Variational Inference and Deep Learning

Can Yang

Department of Mathematics
The Hong Kong University of Science and Technology

October 29, 2018

Learning from data

- Assume the observed variable $\mathbf{x}$ is a random sample from an **unknown underlying process**, whose true distribution $p^*(\mathbf{x})$ is unknown. We attempt to approximate this underlying process with a chosen model $p_{\boldsymbol{\theta}}(\mathbf{x})$, with parameter $\boldsymbol{\theta}$:

$$\mathbf{x} \sim p_{\boldsymbol{\theta}}(\mathbf{x})$$

- Learning is, most commonly, the process of searching for a value of the parameter $\boldsymbol{\theta}$ such that the probablity distribution function given by the model, $p_{\boldsymbol{\theta}}(\mathbf{x})$, approximates the true distribution of the data, denoted by $p^*(\mathbf{x})$, such that for any obseved $\mathbf{x}$:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) \approx p^*(\mathbf{x})$$

Conditional models

- In the case of classification or regression problems, we are not interested in learning an unconditional model $p_{\boldsymbol{\theta}}(\mathbf{x})$, but a conditional model $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ that approximates the underlying conditional distribution $p^*(\mathbf{y}|\mathbf{x})$:

$$p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x}) \approx p^*(\mathbf{y}|\mathbf{x}).$$

- $\mathbf{x}$ is an image, and $\mathbf{y}$ is the image's class (labeled by a human). In this case, $p_{\boldsymbol{\theta}}(\mathbf{y}|\mathbf{x})$ is typically chose to be a categorical distribution.
- Examples include (multi-class) logistic regression and neural networks:

$$\begin{aligned} \mathbf{p} &= \text{NeuralNet}(\mathbf{x}) \\ p_{\boldsymbol{\theta}}(y|\mathbf{x}) &= \text{Categorical}(y; \mathbf{p}) \end{aligned}$$

Learning in fully observed models

- We often collect a dataset $\mathcal{D}$ consisting of $N \geq 1$ datapoints

$$\mathcal{D} = \{\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \ldots, \mathbf{x}^{(N)}\} = \{\mathbf{x}^{(i)}\}_{i=1}^{N}.$$

- The observations $\{\mathbf{x}^{(i)}\}_{i=1}^{N}$ are said to be *i.i.d.*, for independently and identically distributed.
- With i.i.d. dataset $\mathcal{D}$ of size $N_{\mathcal{D}}$, the maximum likelihood criterion is

$$\log p_{\boldsymbol{\theta}}(\mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log p_{\boldsymbol{\theta}}(\mathbf{x}).$$

- Gradient methods can be used to find a local optimum of the ML objective: $\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t+1)} + s_t \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$.

Stochastic gradient descent (SGD)

- Consider a minibatches of data $\mathcal{M} \subset \mathcal{D}$ of size $N_{\mathcal{M}}$. With such minibatches, we can form an unbiased estimator of the ML criterion:

$$\frac{1}{N_{\mathcal{D}}} \log p_{\boldsymbol{\theta}}(\mathcal{D}) \simeq \frac{1}{N_{\mathcal{M}}} \log p_{\boldsymbol{\theta}}(\mathcal{M}) = \frac{1}{N_{\mathcal{M}}} \sum_{\mathbf{x} \in \mathcal{M}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

- The unbiased estimator $\log p_{\boldsymbol{\theta}}(\mathcal{M})$ is differentiable, yielding the unbiased stochasic gradients:

$$\frac{1}{N_{\mathcal{D}}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathcal{D}) \simeq \frac{1}{N_{\mathcal{M}}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathcal{M}) = \frac{1}{N_{\mathcal{M}}} \sum_{\mathbf{x} \in \mathcal{M}} \nabla_{\boldsymbol{\theta}} \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

Learning and inference in deep latenet variable models

- Latent variables are variables that are part of the model, but which we don't observe, and are therefore not part of the data. We typically use $\mathbf{z}$ to denote such latent variables.
- The marginal distribution over the observed variables is given by

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z} = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{z}) d\mathbf{z}$$

- Such an implicit distribution over $\mathbf{x}$ can be quite flexible. If $\mathbf{z}$ is discrete and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ is a Gaussian distribution, then $p_{\boldsymbol{\theta}}(\mathbf{x})$ is a mixture-of-Gaussians distribution. For continuous $\mathbf{z}$, $p_{\boldsymbol{\theta}}(\mathbf{x})$ can be seen as an infinite mixture, which are potentially more powerful than discrete mixtures.
- As we shall see in variational auto encode (VAE), the continuous version is generally more efficient to work with due to the re-parameterization trick.

Deep Latent Variable Model (DLVM)

- Joint distribution

$$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) = p_{\boldsymbol{\theta}}(\mathbf{z}) p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}),$$

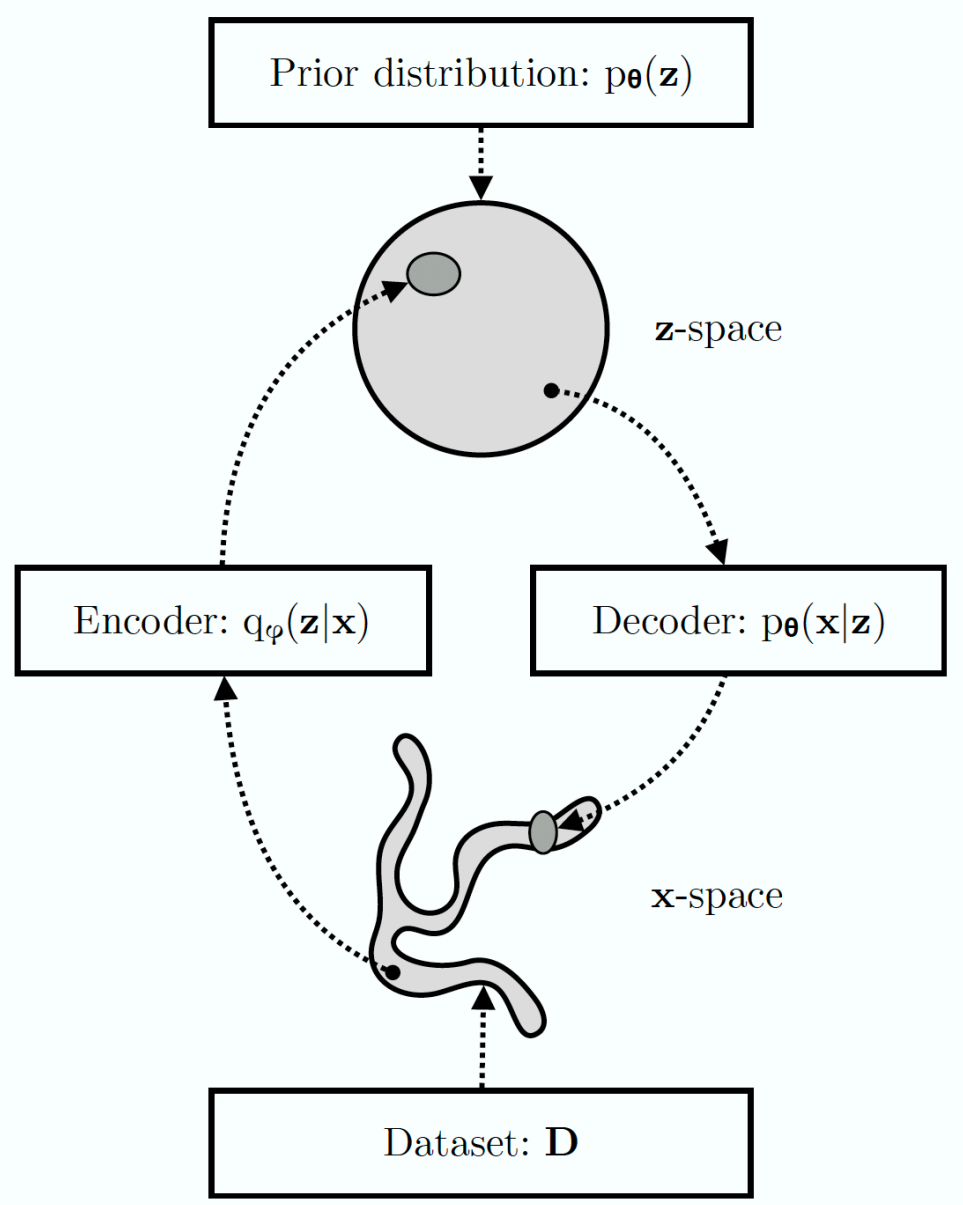
where $p_{\boldsymbol{\theta}}(\mathbf{z})$ is the prior distribution over $\mathbf{z}$ and $p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$ can be specified via deep neural networks.

- Generative model with spherical Gaussian latent space and Bernoulli observation:

$$\begin{aligned} p(\mathbf{z}) &= \mathcal{N}(\mathbf{z}; \mathbf{0}, \mathbf{I}) \\ \mathbf{p} &= \text{DecoderNeuralNet}_{\boldsymbol{\theta}}(\mathbf{z}) \\ \log p(\mathbf{x}|\mathbf{z}) &= \sum_{j=1}^{D} \log p(x_j|\mathbf{z}) = \sum_{j=1}^{D} \log \text{Bernoulli}(x_j; p_j) \\ &= \sum_{j=1}^{D} x_j \log p_j + (1 - x_j) \log(1 - p_j) \end{aligned} \tag{1}$$

where $\forall p_j \in \mathbf{p} : 0 \leq p_j \leq 1$ and $\sum_j p_j = 1$.

- Intractability: $p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}) d\mathbf{z}$ is often intractable.


Prior distribution: $p_\theta(\mathbf{z})$
$\mathbf{z}$-space
Encoder: $q_\varphi(\mathbf{z}|\mathbf{x})$
Decoder: $p_\theta(\mathbf{x}|\mathbf{z})$
$\mathbf{x}$-space
Dataset: $\mathbf{D}$

Variational Autoencoders (VAE)

- To turn the DLVM intractable posterior inference and learning problems into tractable problems, an inference model $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is introduced so that

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}),$$

  where $\boldsymbol{\phi}$ is variational parameter.
- Like the generative model, $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ can be parameterized using deep nueral network. For example:

$$\begin{aligned} (\boldsymbol{\mu}, \log \boldsymbol{\sigma}) &= \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x}) \\ q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma})) \end{aligned} \tag{2}$$

  in this case, the variational parameter $\boldsymbol{\phi}$ include the weights coefficients of the neural network.

Objective: Evidence Lower Bound (ELBO)

- For any choice of $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$,

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\mathbf{x})] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}\right]\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}\right]\right] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\right]\right] + \mathbb{E}_{q_{\boldsymbol{\phi}}}\left[\log\left[\frac{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}\right]\right] \\
&= \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) + \mathrm{KL}(\mathrm{q}_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})||\mathrm{p}_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))
\end{aligned}
\tag{3}
$$

Connection with EM

- For standard EM algorithms, the posterior is often known, $q(\mathbf{z}|\mathbf{x}) = p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$.
- Then the KL term becomes zero.

$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \mathcal{L}_{\boldsymbol{\theta}}(\mathbf{x}) = \mathbb{E}_q[\log p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})] - \mathbb{E}_q[\log q(\mathbf{z}|\mathbf{x})].$$

- The above step is indeed the E-step in the standard EM algorithm.
- The M-step is

$$\theta_{\text{new}} = \arg\max_{\boldsymbol{\theta}} L_{\boldsymbol{\theta}}(\mathbf{x}).$$

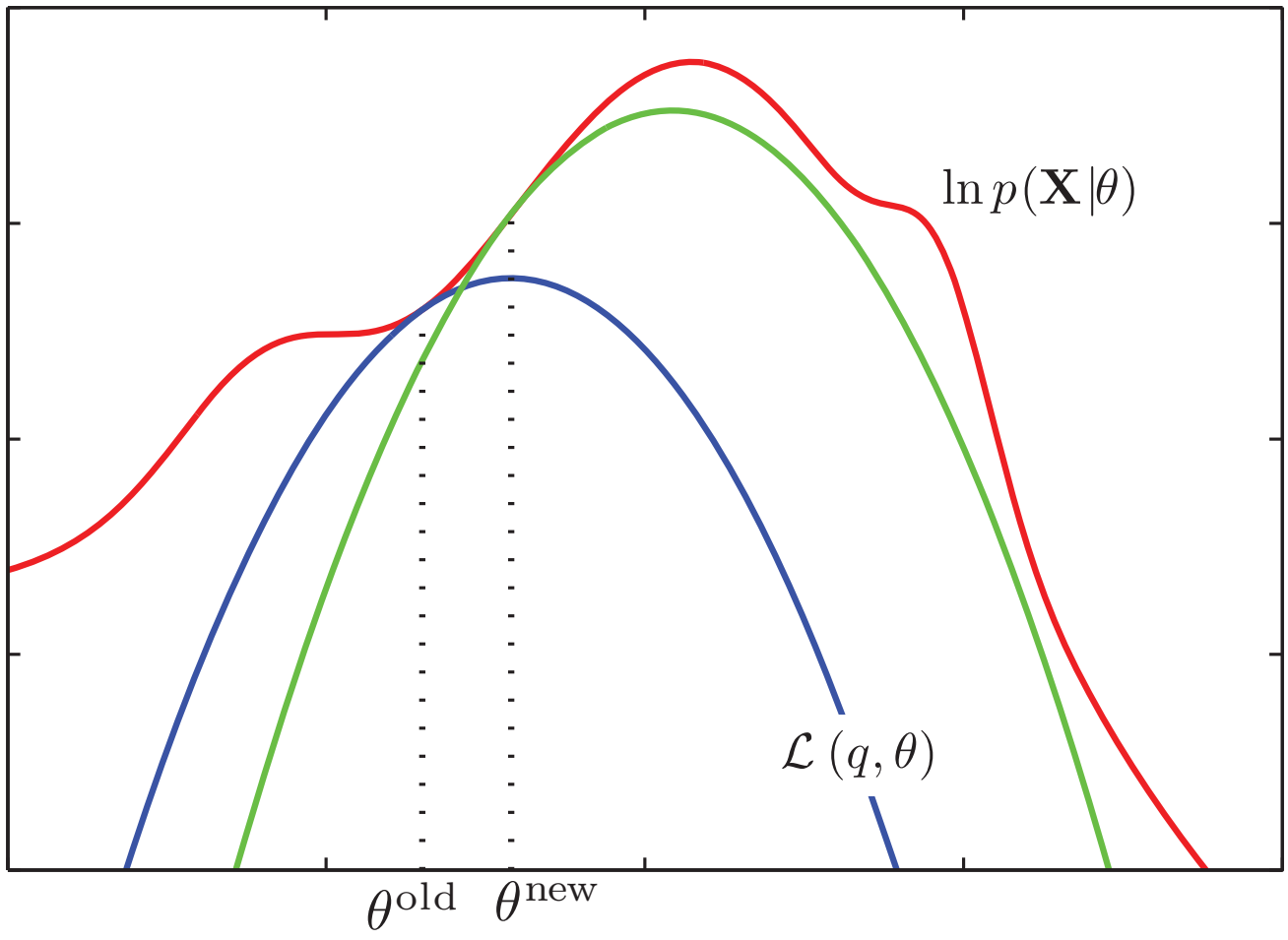


Figure: The EM algorithm involves alternately computing a lower bound on the log likelihood for the current parameter values and then maximizing this bound to obtain the new parameter values.

Stochastic gradient-based optimization

- Given i.i.d. data $\mathbf{x} \in \mathcal{D}$

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{D}) = \sum_{\mathbf{x}\in\mathcal{D}} \mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) \tag{4}$$

- Consider the single-data point ELBO. Its gradient $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$ is in general intractable. But the unbiased estimator exists.

$$\begin{aligned}
\nabla_{\boldsymbol{\theta}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\theta}}\mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \\
&= \mathbb{E}_{q_{\boldsymbol{\phi}}}[\nabla_{\boldsymbol{\theta}}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))] \\
&\simeq \nabla_{\boldsymbol{\theta}}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})) \\
&= \nabla_{\boldsymbol{\theta}}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})),
\end{aligned} \tag{5}$$

  where $\simeq$ means unbiased estimator. The $\mathbf{z}$ in the last two lines are sampled from $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$

- The same equation does not apply to $\boldsymbol{\phi}$ in general

$$\begin{aligned}
\nabla_{\boldsymbol{\phi}}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \nabla_{\boldsymbol{\phi}}\mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \\
&\neq \mathbb{E}_{q_{\boldsymbol{\phi}}}[\nabla_{\boldsymbol{\phi}}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}))]
\end{aligned} \tag{6}$$

Reparameterization trick

- We express random variable $\mathbf{z}$, as differentiable and invertible transformation of another random variable $\boldsymbol{\epsilon}$, given $\mathbf{x}$ and $\boldsymbol{\phi}$,

$$\mathbf{z} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x}) \tag{7}$$

  such that the distribution of $\boldsymbol{\epsilon}$ is independent of $\mathbf{x}$ or $\boldsymbol{\phi}$.

- Using change of variable, the expectation becomes

$$\mathbb{E}_{q_{\boldsymbol{\phi}}}[f(\mathbf{z})] = \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(\mathbf{z})] \tag{8}$$

- The expectation is independent of $\boldsymbol{\phi}$ and the expectation and the gradient becomes communicative

$$\begin{aligned} \nabla_{\boldsymbol{\phi}} \mathbb{E}_{q_{\boldsymbol{\phi}}}[f(\mathbf{z})] &= \nabla_{\boldsymbol{\phi}} \mathbb{E}_{p(\boldsymbol{\epsilon})}[f(g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x}))] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\nabla_{\boldsymbol{\phi}} f(g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x}))] \\ &\simeq \nabla_{\boldsymbol{\phi}} f(g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})), \end{aligned} \tag{9}$$

  where in the last line a random noise sample $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ in $g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$.

Estimate of gradient

- With the reparameterization trick, the ELBO can be written as

$$\begin{aligned}\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \mathbb{E}_{q_{\boldsymbol{\phi}}}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})] \\ &= \mathbb{E}_{p(\boldsymbol{\epsilon})}[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})]\end{aligned} \tag{10}$$

- Using a single noise sample $\boldsymbol{\epsilon}$ from $p(\boldsymbol{\epsilon})$, Monte Carlo estimate of $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}$ can be obtained:

$$\begin{aligned}\boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) \\ \mathbf{z} &= g(\boldsymbol{\epsilon},\boldsymbol{\phi},\mathbf{x}) \\ \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) &= \log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\end{aligned} \tag{11}$$

- Specifically, $\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is computed by

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \log p(\boldsymbol{\epsilon}) - \log \left|\det\left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}}\right)\right| \tag{12}$$

Choosing function $g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$

- A common choice is simple factorized Gaussian encoder $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \text{diag}(\boldsymbol{\sigma}))$:

$$\begin{aligned} (\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2) &= \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x}) \\ q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) &= \prod_i q_{\boldsymbol{\phi}}(z_i|\mathbf{x}) = \prod_i \mathcal{N}(z_i; \mu_i, \sigma_i^2) \end{aligned} \tag{13}$$

- After re-parameterization,

$$\begin{aligned} \boldsymbol{\epsilon} &\sim p(\boldsymbol{\epsilon}) \\ (\boldsymbol{\mu}, \log \boldsymbol{\sigma}^2) &= \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x}) \\ \mathbf{z} &= \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon} \end{aligned} \tag{14}$$

  where $\odot$ denotes element-wise product.

- Since $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} = \text{diag}(\boldsymbol{\sigma})$, we have $\log |\det(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}})| = \sum_i \log \sigma_i$ and the log of posterior is

$$\log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \sum_i \log \mathcal{N}(\epsilon_i; 0, 1) - \log \sigma_i \tag{15}$$

Algorithm 1: Stochastic optimization of the ELBO. Since noise originates from both the minibatch sampling and sampling of $p(\boldsymbol{\epsilon})$, this is a doubly stochastic optimization procedure. We also refer to this procedure as the *Auto-Encoding Variational Bayes* (AEVB) algorithm.

Data:

$\mathcal{D}$: Dataset

$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: Inference model

$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

$\boldsymbol{\theta}, \boldsymbol{\phi}$: Learned parameters

$(\boldsymbol{\theta}, \boldsymbol{\phi}) \leftarrow$ Initialize parameters

while *SGD not converged* **do**

  $\mathcal{M} \sim \mathcal{D}$ (Random minibatch of data)

  $\boldsymbol{\epsilon} \sim p(\boldsymbol{\epsilon})$ (Random noise for every datapoint in $\mathcal{M}$)

  Compute $\tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$ and its gradients $\nabla_{\boldsymbol{\theta},\boldsymbol{\phi}} \tilde{\mathcal{L}}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathcal{M}, \boldsymbol{\epsilon})$

  Update $\boldsymbol{\theta}$ and $\boldsymbol{\phi}$ using SGD optimizer

end

Full covariance Gaussian posterior

- Consider the Gaussian with full covariance:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}; \boldsymbol{\mu}, \boldsymbol{\Sigma}) \tag{16}$$

with re-parameterization:

$$\begin{aligned} \boldsymbol{\epsilon} &\sim \mathcal{N}(0, \mathbf{I}) \\ \mathbf{z} &= \boldsymbol{\mu} + \mathbf{L}\boldsymbol{\epsilon} \end{aligned} \tag{17}$$

where $\mathbf{L}$ is a lower triangular matrix with non-zero entries in the diagonal. The off-diagonal elements define the correlations of the elements in $\mathbf{z}$.

- The Jacobian is $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}} = \mathbf{L}$, hence

$$\log|\det(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}})| = \sum_i \log|L_{ii}| \tag{18}$$

Full covariance Gaussian posterior

- This corresponds to the Cholesky decomposition $\boldsymbol{\Sigma} = \mathbf{L}\mathbf{L}^T$:

$$\begin{aligned} \boldsymbol{\Sigma} &= \mathbb{E}[(\mathbf{z} - \mathbb{E}[\mathbf{z}])(\mathbf{z} - \mathbb{E}[\mathbf{z}])^T] \\ &= \mathbb{E}(\mathbf{L}\boldsymbol{\epsilon}(\mathbf{L}\boldsymbol{\epsilon})^T) = \mathbf{L}\mathbb{E}[\boldsymbol{\epsilon}\boldsymbol{\epsilon}^T]\mathbf{L}^T \\ &= \mathbf{L}\mathbf{L}^T \end{aligned} \tag{19}$$

- To build the desired matrix $\mathbf{L}$, we can

$$\begin{aligned} (\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}') &\leftarrow \text{EncoderNueralNet}_{\boldsymbol{\phi}}(\mathbf{x}) \\ \mathbf{L} &\leftarrow \mathbf{L}_{mask} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma}) \end{aligned} \tag{20}$$

where $\mathbf{L}_{mask}$ is a masking matrix with zeros on and above the diagonal, and ones below the diagonal. Then $\log|\det(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\epsilon}})| = \sum_i \log \sigma_i$.

Algorithm 2: Computation of unbiased estimate of single-datapoint ELBO for example VAE with a full-covariance Gaussian inference model and a factorized Bernoulli generative model. $\mathbf{L}_{mask}$ is a masking matrix with zeros on and above the diagonal, and ones below the diagonal. Note that $\mathbf{L}$ must be a triangular matrix with positive entries on the diagonal.

Data:

$\mathbf{x}$: a datapoint, and optionally other conditioning information
$\boldsymbol{\epsilon}$: a random sample from $p(\boldsymbol{\epsilon}) = \mathcal{N}(0, \mathbf{I})$
$\boldsymbol{\theta}$: Generative model parameters
$\boldsymbol{\phi}$: Inference model parameters
$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$: Inference model
$p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})$: Generative model

Result:

$\tilde{\mathcal{L}}$: unbiased estimate of the single-datapoint ELBO $\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x})$

$(\boldsymbol{\mu}, \log \boldsymbol{\sigma}, \mathbf{L}') \leftarrow \text{EncoderNeuralNet}_{\boldsymbol{\phi}}(\mathbf{x})$
$\mathbf{L} \leftarrow \mathbf{L}_{mask} \odot \mathbf{L}' + \text{diag}(\boldsymbol{\sigma})$
$\boldsymbol{\epsilon} \sim \mathcal{N}(0, \mathbf{I})$
$\mathbf{z} \leftarrow \mathbf{L}\boldsymbol{\epsilon} + \boldsymbol{\mu}$
$\text{logqz} \leftarrow -\texttt{sum}(\frac{1}{2}(\boldsymbol{\epsilon}^2 + \log(2\pi) + \log \boldsymbol{\sigma}))$ $\quad \triangleright = q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$
$\text{logpz} \leftarrow -\texttt{sum}(\frac{1}{2}(\mathbf{z}^2 + \log(2\pi)))$ $\quad \triangleright = p_{\boldsymbol{\theta}}(\mathbf{z})$
$\mathbf{p} \leftarrow \text{DecoderNeuralNet}_{\boldsymbol{\theta}}(\mathbf{z})$
$\text{logpx} \leftarrow \texttt{sum}(\mathbf{x} \odot \log \mathbf{p} + (1 - \mathbf{x}) \odot \log(1 - \mathbf{p}))$ $\quad \triangleright = p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})$
$\tilde{\mathcal{L}} = \text{logpx} + \text{logpz} - \text{logqz}$

Estimation of marginal likelihood

- Importance sampling can be adopted in the estimation of marginal likelihood.
- From

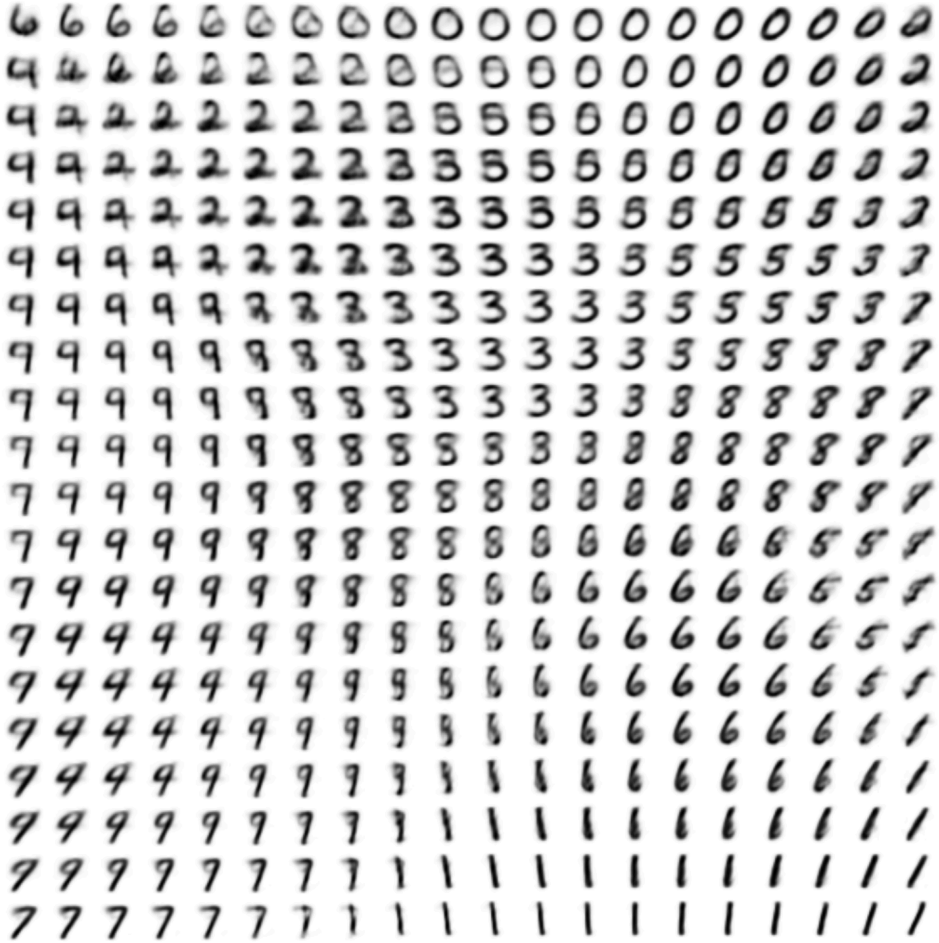
$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) = \log \mathbb{E}_{q_{\boldsymbol{\phi}}}[p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z})/q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})],$$

  we draw random samples form $q_{\boldsymbol{\phi}}$ to obtain a Monte Carlo estimator:

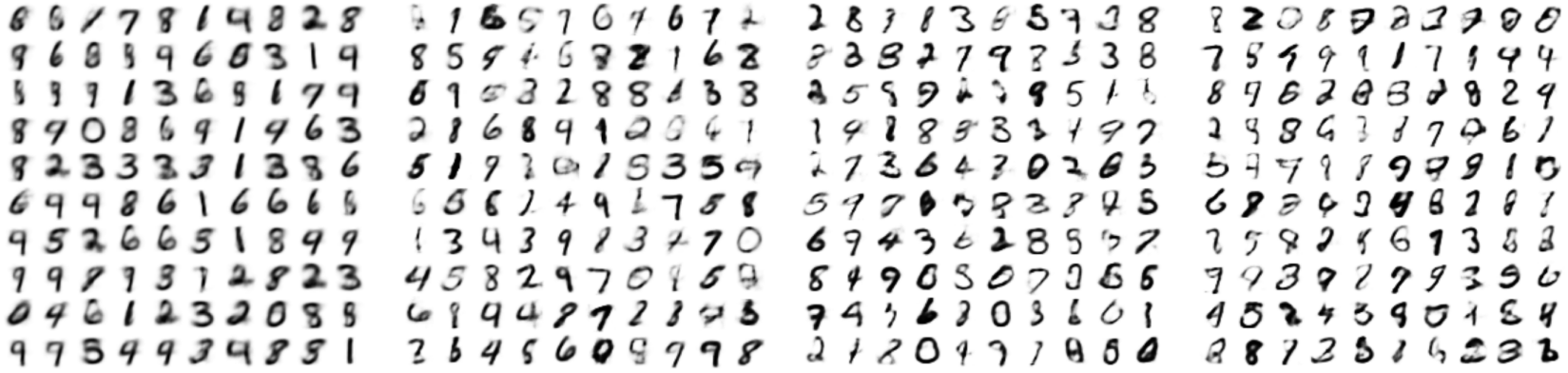
$$\log p_{\boldsymbol{\theta}}(\mathbf{x}) \approx \log \frac{1}{L} \sum_{l=1}^{L} p_{\boldsymbol{\theta}}(\mathbf{x}, \mathbf{z}^l)/q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}^l)$$

- When $L = 1$, this equals the ELBO estimator of the VAE.

(a) Learned Frey Face manifold



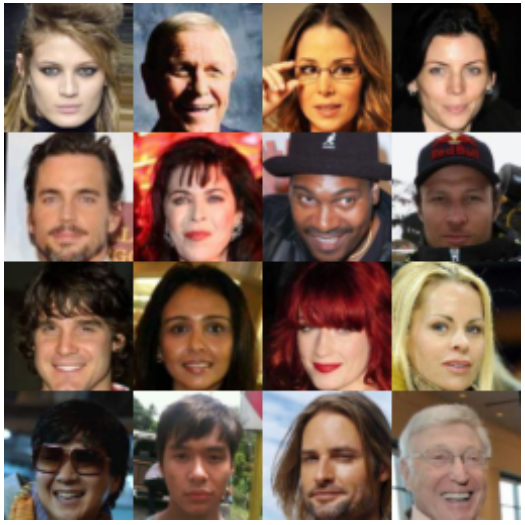
(b) Learned MNIST manifold



(a) 2-D latent space (b) 5-D latent space (c) 10-D latent space (d) 20-D latent space

Figure: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

Plain VAE



(a) Real

(b) Reconstruction

(c) Generation

VAE VGG

(d) Real

(e) Reconstruction

(f) Generation

More on VAE

- Objective function (population version):

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \mathbb{E}_{p(\mathbf{x})}[\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]] + \mathbb{E}_{p(\mathbf{x})}[\mathrm{KL}(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))] \tag{21}$$

- $\mathrm{Loss}(\boldsymbol{\phi},\boldsymbol{\theta}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}[-\log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})]$.
- $\mathrm{KL}(q_{\phi}(\mathbf{z}|\mathbf{x})\|p(\mathbf{z}))$ can be viewed as regularization and it can be evaluated in the closed form (KL of two Gaussians).
- Loss + Regularization
- The derivation can be found in paper Tutorial of VAE by C. Doersch.

Adversarial AutoEncoders

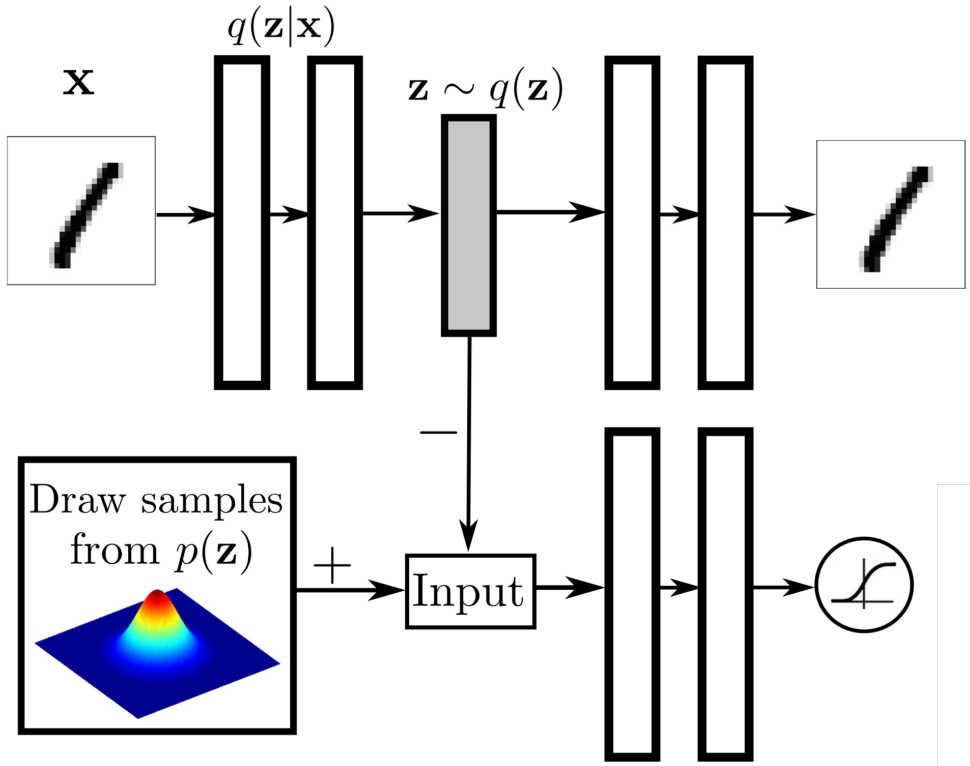


Figure: An AAE example for MNIST

Adversarial AutoEncoders

- AAE: Plain AE in the data space + GAN in the latent space
- Goal: learn regularized PAE
- Latent space: learn to match the prior with adversarial loss
- Treat the encoder as generator in the latent space
- Objective function:

$$\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \left[\text{Loss}(\boldsymbol{\phi},\boldsymbol{\theta}) + \max_{f} \left[\mathbb{E}_{p(\mathbf{z})}[\log f(\mathbf{z})] + \mathbb{E}_{q(\mathbf{z})}[\log(1 - f(\mathbf{z}))] \right] \right]$$

$$q(\mathbf{z}) = \int q_\phi(\mathbf{z}|\mathbf{x}) p(\mathbf{x}) d\mathbf{x} \tag{22}$$

Wasserstein AutoEncoders

- WAE: the optimal transport view of AAE
- Goal: learn regularized PAE with optimal transport in data space
- Latent space: learned by GAN or maximum mean discrepancy (MMD)
- Objective function:

$$\begin{aligned} &\min_{\phi,\theta} \mathbb{E}_{p(x)}\mathbb{E}_{q_\phi(z|x)} c(x, D_\theta(z)) + \lambda \cdot \mathcal{D}(p(z), q(z)) \\ &q(z) = \int_x q_\phi(z|x)p(x)dx \end{aligned} \tag{23}$$

where $c()$ could be the least square loss and $\mathcal{D}(\cdot, \cdot)$ is a distance measure of two density function.

Some New Results

(a) Real

(b) Reconstruction

(c) Generation

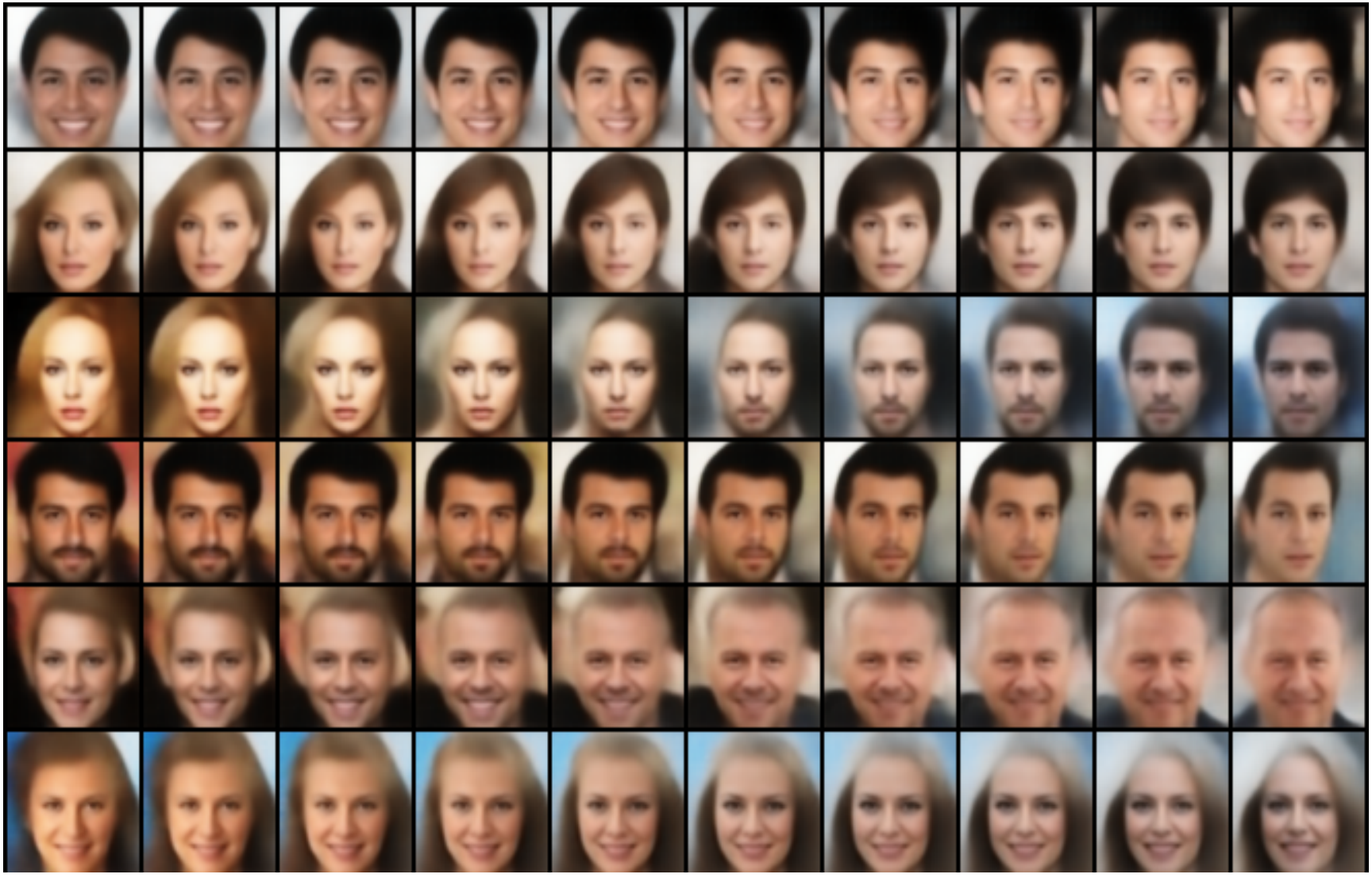
Some New Results

(d) Real

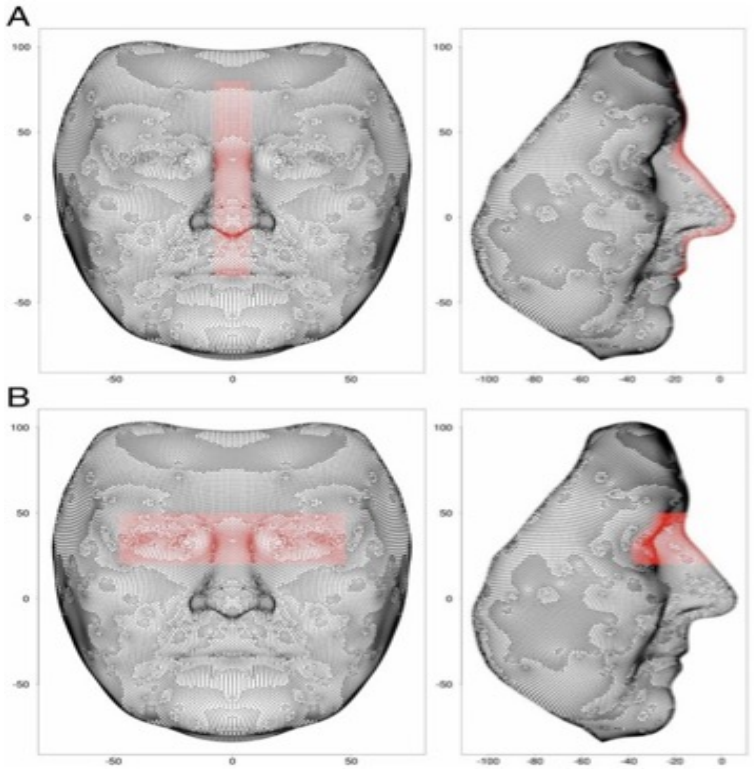
(e) Reconstruction

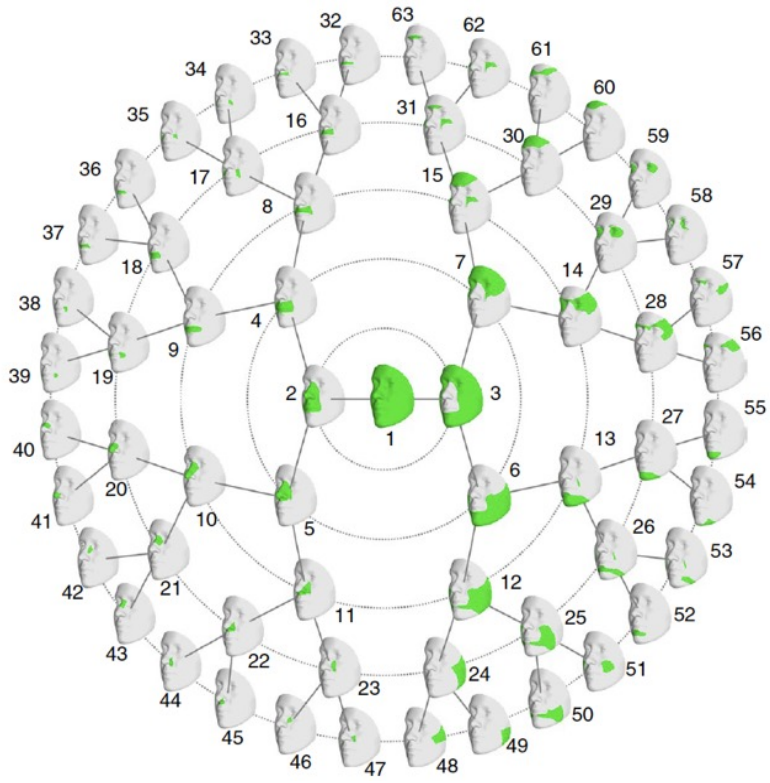
(f) Generation

Some New Results



On-going project: Connecting genome with human face





Thank You!