewrite $\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}[\cdot]$ such that the Monte Carlo estimate of the expectation is differentiable w.r.t. ϕ. More specifically, we reparameterize the latent variable

$$
\mathbf{z} \sim q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) = N(\mathbf{z} \mid \mu_\phi(\mathbf{x}^{(i)}), \Sigma_\phi^2(\mathbf{x}^{(i)}))
$$

as a deterministic function of the input $\mathbf{x}^{(i)}$ and an auxiliary noise variable ϵ:

$$
\mathbf{z} = \mu_\phi(\mathbf{x}^{(i)}) + \Sigma_\phi(\mathbf{x}^{(i)}) \odot \epsilon \qquad\qquad \epsilon \sim N(0, I) \tag{8}
$$

where $\odot$ signifies an element-wise product, and $\Sigma_\phi(\mathbf{x}^{(i)})$ is a vector of the same size as z.

(4 pts) Using this reparameterization, show that the AEVB objective in Eq. (7) can be rewritten as

$$
\tilde{\mathcal{L}}(\mathbf{x}^{(i)}) = \frac{1}{2}\sum_{j=1}^{J}\left(1 + \log(\Sigma_{(i),j}^2) - \mu_{(i),j}^2 - \Sigma_{(i),j}^2\right) + \frac{1}{L}\sum_{l=1}^{L}\log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)}). \tag{9}
$$

where $\mu_{(i)} := \mu_\phi(\mathbf{x}^{(i)})$ and $\Sigma_{(i)} := \Sigma_\phi(\mathbf{x}^{(i)})$.

Solution

Let J be the dimensionality of $\mathbf{z}$. Let $\mu_{(i)}$ and $\Sigma_{(i)}$ denote the variational mean and s.d. evaluated at datapoint i, and let $\mu_{(i),j}$ and $\Sigma_{(i),j}$ simply denote the j -th element of these vectors. Then:

$$
\begin{aligned}
\int q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \log p(\mathbf{z}) d\mathbf{z} &= \int N(\mathbf{z} \mid \mu_\phi(\mathbf{x}^{(i)}), \Sigma_\phi^2(\mathbf{x}^{(i)})) \log N(\mathbf{z}; \mathbf{0}, \mathbf{I}) d\mathbf{z} \\
&= -\frac{J}{2}\log(2\pi) - \frac{1}{2}\sum_{j=1}^{J}\left(\mu_{(i),j}^2 + \Sigma_{(i),j}^2\right)
\end{aligned}
$$

$$\int q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)})) \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) d\mathbf{z} = \int N(\mathbf{z} \mid \mu_\phi(\mathbf{x}^{(i)}), \Sigma^2_\phi(\mathbf{x}^{(i)})) \log N(\mathbf{z} \mid \mu_\phi(\mathbf{x}^{(i)}), \Sigma^2_\phi(\mathbf{x}^{(i)})) d\mathbf{z}$$
$$= -\frac{J}{2} \log(2\pi) - \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log(\Sigma^2_{(i),j})\right)$$

Then,

$$-D_{\mathrm{KL}} \left[q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \parallel p(\mathbf{z})\right] = \int q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) (\log p(\mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) d\mathbf{z}$$
$$= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log(\Sigma^2_{(i),j}) - \mu^2_{(i),j} - \Sigma^2_{(i),j}\right)$$

Thus,

$$\tilde{\mathcal{L}}(\mathbf{x}^{(i)}) = -D_{\mathrm{KL}} \left[q_\phi(\mathbf{z} \mid \mathbf{x}^{(i)}) \parallel p(\mathbf{z})\right] + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)}))$$
$$= \frac{1}{2} \sum_{j=1}^{J} \left(1 + \log(\Sigma^2_{(i),j}) - \mu^2_{(i),j} - \Sigma^2_{(i),j}\right) + \frac{1}{L} \sum_{l=1}^{L} \log p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)})$$

The AEVB optimization procedure works as follows:

1. For each $l \in [L]$, draw $\epsilon^{(l)} \sim N(0, I)$, and compute $\mathbf{z}^{(i,l)} := \mu_{(i)} + \Sigma_{(i)} \odot \epsilon^{(l)}$.
2. Optimize the AEVB objective in Eq. (9) w.r.t. μ, Σ, and θ.

(3 pt) Derive the gradients of the AEVB objective in Eq. (9) w.r.t. $\mu_{(i),j}$, $\Sigma_{(i),j}$, and θ. (For the gradient w.r.t. θ, you can leave the answer in terms of $p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)})$.)

Solution

$$\nabla_{\mu_{(i),j}} \tilde{\mathcal{L}}(\mathbf{x}^{(i)}) = -\sum_{j=1}^{J} \mu_{(i),j}$$

$$\nabla_{\Sigma_{(i),j}} \tilde{\mathcal{L}}(\mathbf{x}^{(i)}) = \sum_{j=1}^{J} \left(\frac{1}{\Sigma_{(i),j}} - \Sigma_{(i),j}\right)$$

$$\nabla_\theta \tilde{\mathcal{L}}(\mathbf{x}^{(i)}) = \frac{1}{L} \sum_{l=1}^{L} \frac{\nabla_\theta p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)})}{p_\theta(\mathbf{x}^{(i)} \mid \mathbf{z}^{(i,l)})}$$

1.4 Short-answer Questions

(1 pts) Wake-Sleep requires a concurrent optimization of two objective functions, which together do not correspond to the optimization of (a bound of) the marginal likelihood. There is no guarantee that optimizing the Wake-Sleep objectives leads to a decrease in the free energy because:

(Choose 0-2 of the following choices)

A. The sleep phase trains the recognition model to invert the generative model for input vectors that are distributed according to the generative model rather than according to the real data.

B. The sleep phase learning does not follow the correct gradient.

Solution

A B(Because when doing real implementation, we are minimizing the KL divergence the wrong way round).

(1 pt) Between Wake-Sleep and AEVB, which algorithm(s) can be applied to models with discrete latent variables?

Solution

Wake-Sleep algorithm

(1 pts) (True or False) Wake-Sleep and AEVB have the same computational complexity per datapoint.

Solution

True

AEVB is an elegant way to link directed graphical models to neural networks, and is theoretically appeasing because we optimise a (stochastic estimate of the) bound on the likelihood of the data. If the approximations made while performing variational bayes are valid, the training algorithm is guaranteed to increase the likelihood of the generative model. Moreover, there is a clear and recognized way to evaluate the quality of the model using the log-likelihood (either estimated by importance sampling or lower-bounded).

For i.i.d. datasets with continuous latent variables per datapoint, posterior inference for AEVB can be made especially efficient by fitting an approximate inference model (also called a recognition model) to the intractable posterior using the proposed lower bound estimator.

1.5 An Alternate Lower Bound on the Log-Likelihood

To compare trained models, we could simply look at the values of the lower bound. However, the bound could be loose and hence the numbers could be misleading. Here, we derive and prove a tighter approximation of the lower bound on the marginal likelihood, defined as follows:

$$\mathcal{L}_k(\mathbf{x}) = \mathbb{E}_{\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(k)}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta(\mathbf{x},\mathbf{z}^{(i)})}{q_\phi(\mathbf{z}^{(i)}\mid\mathbf{x})}\right]. \tag{10}$$

(4 pts) Prove that $\log p(\mathbf{x}) \geq \mathcal{L}_k(\mathbf{x})$ for any $k \in \mathbb{N}$. (*Hint*: Use Jensen's inequality.)

Solution

Using Jensen's inequality, we get

$$\log p(\mathbf{x}) = \log \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)}\right] \geq \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)}\right] = \mathcal{L}_k(\mathbf{x})$$

(4 pts) Prove that $\mathcal{L}_{k+1}(\mathbf{x}) \geq \mathcal{L}_k(\mathbf{x})$ for any $k \in \mathbb{N}$. You can use the following lemma without proof:

Lemma: Let $I_k \subset [k+1] := \{1,\ldots,k+1\}$ with $|I_k| = k$ be a uniformly distributed subset of distinct

indices from $[k+1]$. Then for any sequence of numbers $a_1, \ldots, a_{k+1}$,

$$\mathbb{E}_{I_k}\left[\frac{\sum_{i\in I_k} a_i}{k}\right] = \frac{\sum_{i=1}^{k+1} a_i}{k+1} \tag{11}$$

Solution

From the lemma and the Jensen's inequality, we get

$$\begin{aligned}
\mathcal{L}_{k+1} &= \mathbb{E}_{\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(k+1)}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k+1}\sum_{i=1}^{k+1}\frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)}\right] && \text{Lemma (11) with } a_i = \frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)} \\
&= \mathbb{E}_{\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(k+1)}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \mathbb{E}_{I_k}\left[\frac{1}{k}\sum_{j=1}^{k}\frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)}\right]\right] && \text{Pick } I_k = \{1,\ldots,k\} \\
&\geq \mathbb{E}_{\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(k+1)}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\mathbb{E}_{I_k}\left[\log \frac{1}{k}\sum_{j=1}^{k}\frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)}\right]\right] && \text{Jensen's inequality} \\
&= \mathbb{E}_{\mathbf{z}^{(1)},\ldots,\mathbf{z}^{(k)}\sim q_\phi(\mathbf{z}|\mathbf{x})}\left[\log \frac{1}{k}\sum_{i=1}^{k}\frac{p_\theta\left(\mathbf{x},\mathbf{z}^{(i)}\right)}{q_\phi\left(\mathbf{z}^{(i)}|\mathbf{x}\right)}\right] = \mathcal{L}_k
\end{aligned}$$

The above two results show that

$$\log p(\mathbf{x}) \geq \mathcal{L}_{k+1}(\mathbf{x}) \geq \mathcal{L}_k(\mathbf{x}).$$

However, the above inequalities do not guarantee $\mathcal{L}_k(\mathbf{x}) \to \log p(\mathbf{x})$ when $k \to \infty$. (The proof is left as an exercise to the reader. Or you can come to my office hours.)

1.6 Experiments (35 pts)

We provide a Jupyter notebook which already contains a working implementation of AEVB:

`https://colab.research.google.com/drive/1nmPXgoNLUKxj-VTwB0l0956Swiv5jrVF`

Note the code is in Tensorflow 1. You are welcome to use Tensorflow 2 or Pytorch for the implementation. If you are using the code framework we have provided, please follow the instructions in the Colab notebook. All provided code will run as is, but you will also need to complete the following TODO's:

1. Implement the Wake-Sleep algorithm by modifying the provided AEVB code.
2. Implement the lower bound metric $\mathcal{L}_{100}$ for $k = 100$ as defined in Eq. (10).
3. Submit your modified .ipynb notebook on Gradescope.

You will use these implementations to complete the remaining sections.

1.6.1 Training

Train both Wake-Sleep and AEVB on the MNIST dataset for 100 epochs (using batch size 100.

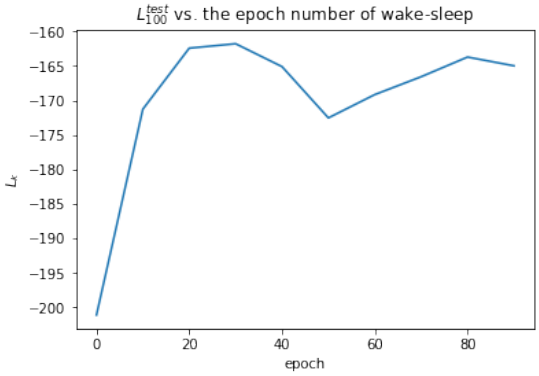
If your Wake-Sleep implementation does not seem to learn a reasonable representation after 100 epochs, you can train it for longer than 100 epochs. If you show that the Wake-Sleep training losses and $\mathcal{L}_{100}^{\text{test}}$ continue to decrease past 100 epochs, and it yields better visualization results, then it would show that your Wake-Sleep implementation is working, and you will still get full points. Report the number of epochs used to train your Wake-Sleep model.

If you encounter `nan`'s even after adjusting your hyperparameters, the problem may be your loss implementation that causes numerical instability issues.

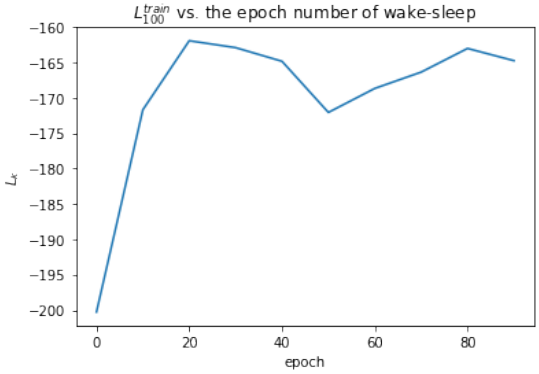
(12 pts) For both Wake-Sleep and AEVB, plot $\mathcal{L}_{100}^{\text{test}}$ vs. the epoch number. To save computation time, you can evaluate $\mathcal{L}_{100}$ every 10 epochs. Which algorithm converged faster? (Grading: 2 points for Wake-Sleep. You can still get 10 points without implementing Wake-Sleep, if you implement $\mathcal{L}_k$ correctly. For each algorithm, you can also get 3 extra credit points for plotting $\mathcal{L}_{100}^{\text{train}}$ vs. the epoch number.)

Solution

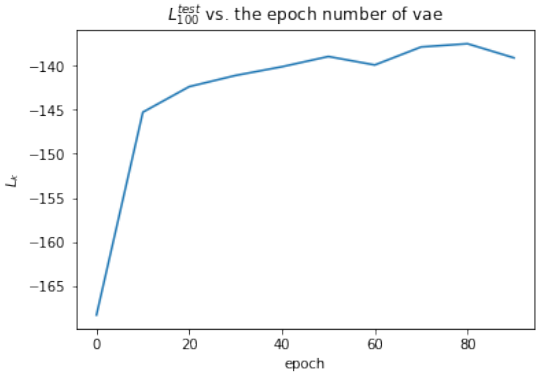
For the Wake-Sleep algorithm, the $\mathcal{L}_k$ on test dataset is shown as below.



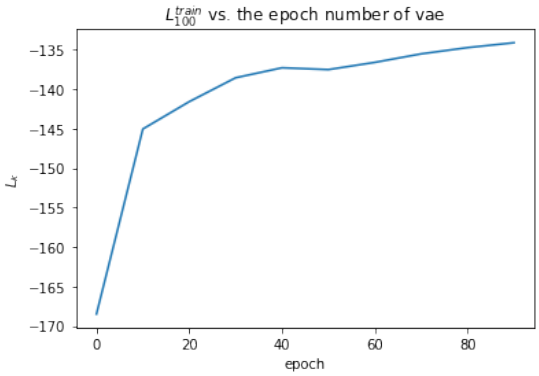
While the $\mathcal{L}_k$ on train dataset of the Wake-Sleep algorithm is as below.



For the AEVB algorithm, the $\mathcal{L}_k$ on test dataset is shown as below.



While the $\mathcal{L}_k$ on train dataset of the AEVB algorithm is as below.

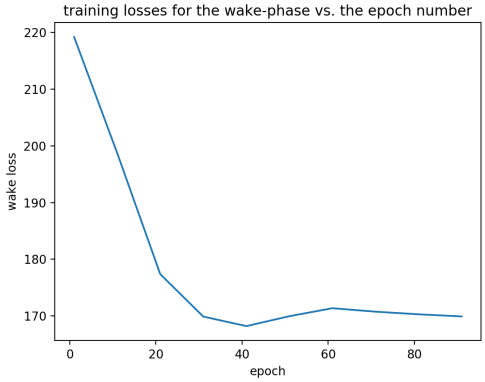


Comparison: As we can see from the $\mathcal{L}_{100}^{\text{test}}$ plots, AEVB algorithm converged faster(AEVB almost converged after 100 epochs of training, however Wake-Sleep is still very shaky) and AEVB yields higher L_k after convergence on both training data and test data.

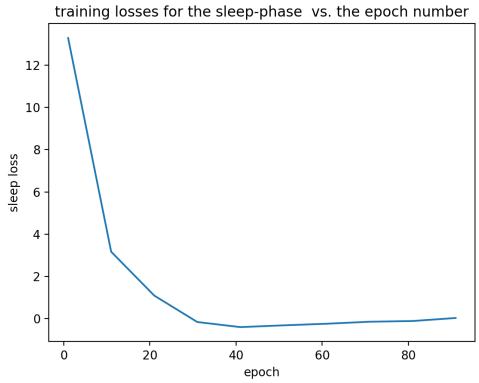
(8 pts) For Wake-Sleep, also plot the training losses for the wake-phase and sleep-phase vs. the epoch number. Specify the hyperparameters used, such as the learning rates for Wake-phase and Sleep-phase, and the sample size used to compute the Monte Carlo estimate of the gradient for the Sleep objective.

Solution

The training losses of the wake-phase is shown as below.



The training losses of the sleep-phase is shown as below.



Here I am using learning rate of 1e-3 for the wake-phase and learning rate of 1e-5 for the sleep-phase. The sample size used to compute the Monte Carlo estimate of the gradient for the sleep objective is set to 100. Training batch size is 100, and epochs number is 100.

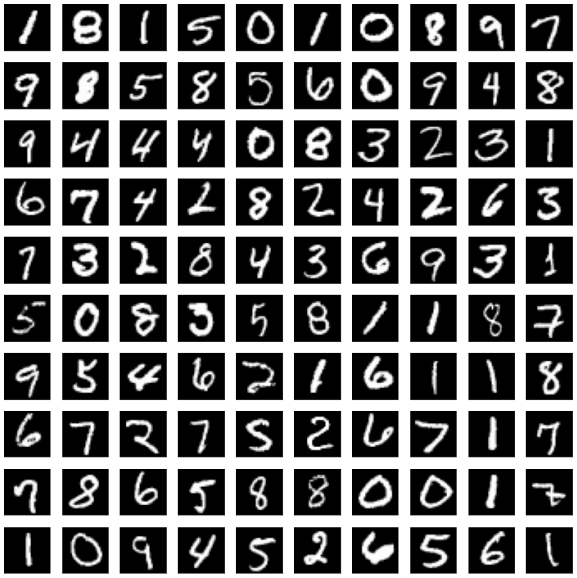
1.6.2 Reconstructed Images

Next, we provide code that samples 100 MNIST images from the test set, uses the recognition network to map them to latent space, then applies the generator network to reconstruct the images.

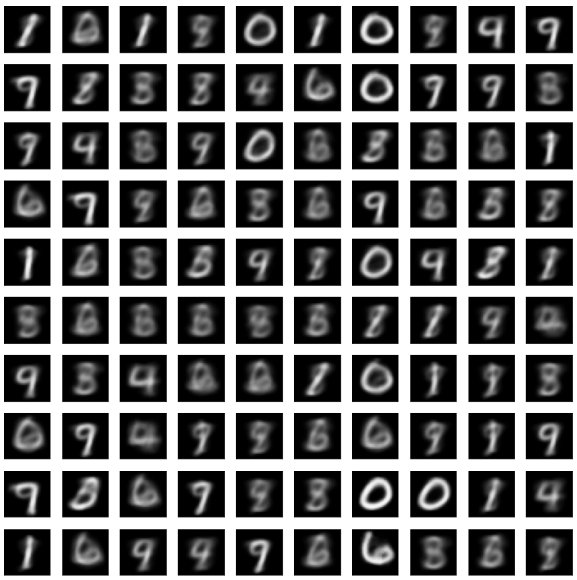
(5 pts) Run this code to visualize these reconstruction samples $\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(100)}$ on a 10×10 tile grid. Also visualize the original MNIST images $\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(100)}$ on a 10×10 tile grid. Briefly compare the results for Wake-Sleep vs. AEVB. (Grading: 2 points for Wake-Sleep, 2 points for AEVB, 1 point for comparing the two.)

Solution

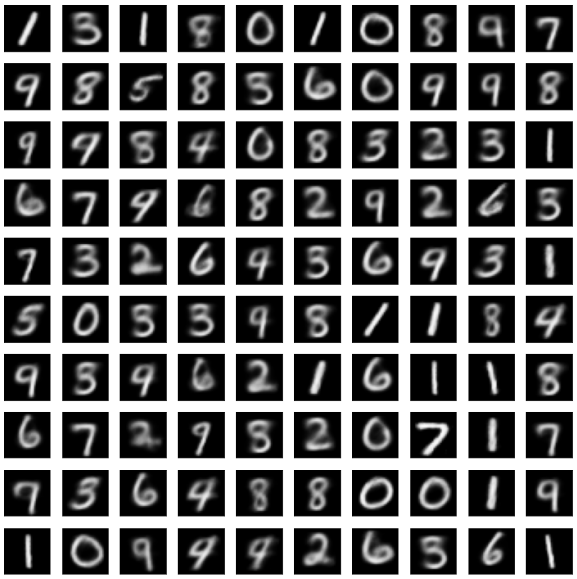
The original MNIST images are shown as below:



The reconstruction samples visualized from Wake-Sleep algorithm are



The reconstruction samples visualized from AEVB are



Comparison: As shown above, the reconstruction samples of wake-sleep algorithm are pretty blurred, for example some images show digits part like "8" and part like"6", while the reconstruction samples from AEVB are more similar to real digits. Thus, the AVEB results here are better.

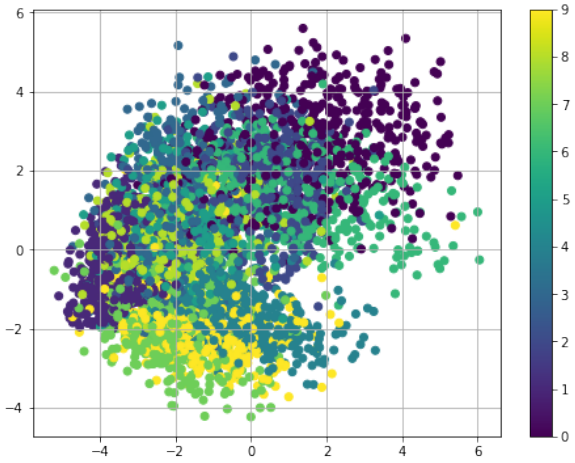
1.6.3 Latent Space Visualization (Part 1)

Since we have specifically chosen the latent space to be 2-dimensional, now we can easily visualize the learned latent manifold of digits. We provide code that samples 5000 MNIST images from the test set, and visualize their latent representations as a scatter plot, where colors of the points correspond to the digit labels.

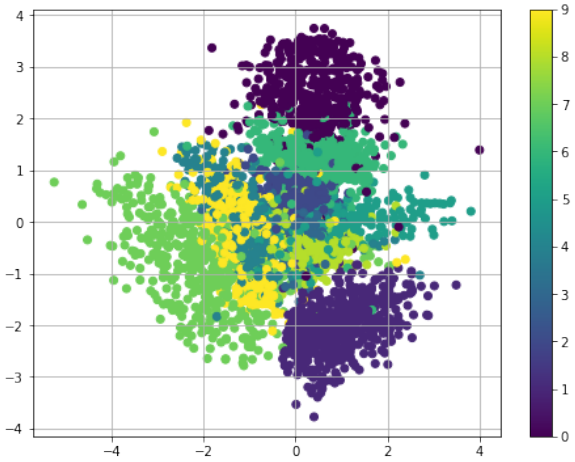
(5 pts) Run this code to visualize the latent space scatterplot. Briefly compare the results for Wake-Sleep vs. AEVB. (Grading: 2 points for Wake-Sleep, 2 points for AEVB, 1 point for comparing the two.)

Solution

The learned latent space representations of Wake-Sleep is shown as below.



The learned latent space representations of AEVB is shown as below.



Comparison: As shown above, the learned latent space representations from Wake-Sleep algorithm are more dispersive. Though with some clustering trend, the representations of some digits still overlapped a lot. While the representations learned from AVEB are more clustered and more disjuct among different digit types. Thus here the results from AEVB are better.

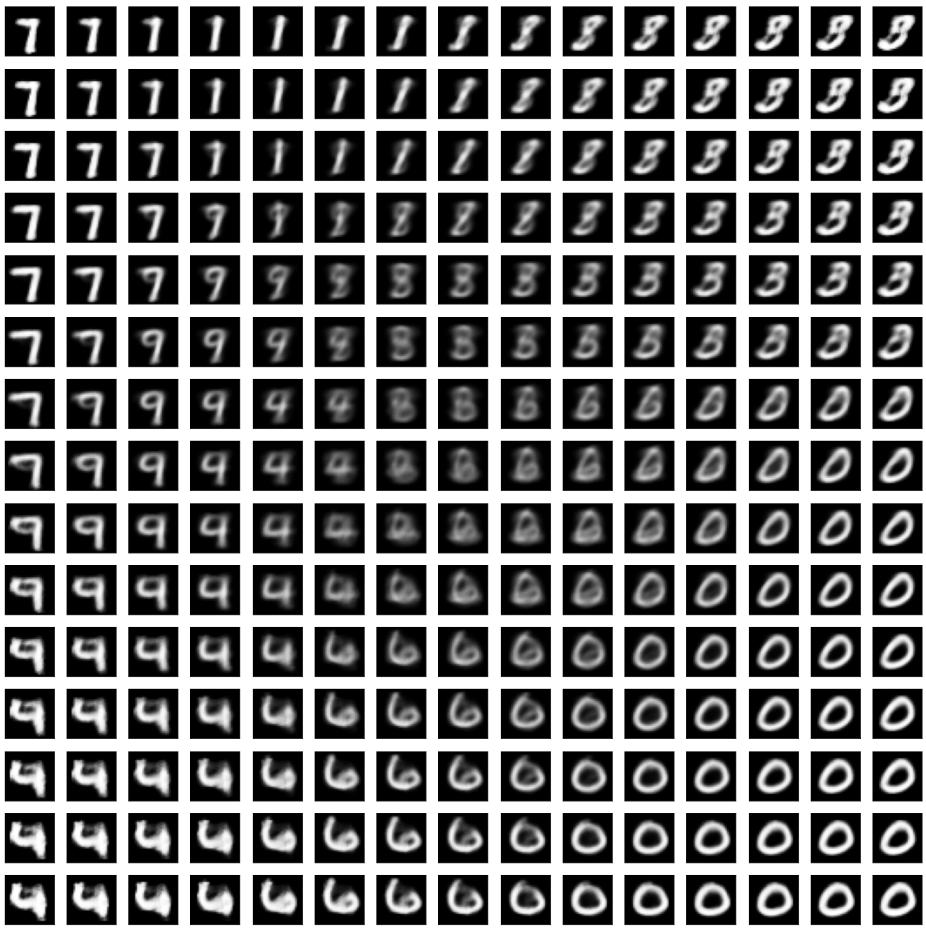
1.6.4 Latent Space Visualization (Part 2)

Finally, we provide code that uses the generator network to plot reconstructions at the positions in the latent space for which they have been generated.

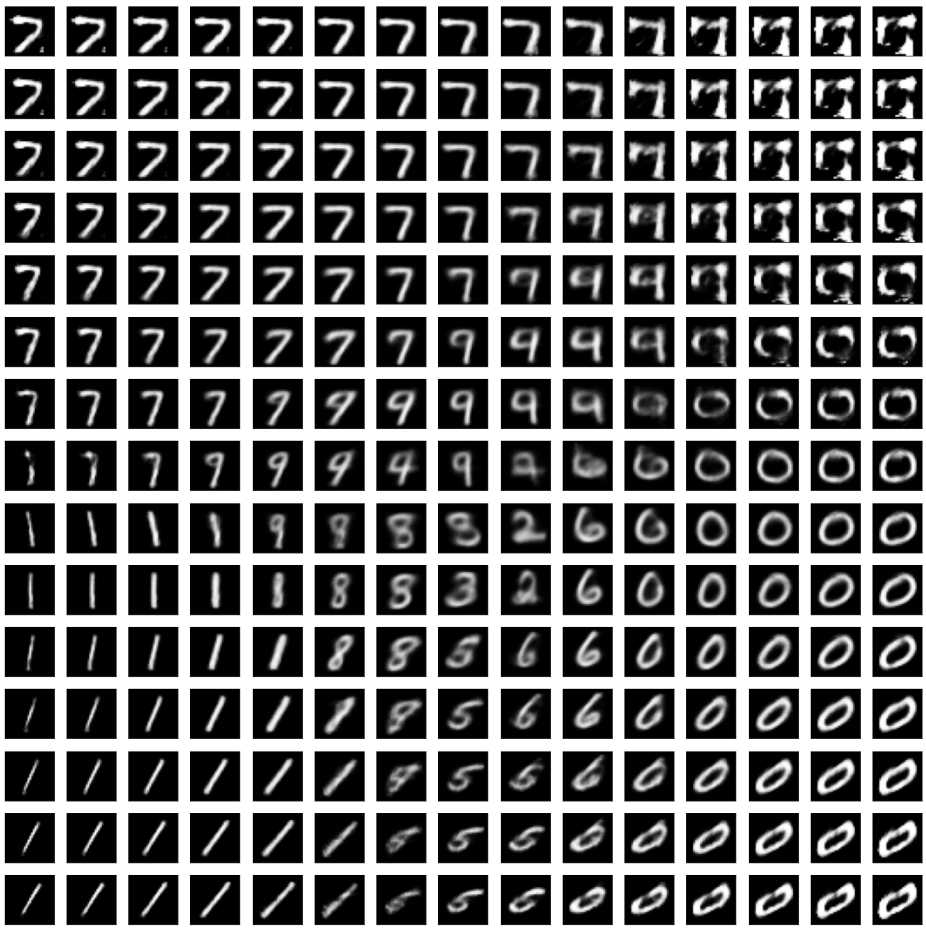
(5 pts) Run this code to visualize the latent space reconstructions. Briefly compare the results for Wake-Sleep vs. AEVB. (Grading: 2 points for Wake-Sleep, 2 points for AEVB, 1 point for comparing the two.)

Solution

The latent space reconstructions from Wake-Sleep are as below.



The latent space reconstructions from AEVB are as below.



Comparison: As the visualization above shown, the latent variable from Wake-Sleep learned some features from the digits, such as angle of "7" and circle pattern of "0", however some of the features learned such as presented on the left lower corner are not high level enough to be related with specific digits. The overall latent space is blurry with only shape like 0,1,7,9 can be seen. While the latent variable from AEVB learned more distinct features such as lines, angles, cube corners and circles, and we can clearly see features with shape like 0, 1, 5, 6, 8, 9 in AEVB latent space.

References

[1] Peter Dayan. Helmholtz machines and wake-sleep learning. *Handbook of Brain Theory and Neural Network. MIT Press, Cambridge, MA*, 44(0), 2000.

[2] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.

[3] Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The "wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

[4] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.