Modern Generative Models:

Restricted Boltzmann Machines

Jun Luo
02/2020

Based on presentation by Hung Chao
https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class22.pdf

RESTRICTED BOLTZMANN MACHINE

- Unsupervised Learning: use only the inputs $\mathbf{x}^{(t)}$ for learning
  - automatically extract meaningful features for data
  - Leverage the availability of unlabeled data
  - Can use negative log-likelihood to learn the underlying feature
- We will see 2 neural networks for unsupervised learning
  - **Restricted Boltzmann Machines**
  - Variational Autoencoders

GENERATIVE MODELS

- Given training data, we want to generate new samples from the same distribution

Training data ~ $p_{data}(x)$

Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Figure source: CIFAR-10 dataset (Krizhevsky and Hinton, 2009)

GENERATIVE MODELS

- Why generative models?
  - Realistic samples for artwork, super-resolution, colorization, etc.

  - Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
  - Training generative models can also enable inference of latent representation that can be useful as general features

Figure source: Internet

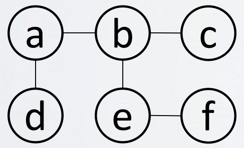
RESTRICTED BOLTZMANN MACHINE

- Many interesting theoretical results about undirected models depends on the assumption that $\forall x, \tilde{p}(x) > 0$ A convenient way to enforce this condition is to use an energy-based model where

$$\tilde{p}(x) = \exp(-E(x))$$

  - Normalized probability $\longrightarrow$ $p(x) = \frac{1}{Z}\tilde{p}(x)$
- E(x) is known as the *energy function*
- Any distribution of this form is an example of a Boltzmann distribution. For this reason, many energy-based models are called Boltzmann machines.

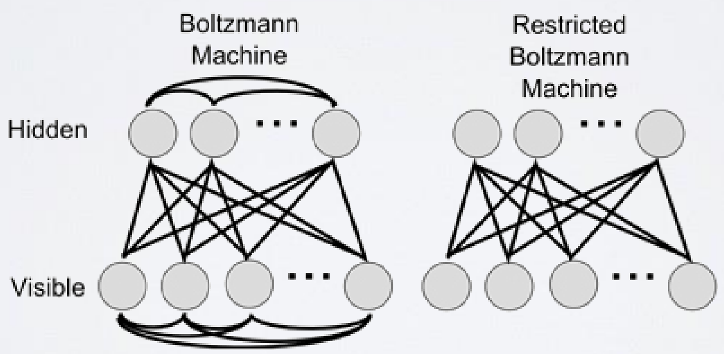
e.g.



E(a, b, c, d, e, f) can be written as $E_{a,b}$(a,b) + $E_{b,c}$(b,c) + $E_{a,d}$(a,d) + $E_{b,e}$(b,e) + $E_{e,f}$(e,f)

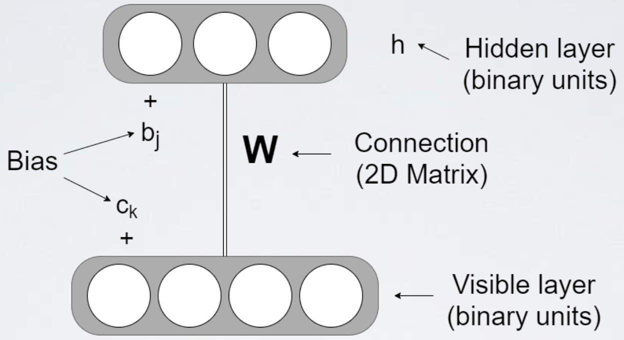
RESTRICTED BOLTZMANN MACHINE

- **Restricted Boltzmann machines (RBMs)** are undirected probabilistic graphical models containing a layer of observable variables and a single layer of latent variables
- RBM is a bipartite graph, with no connections permitted between any variables in the observed layer or between any units in the latent layer



RESTRICTED BOLTZMANN MACHINE

Structure:



Energy function: $E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W} \mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$

$$= -\sum_j \sum_k W_{j,k} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution: $p(\mathbf{x}, \mathbf{h}) = \exp(-E(\mathbf{x}, \mathbf{h}))/Z$

partition function
(intractable)

RESTRICTED BOLTZMANN MACHINE

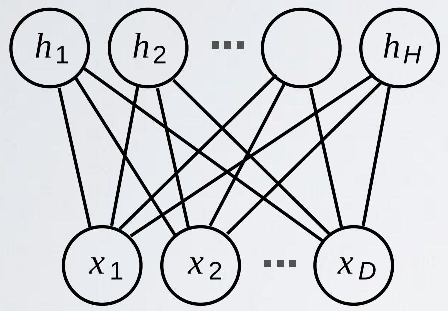
Markov network view

$$\begin{aligned} p(\mathbf{x}, \mathbf{h}) &= \exp(-E(\mathbf{x}, \mathbf{h}))/Z \\ &= \exp\left(\mathbf{h}^\top \mathbf{W} \mathbf{x} + \mathbf{c}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}\right)/Z \\ &= \underbrace{\exp\left(\mathbf{h}^\top \mathbf{W} \mathbf{x}\right) \exp\left(\mathbf{c}^\top \mathbf{x}\right) \exp\left(\mathbf{b}^\top \mathbf{h}\right)}_{\text{Factors}}/Z \end{aligned}$$

The notation based on an energy function is simply an alternative to the representation as the product of factors

RESTRICTED BOLTZMANN MACHINE

Markov network view



pair-wise factors

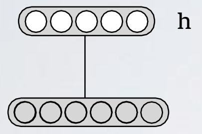
$$p(\mathbf{x}, \mathbf{h}) = \frac{1}{Z} \prod_j \prod_k \exp\left(W_{j,k} h_j x_k\right)$$

$$\left.\begin{aligned} &\prod_k \exp\left(c_k x_k\right) \\ &\prod_j \exp\left(b_j h_j\right) \end{aligned}\right\} \text{Unary Factors}$$

The scalar visualization is more informative of the structure within the vectors

INFERENCE

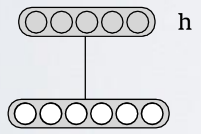
Conditional Distribution:



$$p(\mathbf{h}|\mathbf{x}) = \prod_j p\left(h_j|\mathbf{x}\right)$$

$$p\left(h_j = 1|\mathbf{x}\right) = \frac{1}{1 + \exp\left(-\left(b_j + \mathbf{W}_j \cdot \mathbf{x}\right)\right)}$$

$$= \operatorname{sigm}\left(b_j + \mathbf{W}_j \cdot \mathbf{x}\right)$$



$$p(\mathbf{x}|\mathbf{h}) = \prod_k p\left(x_k|\mathbf{h}\right)$$

$$p\left(x_k = 1|\mathbf{h}\right) = \frac{1}{1 + \exp\left(-\left(c_k + \mathbf{h}^\top \mathbf{W} \cdot k\right)\right)}$$

$$= \operatorname{sigm}\left(c_k + \mathbf{h}^\top \mathbf{W}_{\cdot k}\right)$$

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{x}) &= p(\mathbf{x},\mathbf{h}) / \sum_{\mathbf{h}'} p\left(\mathbf{x},\mathbf{h}'\right) \\
&= \frac{\exp\left(\mathbf{h}^\top \mathbf{W}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}\right)/Z}{\sum_{\mathbf{h}'\in\{0,1\}^H} \exp\left(\mathbf{h}'^\top \mathbf{W}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}'\right)/Z} \\
&= \frac{\exp\left(\sum_j h_j \mathbf{W}_j.\mathbf{x} + b_j h_j\right)}{\sum_{h'_1\in\{0,1\}} \cdots \sum_{h'_H\in\{0,1\}} \exp\left(\sum_j h'_j \mathbf{W}_{j.}\mathbf{x}_{+b_j h'_j}\right)}
\end{aligned}
$$

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{x}) &= p(\mathbf{x},\mathbf{h}) / \sum_{\mathbf{h}'} p\left(\mathbf{x},\mathbf{h}'\right) \\
&= \frac{\exp\left(\mathbf{h}^\top \mathbf{W}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}\right)/Z}{\sum_{\mathbf{h}'\in\{0,1\}^H} \exp\left(\mathbf{h}'^\top \mathbf{W}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}'\right)/Z} \\
&= \frac{\exp\left(\sum_j h_j \mathbf{W}_j.\mathbf{x} + b_j h_j\right)}{\sum_{h'_1\in\{0,1\}} \cdots \sum_{h'_H\in\{0,1\}} \exp\left(\sum_j h'_j \mathbf{W}_{j.}\mathbf{x}_{+b_j h'_j}\right)}
\end{aligned}
$$

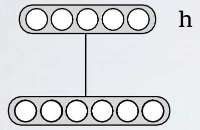
$$
\begin{aligned}
&= \frac{\prod_j \exp\left(h_j \mathbf{W}_j \cdot \mathbf{x} + b_j h_j\right)}{\sum_{h'_1\in\{0,1\}} \cdots \sum_{h'_H\in\{0,1\}} \prod_j \exp\left(h'_j \mathbf{W}_{j.}\mathbf{x}_{+b_j h'_j}\right)} \\
&= \frac{\prod_j \exp\left(h_j \mathbf{W}_j \cdot \mathbf{x} + b_j h_j\right)}{\left(\sum_{h'_1\in\{0,1\}} \exp\left(h'_1 \mathbf{W}_1 \cdot \mathbf{x} + b_1 h'_1\right)\right) \dots \left(\sum_{h'_H\in\{0,1\}} \exp\left(h'_H \mathbf{W}_H \cdot \mathbf{x} + b_H h'_H\right)\right)}
\end{aligned}
$$

$$
\begin{aligned}
p(\mathbf{h}|\mathbf{x}) &= p(\mathbf{x},\mathbf{h}) / \sum_{\mathbf{h}'} p(\mathbf{x},\mathbf{h}') \\
&= \frac{\exp\left(\mathbf{h}^\top \mathbf{W}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}\right)/Z}{\sum_{\mathbf{h}' \in \{0,1\}^H} \exp\left(\mathbf{h}'^\top \mathbf{W}\mathbf{x} + \mathbf{e}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}'\right)/Z} \\
&= \frac{\exp\left(\sum_j h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j\right)}{\sum_{h'_1 \in \{0,1\}} \cdots \sum_{h'_H \in \{0,1\}} \exp\left(\sum_j h'_j \mathbf{W}_{j\cdot \mathbf{x} + b_j h'_j}\right)} \\
&= \frac{\prod_j \exp\left(h_j \mathbf{W}_j \cdot \mathbf{x} + b_j h_j\right)}{\sum_{h'_1 \in \{0,1\}} \cdots \sum_{h'_H \in \{0,1\}} \prod_j \exp\left(h'_j \mathbf{W}_{j\cdot \mathbf{x} + b_j h'_j}\right)} \\
&= \frac{\prod_j \exp\left(h_j \mathbf{W}_j \cdot \mathbf{x} + b_j h_j\right)}{\left(\sum_{h'_1 \in \{0,1\}} \exp\left(h'_1 \mathbf{W}_1 \cdot \mathbf{x} + b_1 h'_1\right)\right) \ldots \left(\sum_{h'_H \in \{0,1\}} \exp\left(h'_H \mathbf{W}_H \cdot \mathbf{x} + b_H h'_H\right)\right)} \\
&= \frac{\prod_j \exp\left(h_j \mathbf{W}_j \cdot \mathbf{x} + b_j h_j\right)}{\prod_j \left(\sum_{h'_j \in \{0,1\}} \exp\left(h'_j \mathbf{W}_{j\cdot \mathbf{x} + b_j h'_j}\right)\right)} \\
&= \frac{\prod_j \exp\left(h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j\right)}{\prod_j \left(1 + \exp\left(b_j + \mathbf{W}_j \cdot \mathbf{x}\right)\right)} \\
&= \prod_j \frac{\exp\left(h_j \mathbf{W}_j \cdot \mathbf{x} + b_j h_j\right)}{1 + \exp\left(b_j + \mathbf{W}_j \cdot \mathbf{x}\right)} \\
&= \prod_j p\left(h_j | \mathbf{x}\right)
\end{aligned}
$$

$$
\begin{aligned}
p\left(h_j = 1 | \mathbf{x}\right) &= \frac{\exp\left(b_j + \mathbf{W}_j \cdot \mathbf{x}\right)}{1 + \exp\left(b_j + \mathbf{W}_j \cdot \mathbf{x}\right)} \\
&= \frac{1}{1 + \exp\left(-b_j - \mathbf{W}_j \cdot \mathbf{x}\right)} \\
&= \operatorname{sigm}\left(b_j + \mathbf{W}_{j\cdot}\mathbf{x}\right)
\end{aligned}
$$

FREE ENERGY

What about $p(x)$?



$$p(\mathbf{x}) = \sum_{\mathbf{h}\in\{0,1\}^H} p(\mathbf{x},\mathbf{h}) = \sum_{\mathbf{h}\in\{0,1\}^H} \exp(-E(\mathbf{x},\mathbf{h}))/Z$$

$$= \exp\left(\mathbf{c}^\top\mathbf{x} + \sum_{j=1}^{H} \log\left(1+\exp\left(b_j + \mathbf{W}_{j\cdot}\mathbf{x}\right)\right)\right)/Z$$

$$= \exp(-F(\mathbf{x}))/Z$$

Free Energy

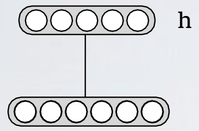
$$p(\mathrm{x}) = \sum_{\mathrm{h}\in\{0,1\}^H} \exp\left(\mathrm{h}^\top\mathbf{W}\mathrm{x} + \mathbf{c}^\top\mathbf{x} + \mathbf{b}^\top\mathbf{h}\right)/Z$$

$$= \exp\left(\mathbf{c}^\top\mathbf{x}\right) \sum_{h_1\in\{0,1\}} \cdots \sum_{h_H\in\{0,1\}} \exp\left(\sum_j h_j\mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j\right)/Z$$

$$
\begin{aligned}
p(\mathrm{x}) &= \sum_{\mathrm{h}\in\{0,1\}^H} \exp\left(\mathrm{h}^\top \mathbf{W}\mathrm{x} + \mathbf{c}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \sum_{h_1\in\{0,1\}} \cdots \sum_{h_H\in\{0,1\}} \exp\left(\sum_j h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \left(\sum_{h_1\in\{0,1\}} \exp\left(h_1 \mathbf{W}_1.\mathbf{x} + b_1 h_1\right)\right) \ldots \left(\sum_{h_H\in\{0,1\}} \exp\left(h_H \mathbf{W}_H \cdot \mathbf{x} + b_H h_H\right)\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \left(1 + \exp\left(b_1 + \mathbf{W}_1 \cdot \mathbf{x}\right)\right) \ldots \left(1 + \exp\left(b_H + \mathbf{W}_H \cdot \mathbf{x}\right)\right)/Z
\end{aligned}
$$

$$
\begin{aligned}
p(\mathrm{x}) &= \sum_{\mathrm{h}\in\{0,1\}^H} \exp\left(\mathrm{h}^\top \mathbf{W}\mathrm{x} + \mathbf{c}^\top \mathbf{x} + \mathbf{b}^\top \mathbf{h}\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \sum_{h_1\in\{0,1\}} \cdots \sum_{h_H\in\{0,1\}} \exp\left(\sum_j h_j \mathbf{W}_{j\cdot}\mathbf{x} + b_j h_j\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \left(\sum_{h_1\in\{0,1\}} \exp\left(h_1 \mathbf{W}_1.\mathbf{x} + b_1 h_1\right)\right) \ldots \left(\sum_{h_H\in\{0,1\}} \exp\left(h_H \mathbf{W}_H \cdot \mathbf{x} + b_H h_H\right)\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \left(1 + \exp\left(b_1 + \mathbf{W}_1 \cdot \mathbf{x}\right)\right) \ldots \left(1 + \exp\left(b_H + \mathbf{W}_H \cdot \mathbf{x}\right)\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x}\right) \exp\left(\log\left(1 + \exp\left(b_1 + \mathbf{W}_1 \cdot \mathbf{x}\right)\right)\right) \ldots \exp\left(\log\left(1 + \exp\left(b_H + \mathbf{W}_H \cdot \mathbf{x}\right)\right)\right)/Z \\
&= \exp\left(\mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{H} \log\left(1 + \exp\left(b_j + \mathbf{W}_{j\cdot}\mathbf{x}\right)\right)\right)/Z
\end{aligned}
$$

FREE ENERGY



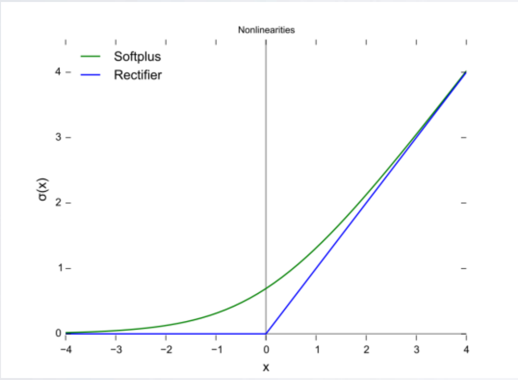
$$p(\mathbf{x}) = \exp\left(\mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{H} \log\left(1 + \exp\left(b_j + \mathbf{W}_{j\cdot}\mathbf{x}\right)\right)\right) / Z$$

$$= \exp\left(\mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{H} \operatorname{softplus}\left(b_j + \mathbf{W}_{j\cdot}\mathbf{x}\right)\right) / Z$$

Bias of each feature

Bias the probability of x

Feature expected in x



Training

Training objective

To train an RBM, we minimize the average negative log-likelihood (NLL)

$$\frac{1}{T}\sum_t l\left(f\left(\mathbf{x}^{(t)}\right)\right) = \frac{1}{T}\sum_t -\log p\left(\mathbf{x}^{(t)}\right)$$

$$\log p\left(x^{(t)}\right) = \log\left(\sum_h p\left(x^{(t)}, h\right)\right)$$

$$= \log\left(\sum_h \frac{\exp\left(-E\left(x^{(t)}, h\right)\right)}{Z}\right)$$

$$= \log\left(\sum_h \exp\left(-E\left(x^{(t)}, h\right)\right)\right) - \log Z$$

We'd like to proceed by stochastic gradient descent

$$\frac{\partial -\log p\left(\mathbf{x}^{(t)}\right)}{\partial\theta} = \underbrace{\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{x}^{(t)}, \mathbf{h}\right)}{\partial\theta} \middle| \mathbf{x}^{(t)}\right]}_{\text{Positive Phase}} - \underbrace{\mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial\theta}\right]}_{\text{Negative Phase}}$$

Training

Training objective

To train an RBM, we minimize the average negative log-likelihood (NLL)

$$\frac{1}{T}\sum_t l\left(f\left(\mathbf{x}^{(t)}\right)\right) = \frac{1}{T}\sum_t -\log p\left(\mathbf{x}^{(t)}\right)$$

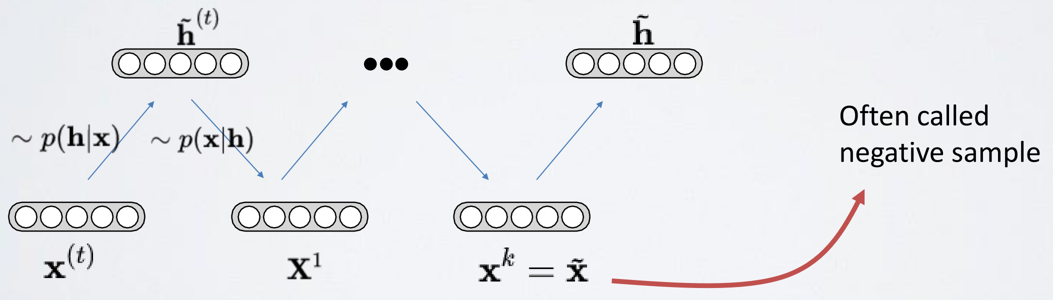
We'd like to proceed by stochastic gradient descent

$$\frac{\partial -\log p\left(\mathbf{x}^{(t)}\right)}{\partial\theta} = \underbrace{\mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{x}^{(t)},\mathbf{h}\right)}{\partial\theta}\middle|\mathbf{x}^{(t)}\right]}_{\text{Positive Phase}} - \underbrace{\overset{\text{Hard to compute}}{\boxed{\mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right]}}}_{\text{Negative Phase}}$$

Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

Idea:
1. obtain the point $\tilde{\mathbf{X}}$ by Gibbs sampling
2. start sampling chain at $\mathbf{x}^{(t)}$
3. replace the expectation by a point estimate at $\tilde{\mathbf{X}}$



Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

Replace the expectation by a point estimate

$$\frac{\partial - \log p\left(\mathbf{x}^{(t)}\right)}{\partial\theta} = \mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{x}^{(t)},\mathbf{h}\right)}{\partial\theta}|\mathbf{x}^{(t)}\right] - \mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right]$$

$$\mathrm{E}_{\mathrm{h}}\left[\frac{\partial E\left(\mathrm{x}^{(t)},\mathrm{h}\right)}{\partial\theta}|\mathrm{x}^{(t)}\right] \approx \frac{\partial E\left(\mathrm{x}^{(t)},\tilde{\mathrm{h}}^{(t)}\right)}{\partial\theta}$$

$$\mathrm{E}_{\mathrm{x},\mathrm{h}}\left[\frac{\partial E(\mathrm{x},\mathrm{h})}{\partial\theta}\right] \approx \frac{\partial E(\tilde{\mathrm{x}},\tilde{\mathrm{h}})}{\partial\theta}$$

Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

Replace the expectation by a point estimate

$$\frac{\partial - \log p\left(\mathbf{x}^{(t)}\right)}{\partial\theta} = \mathrm{E}_{\mathbf{h}}\left[\frac{\partial E\left(\mathbf{x}^{(t)},\mathbf{h}\right)}{\partial\theta}|\mathbf{x}^{(t)}\right] - \mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial\theta}\right]$$

$$\mathrm{E}_{\mathrm{h}}\left[\frac{\partial E\left(\mathrm{x}^{(t)},\mathrm{h}\right)}{\partial\theta}|\mathrm{x}^{(t)}\right] \approx \frac{\partial E\left(\mathrm{x}^{(t)},\tilde{\mathrm{h}}^{(t)}\right)}{\partial\theta}$$

Probability goes up

$$\mathrm{E}_{\mathrm{x},\mathrm{h}}\left[\frac{\partial E(\mathrm{x},\mathrm{h})}{\partial\theta}\right] \approx \frac{\partial E(\tilde{\mathrm{x}},\tilde{\mathrm{h}})}{\partial\theta}$$

Probability goes down

Parameter Update

Derivation of $\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \theta}$ for $\theta = W_{jk}$

$$\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial W_{jk}} = \frac{\partial}{\partial W_{jk}}\left(-\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j\right)$$
$$= -\frac{\partial}{\partial W_{jk}} \sum_{jk} W_{jk} h_j x_k$$
$$= -h_j x_k$$

$$\nabla_{\mathbf{W}} E(\mathbf{x},\mathbf{h}) = -\mathbf{h}\mathbf{x}^{\top}$$

Parameter Update

Derivation of $\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \theta}|\mathbf{x}\right]$ for $\theta = W_{jk}$

$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial W_{jk}}|\mathbf{x}\right] = \mathbb{E}_{\mathbf{h}}\left[-h_j x_k|\mathbf{x}\right] = \sum_{h_j \in \{0,1\}} -h_j x_k p\left(h_j|\mathbf{x}\right)$$
$$= -x_k p\left(h_j = 1|\mathbf{x}\right)$$

Parameter Update

Derivation of $\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial \theta}|\mathbf{x}\right]$ for $\theta = W_{jk}$

$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x},\mathbf{h})}{\partial W_{jk}}|\mathbf{x}\right] = \mathbb{E}_{\mathbf{h}}\left[-h_j x_k|\mathbf{x}\right] = \sum_{h_j\in\{0,1\}} -h_j x_k p\left(h_j|\mathbf{x}\right)$$
$$= -x_k p\left(h_j = 1|\mathbf{x}\right)$$

If we define:

$$\mathbf{h}(\mathbf{x}) = \begin{pmatrix} p\left(h_1 = 1|\mathbf{x}\right) \\ \cdots \\ p\left(h_H = 1|\mathbf{x}\right) \end{pmatrix}$$
$$= \mathrm{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})$$

Then,

$$\mathrm{E}_{\mathrm{h}}\left[\nabla_{\mathrm{W}} E(\mathrm{x},\mathrm{h})|\mathrm{x}\right] = -\mathrm{h}(\mathrm{x})\mathrm{x}^{\top}$$

Parameter Update

Update of W

Given $\mathbf{x}^{(t)}$ and $\tilde{\mathbf{X}}$, the learning rule of $\theta = \mathbf{W}$ becomes:

$$\begin{aligned}
\mathbf{W} &\Longleftarrow \mathbf{W} - \alpha\left(\nabla_{\mathbf{W}} - \log p\left(\mathbf{x}^{(t)}\right)\right) \\
&\Longleftarrow \mathbf{W} - \alpha\left(\mathrm{E}_{\mathbf{h}}\left[\nabla_{\mathbf{W}} E\left(\mathbf{x}^{(t)},\mathbf{h}\right)|\mathbf{x}^{(t)}\right] - \mathrm{E}_{\mathbf{x},\mathbf{h}}\left[\nabla_{\mathbf{W}} E(\mathbf{x},\mathbf{h})\right]\right) \\
&\Longleftarrow \mathbf{W} - \alpha\left(\mathrm{E}_{\mathbf{h}}\left[\nabla_{\mathbf{W}} E\left(\mathbf{x}^{(t)},\mathbf{h}\right)|\mathbf{x}^{(t)}\right] - \mathrm{E}_{\mathbf{h}}\left[\nabla_{\mathbf{W}} E(\tilde{\mathbf{x}},\mathbf{h})|\tilde{\mathbf{x}}\right]\right) \\
&\Longleftarrow \mathbf{W} + \alpha\left(\mathbf{h}\left(\mathbf{x}^{(t)}\right)\mathbf{x}^{(t)} - \mathbf{h}(\tilde{\mathbf{x}})\tilde{\mathbf{x}}^{\top}\right)
\end{aligned}$$

CD-K: PSEUDOCODE

Contrastive Divergence:

1. For each training sample $\mathbf{x}^{(t)}$
   i. Generate a negative sample $\tilde{\mathbf{x}}$ using k steps of Gibbs sampling, starting at $\mathbf{x}^{(t)}$
   ii. Update parameters:

$$\mathbf{W} \Longleftarrow \mathbf{W} + \alpha \left(\mathbf{h}\left(\mathbf{x}^{(t)}\right) \mathbf{x}^{(t)^{\top}} - \mathbf{h}(\tilde{\mathbf{x}})\tilde{\mathbf{x}}^{\top}\right)$$
$$\mathbf{b} \Longleftarrow \mathbf{b} + \alpha \left(\mathbf{h}\left(\mathbf{x}^{(t)}\right) - \mathbf{h}(\tilde{\mathbf{x}})\right)$$
$$\mathbf{c} \Longleftarrow \mathbf{c} + \alpha \left(\mathbf{x}^{(t)} - \tilde{\mathbf{x}}\right)$$

2. Go back to 1. until stopping criteria is reached

Contrastive Divergence-k

Contrastive Divergence:

- CD-k: contrastive divergence with k iterations of Gibbs sampling
- In general, the bigger k is, the less biased the estimate of the gradient will be
- In practice, k=1 works well for pre-training

Software

- Sci-kit learn
  https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.BernoulliRBM.html
- Pydbm
  https://pypi.org/project/pydbm/
- Other self-implemented versions on github

References

[1] Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016. Chapter 20

[2] Hung Chao's CS 3750 year 2018 slides
https://people.cs.pitt.edu/~milos/courses/cs3750/lectures/class22.pdf

[3] Hugo Larochelle. Neural networks class series - Université de Sherbrooke

[4] Hinton, Geoffrey E. "Training products of experts by minimizing contrastive divergence." Neural computation 14.8 (2002): 1771-1800.

[5] Tieleman, Tijmen. "Training restricted Boltzmann machines using approximations to the likelihood gradient." Proceedings of the 25th international conference on Machine learning. 2008.

Modern Generative Models:

Variational Autoencoders

Jun Luo
02/2020

GENERATIVE MODELS

- Given training data, we want to generate new samples from the same distribution

Training data ~ $p_{data}(x)$ Generated samples ~ $p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

Figure source: CIFAR-10 dataset (Krizhevsky and Hinton, 2009)

GENERATIVE MODELS

- Why generative models?
  - Realistic samples for artwork, super-resolution, colorization, etc.

  - Generative models of time-series data can be used for simulation and planning (reinforcement learning applications!)
  - Training generative models can also enable inference of latent representation that can be useful as general features

Figure source: Internet

Autoencoders (Recap)

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

z usually smaller than **x** (dimensionality reduction)

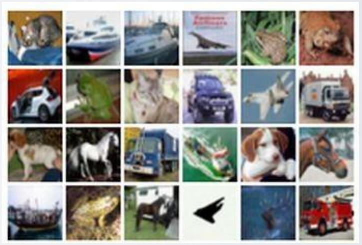
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

Features vector are generally shorter than input vector to extract meaningful features

Features z

Encoder

Input data x



Autoencoders (Recap)

How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself

Reconstructed data $\hat{x}$

Decoder

Features z

Encoder

Input data x

Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN

Autoencoders (Recap)

How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself

Reconstructed data $\hat{x}$

Decoder

Features z

Encoder

Input data x

Reconstructed data

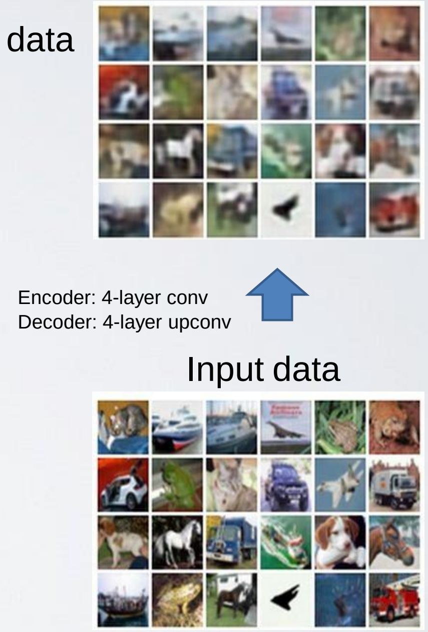
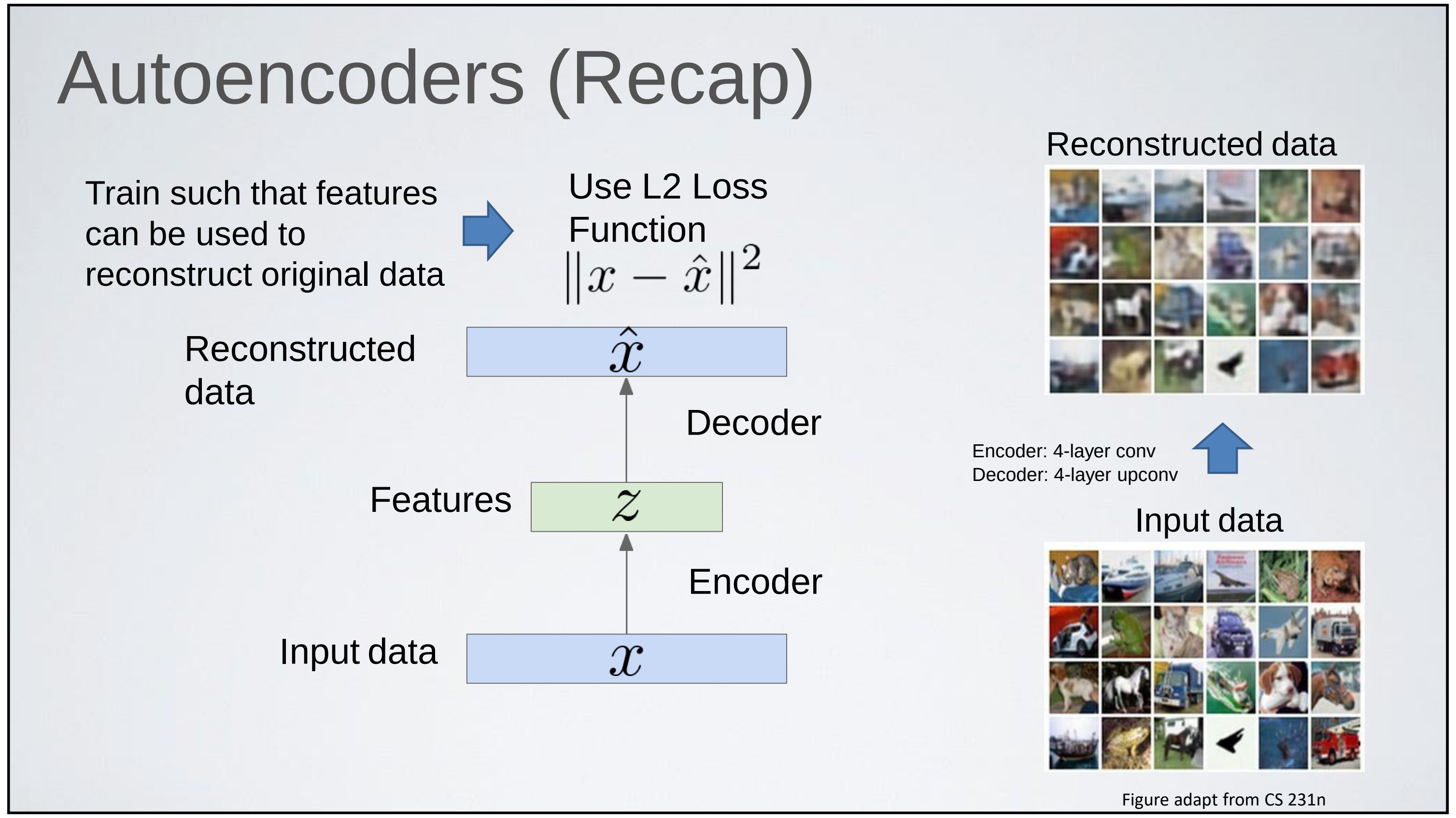
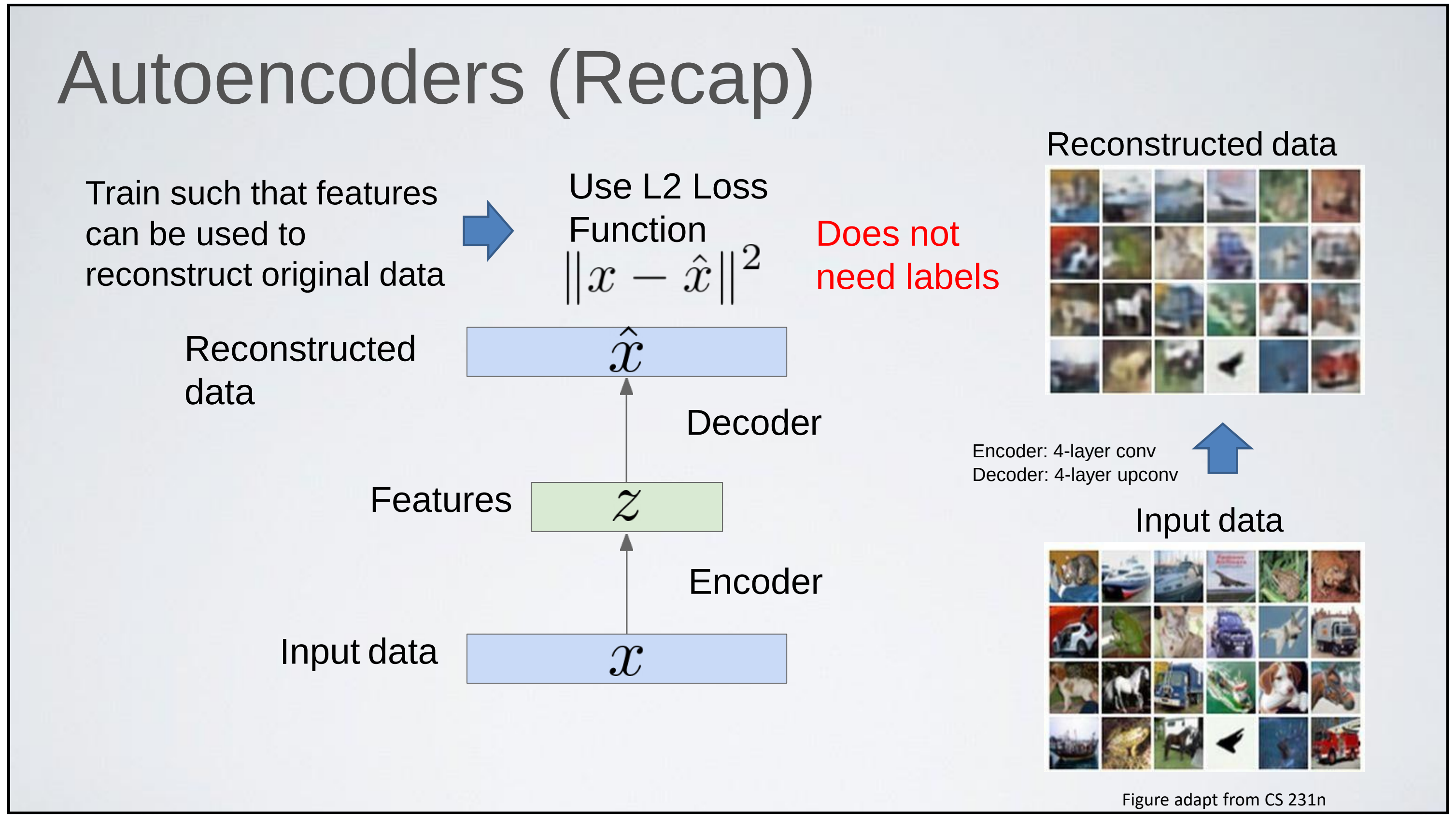


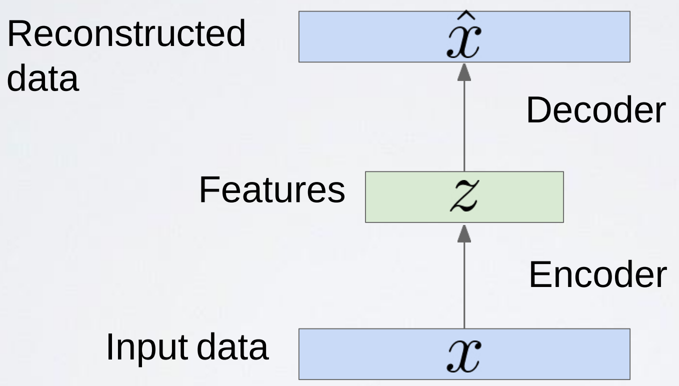
Figure adapt from CS 231n


Autoencoders (Recap)
Train such that features can be used to reconstruct original data
Use L2 Loss Function
$||x - \hat{x}||^2$
Reconstructed data
$\hat{x}$
Decoder
Features
z
Encoder
Input data
x
Reconstructed data
Encoder: 4-layer conv
Decoder: 4-layer upconv
Input data
Figure adapt from CS 231n


Autoencoders (Recap)
Train such that features can be used to reconstruct original data
Use L2 Loss Function
$||x - \hat{x}||^2$
Does not need labels
Reconstructed data
$\hat{x}$
Decoder
Features
z
Encoder
Input data
x
Reconstructed data
Encoder: 4-layer conv
Decoder: 4-layer upconv
Input data
Figure adapt from CS 231n

Autoencoders (Recap)

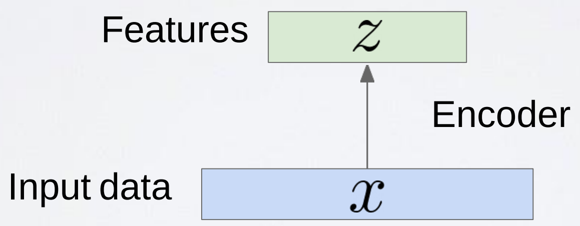
Encoder can be used to initialize a **supervised** model

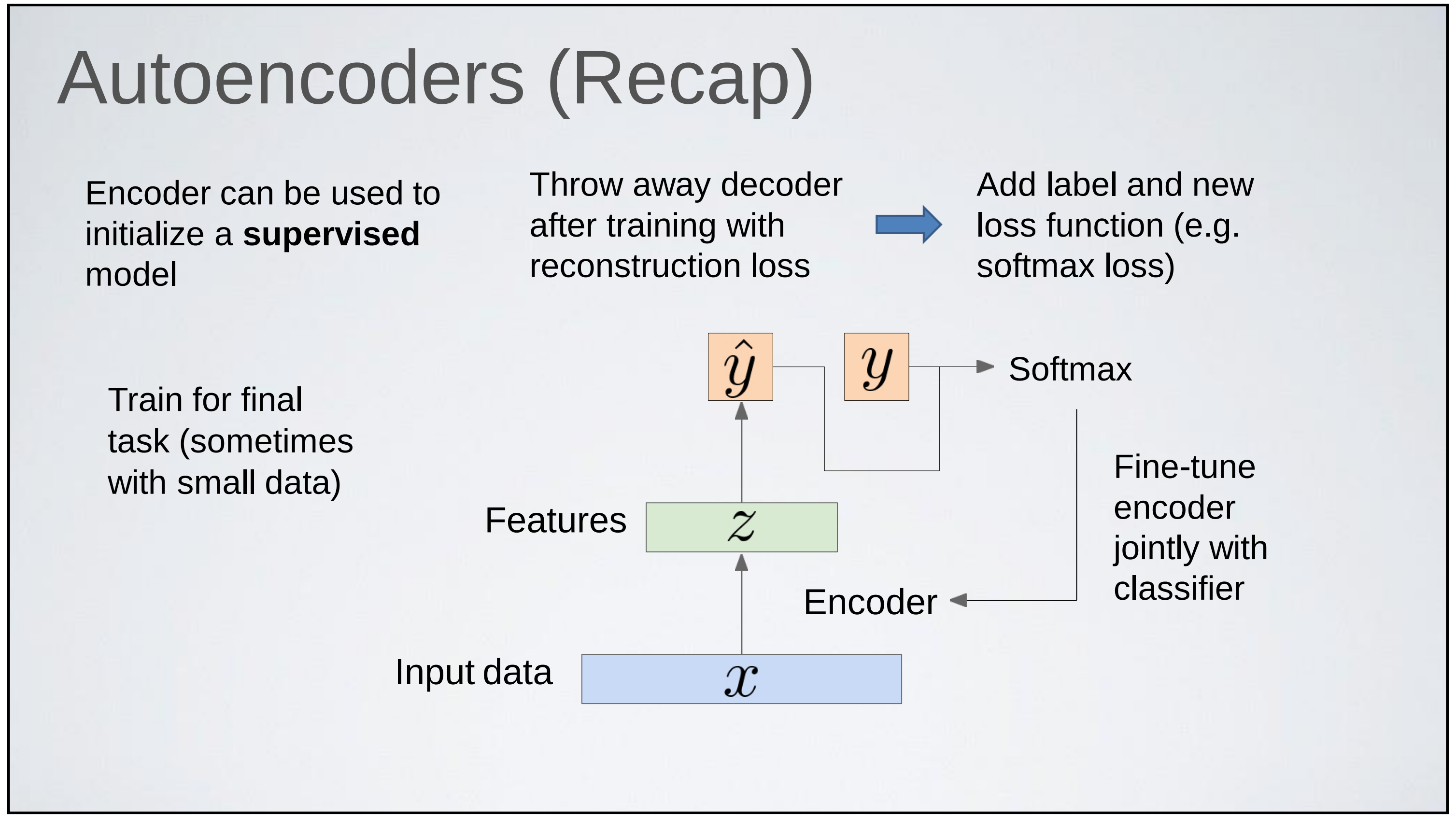


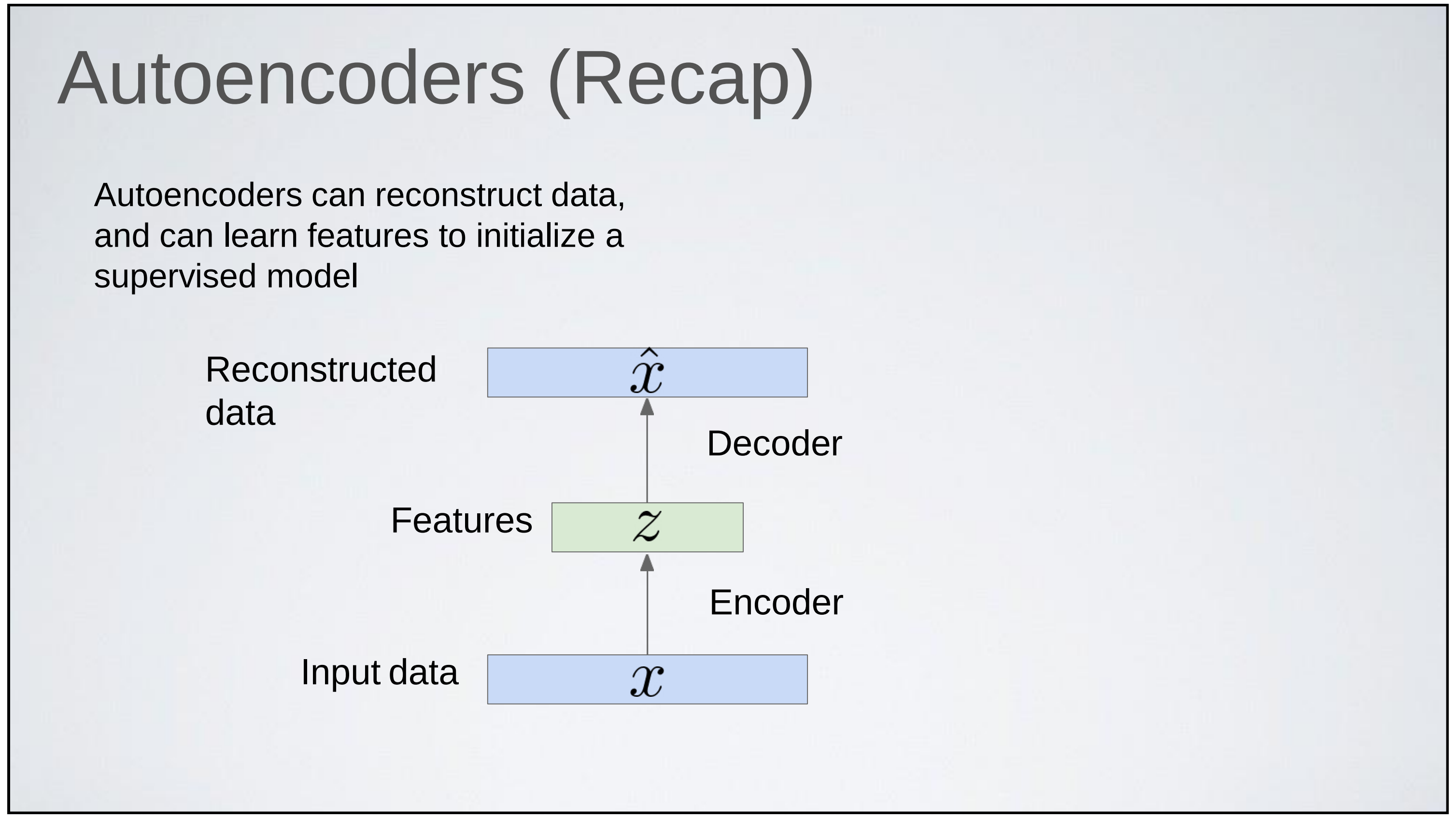
Autoencoders (Recap)

Encoder can be used to initialize a **supervised** model

Throw away decoder

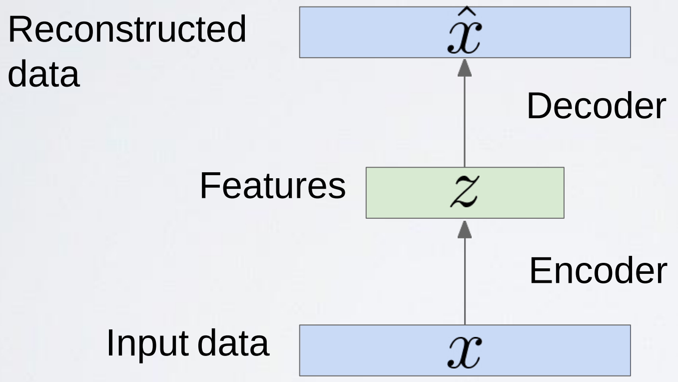



Autoencoders (Recap)
Encoder can be used to initialize a **supervised** model
Throw away decoder after training with reconstruction loss
Add label and new loss function (e.g. softmax loss)
Train for final task (sometimes with small data)
$\hat{y}$
y
Softmax
Fine-tune encoder jointly with classifier
Features
z
Encoder
Input data
x


Autoencoders (Recap)
Autoencoders can reconstruct data, and can learn features to initialize a supervised model
Reconstructed data
$\hat{x}$
Decoder
Features
z
Encoder
Input data
x

Autoencoders (Recap)

Autoencoders can reconstruct data, and can learn features to initialize a supervised model

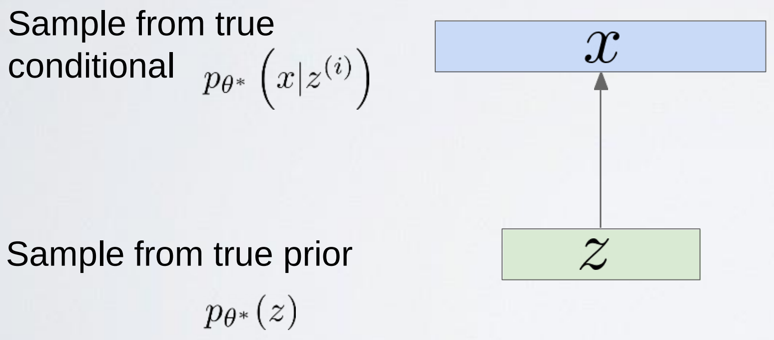


Features capture factors of variation in training data. Can we generate new images from an autoencoder?

Variational Autoencoders

Probabilistic spin on autoencoders - will let us sample from the model to generate data!

Assume training data $\left\{x^{(i)}\right\}_{i=1}^{N}$ is generated from underlying unobserved (latent) representation **z**



Intuition (remember from autoencoders!): **x** is an image; **z** is latent factors used to generate

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Sample from true conditional $p_{\theta^*}\left(x|z^{(i)}\right)$

x

Decoder network

z

Sample from true prior $p_{\theta^*}(z)$

Choose prior $p(z)$ to be simple, e.g. Gaussian. Reasonable for latent attributes, e.g. pose, how much smile.

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model.

How should we represent this model?

Sample from true conditional $p_{\theta^*}\left(x|z^{(i)}\right)$

x

Decoder network

z

Sample from true prior $p_{\theta^*}(z)$

Choose prior $p(z)$ to be simple, e.g. Gaussian.

Conditional $p(x|z)$ is complex (generates image) => represent with neural network

Variational Autoencoders

We want to estimate the true parameters θ^* of this generative model.

How to train the model?

Sample from true conditional $p_{\theta^*}\left(x|z^{(i)}\right)$

x

Decoder network

z

Sample from true prior $p_{\theta^*}(z)$

Data likelihood

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$$

Similar as Restricted Boltzman Machines, here, we maximize the data likelihood.

Variational Autoencoders

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$ ✔ ✔

Simple Gaussian Prior

Decoder neural network

Variational Autoencoders

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Intractable to compute $p(x|z)$ for every z

Variational Autoencoders

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

Because of intractable data likelihood

Variational Autoencoders

Data likelihood: $p_\theta(x) = \int p_\theta(z) p_\theta(x|z) dz$

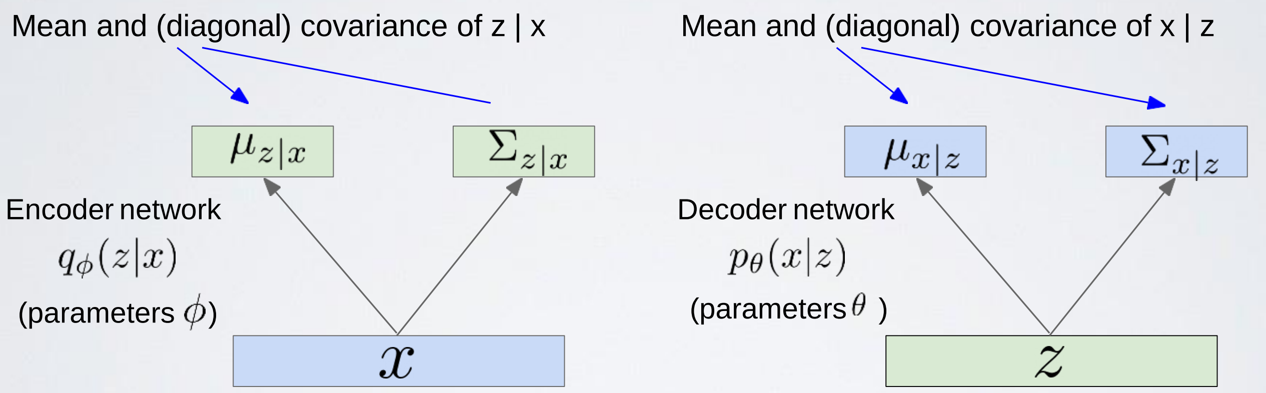
Posterior density also intractable: $p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / p_\theta(x)$

Solution: In addition to decoder network modeling $p_\theta(x|z)$, define additional encoder network $q_\phi(z|x)$ that approximates $p_\theta(z|x)$

Will see that this allows us to derive a lower bound on the data likelihood that is tractable, which we can optimize

Variational Autoencoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic



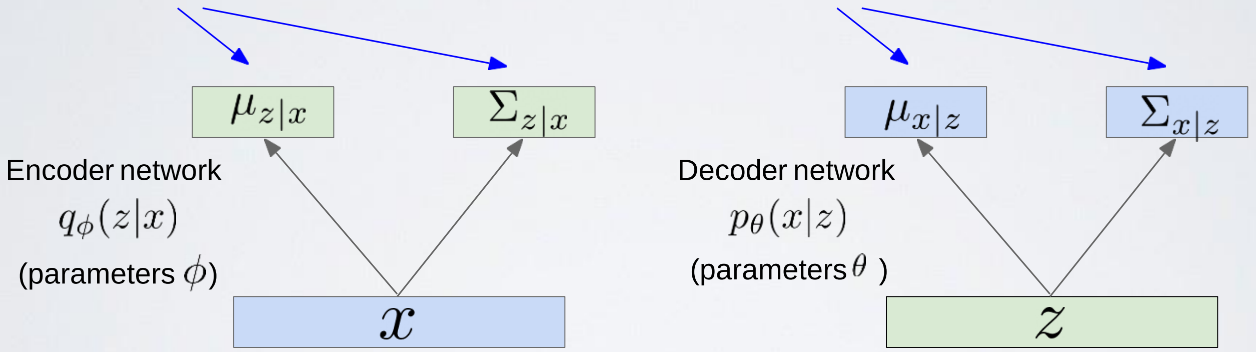
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

Since we're modeling probabilistic generation of data, encoder and decoder networks are probabilistic

Sample z|x from $z|x \sim \mathcal{N}\left(\mu_{z|x}, \Sigma_{z|x}\right)$

Sample x|z from $x|z \sim \mathcal{N}\left(\mu_{x|z}, \Sigma_{x|z}\right)$



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right]$ ($p_\theta\left(x^{(i)}\right)$ Does not depend on z)

Taking expectation over z
(using encoder network)
will be helpful later on

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right]$$ ($p_\theta\left(x^{(i)}\right)$ Does not depend on z)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Bayes' rule)

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right]$$ ($p_\theta\left(x^{(i)}\right)$ Does not depend on z)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Bayes' rule)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right]$$ (Multiply by 1)

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right] \quad \text{(} p_\theta\left(x^{(i)}\right) \text{ Does not depend on } z \text{)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Bayes' rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right] \quad \text{(Multiply by 1)}$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Logarithm)}$$

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right] \quad \text{(} p_\theta\left(x^{(i)}\right) \text{ Does not depend on } z \text{)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Bayes' rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right] \quad \text{(Multiply by 1)}$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Logarithm)}$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right) + D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta\left(z|x^{(i)}\right)\right)$$

The expectation over z lead to
nice KL Divergence form

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right] \quad \left(p_\theta\left(x^{(i)}\right) \text{ Does not depend on } z\right)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Bayes' rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right] \quad \text{(Multiply by 1)}$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Logarithm)}$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right) + D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta\left(z|x^{(i)}\right)\right)$$

$z = \mu + \sigma \odot \varepsilon$

Decoder network gives $p_\theta(x|z)$, can compute estimate of this term through sampling. (Sampling differentiable through reparam. trick, see paper.)

This KL term (between Gaussians for encoder and z prior) has nice closed-form solution!

$p_\theta(z|x)$ intractable (saw earlier), can't compute this KL term. But we know KL divergence always >= 0.

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi(z|x^{(i)})}\left[\log p_\theta\left(x^{(i)}\right)\right] \quad \left(p_\theta\left(x^{(i)}\right) \text{ Does not depend on } z\right)$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Bayes' rule)}$$

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right)p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right] \quad \text{(Multiply by 1)}$$

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right] \quad \text{(Logarithm)}$$

$$= \boxed{\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}\left(x^{(i)},\theta,\phi\right)}} + \underbrace{D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta\left(z|x^{(i)}\right)\right)}_{\geq 0}$$

Tractable lower bound which we can take gradient of and optimize! $p_\theta(x|z)$ differentiable, KL term differentiable)

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi\left(z|x^{(i)}\right)}\left[\log p_\theta\left(x^{(i)}\right)\right]$$ ($p_\theta\left(x^{(i)}\right)$ Does not depend on z)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right) p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Bayes' rule)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right) p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)} \frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right]$$ (Multiply by 1)

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Logarithm)

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right) \| p_\theta(z)\right)}_{\mathcal{L}\left(x^{(i)},\theta,\phi\right)} + \underbrace{D_{KL}\left(q_\phi\left(z|x^{(i)}\right) \| p_\theta\left(z|x^{(i)}\right)\right)}_{\geq 0}$$

$$\log p_\theta\left(x^{(i)}\right) \geq \mathcal{L}\left(x^{(i)},\theta,\phi\right)$$

Variational Lower Bound
"ELBO"

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}\left(x^{(i)},\theta,\phi\right)$$ Training: Maximize lower bound

Variational Autoencoders

Now let us see the (log) data likelihood again with encoder and decoder

$$\log p_\theta\left(x^{(i)}\right) = \mathbf{E}_{z\sim q_\phi\left(z|x^{(i)}\right)}\left[\log p_\theta\left(x^{(i)}\right)\right]$$ ($p_\theta\left(x^{(i)}\right)$ Does not depend on z)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right) p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Bayes' rule)

$$= \mathbf{E}_z\left[\log \frac{p_\theta\left(x^{(i)}|z\right) p_\theta(z)}{p_\theta\left(z|x^{(i)}\right)} \frac{q_\phi\left(z|x^{(i)}\right)}{q_\phi\left(z|x^{(i)}\right)}\right]$$ (Multiply by 1)

Make approximate posterior distribution close to prior

Reconstruct the input data

$$= \mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta(z)}\right] + \mathbf{E}_z\left[\log \frac{q_\phi\left(z|x^{(i)}\right)}{p_\theta\left(z|x^{(i)}\right)}\right]$$ (Logarithm)

$$= \underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right) \| p_\theta(z)\right)}_{\mathcal{L}\left(x^{(i)},\theta,\phi\right)} + \underbrace{D_{KL}\left(q_\phi\left(z|x^{(i)}\right) \| p_\theta\left(z|x^{(i)}\right)\right)}_{\geq 0}$$

$$\log p_\theta\left(x^{(i)}\right) \geq \mathcal{L}\left(x^{(i)},\theta,\phi\right)$$

Variational Lower Bound
"ELBO"

$$\theta^*, \phi^* = \arg\max_{\theta,\phi} \sum_{i=1}^{N} \mathcal{L}\left(x^{(i)},\theta,\phi\right)$$ Training: Maximize lower bound

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

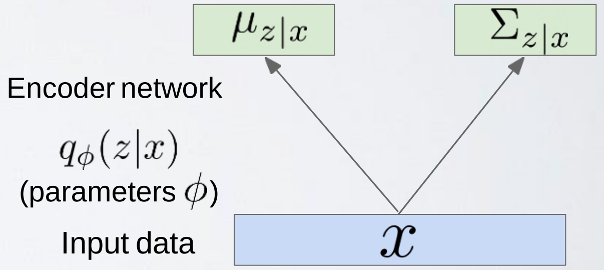
$$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Let’s look at computing the bound (forward pass) for a given minibatch of input

Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

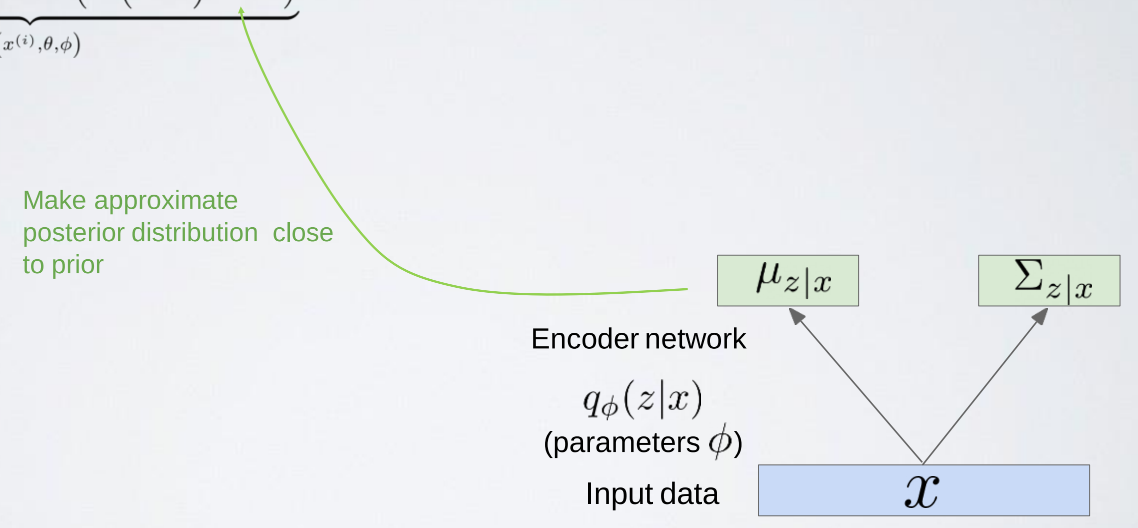
$$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$



Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

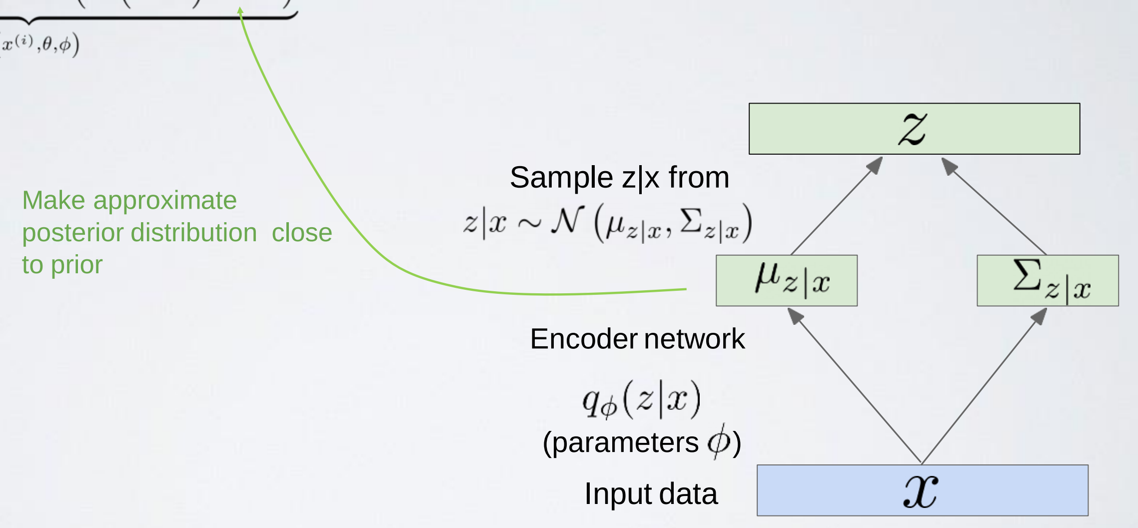
$$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

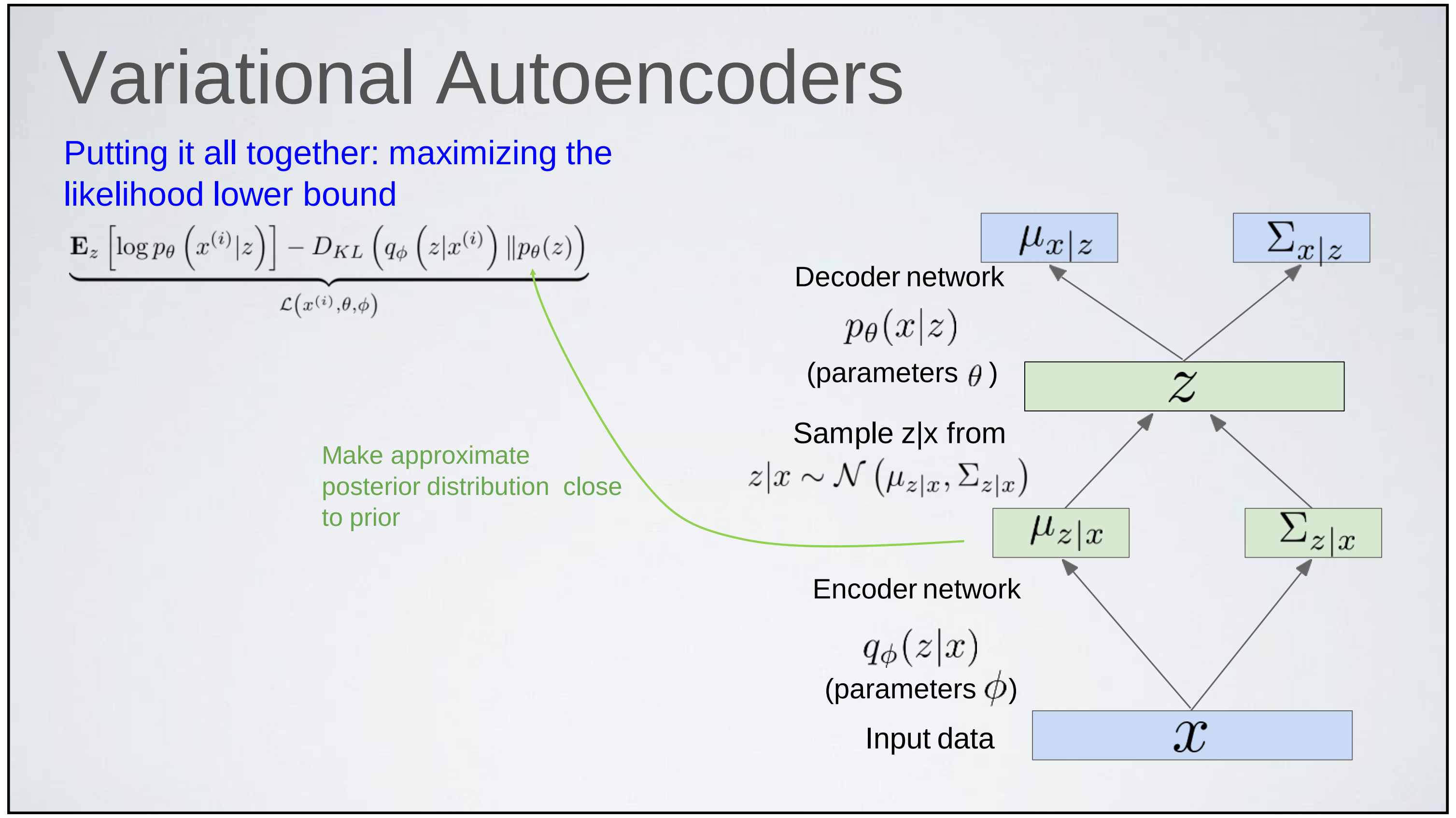


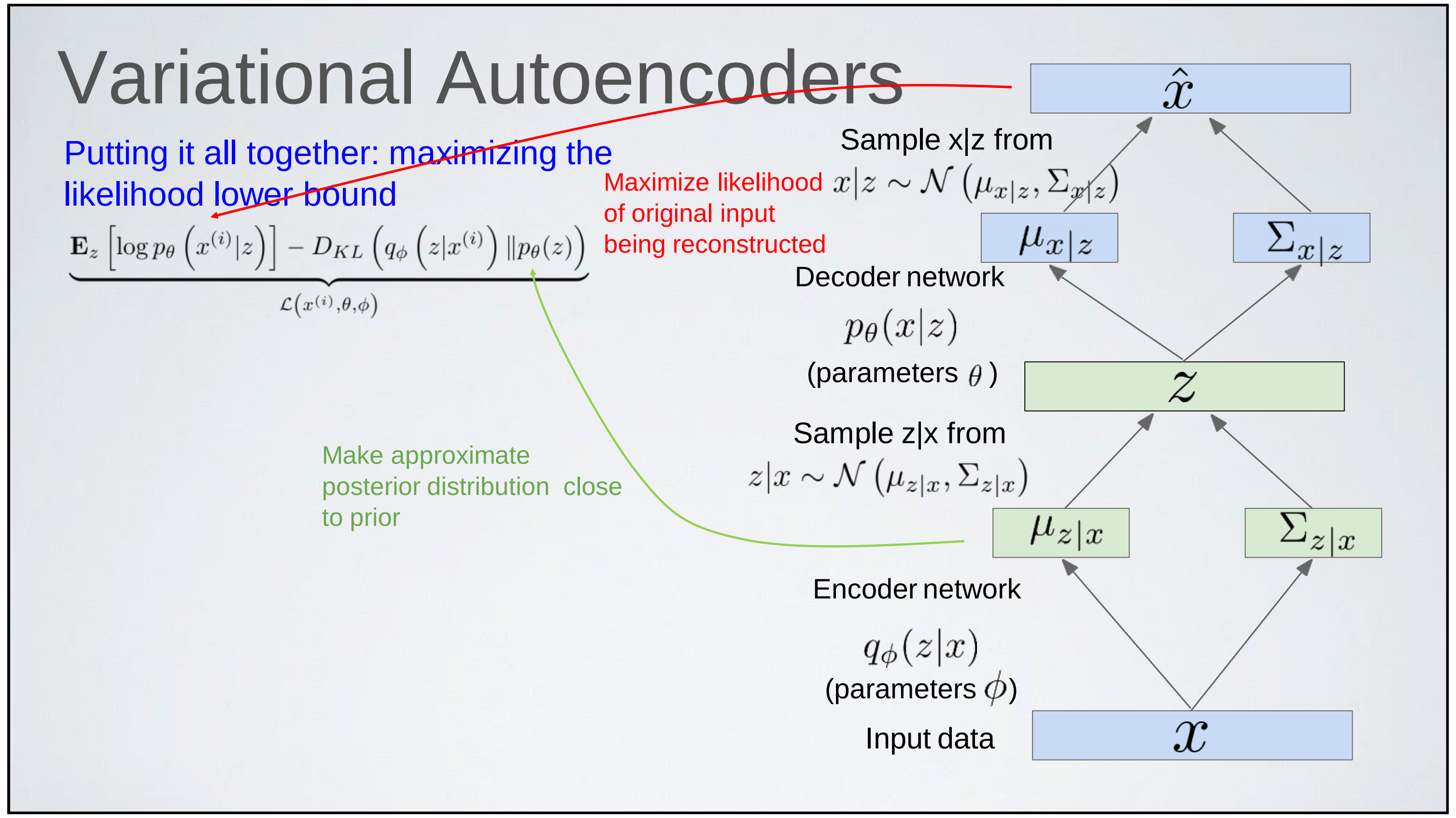
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$




Variational Autoencoders
Putting it all together: maximizing the likelihood lower bound
$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$
$\mu_{x|z}$
$\Sigma_{x|z}$
Decoder network
$p_\theta(x|z)$
(parameters θ)
z
Sample z|x from
$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
Make approximate posterior distribution close to prior
$\mu_{z|x}$
$\Sigma_{z|x}$
Encoder network
$q_\phi(z|x)$
(parameters ϕ)
Input data
x


Variational Autoencoders
$\hat{x}$
Putting it all together: maximizing the likelihood lower bound
Sample x|z from
$x|z \sim \mathcal{N}(\mu_{x|z}, \Sigma_{x|z})$
Maximize likelihood of original input being reconstructed
$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$
$\mu_{x|z}$
$\Sigma_{x|z}$
Decoder network
$p_\theta(x|z)$
(parameters θ)
z
Sample z|x from
$z|x \sim \mathcal{N}(\mu_{z|x}, \Sigma_{z|x})$
Make approximate posterior distribution close to prior
$\mu_{z|x}$
$\Sigma_{z|x}$
Encoder network
$q_\phi(z|x)$
(parameters ϕ)
Input data
x

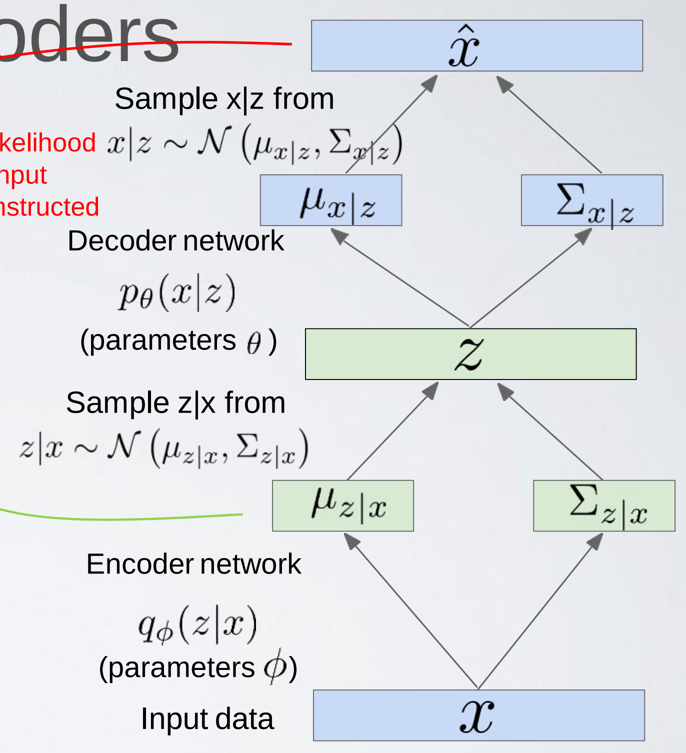
Variational Autoencoders

Putting it all together: maximizing the likelihood lower bound

$$\underbrace{\mathbf{E}_z\left[\log p_\theta\left(x^{(i)}|z\right)\right] - D_{KL}\left(q_\phi\left(z|x^{(i)}\right)\|p_\theta(z)\right)}_{\mathcal{L}(x^{(i)},\theta,\phi)}$$

Maximize likelihood of original input being reconstructed

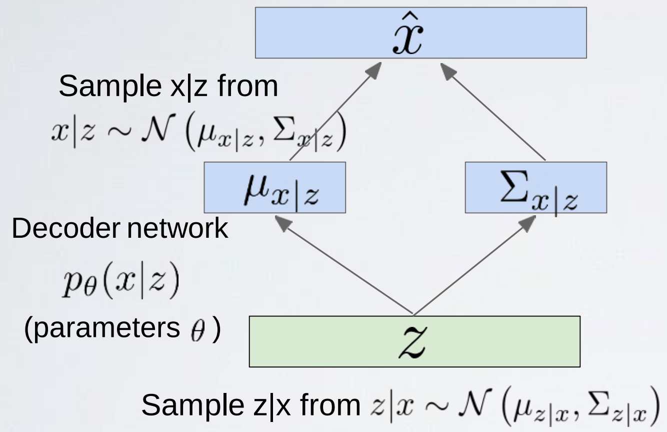
Make approximate posterior distribution close to prior



For every minibatch of input data: compute this forward pass, and then backprop!

VAE: Generate data

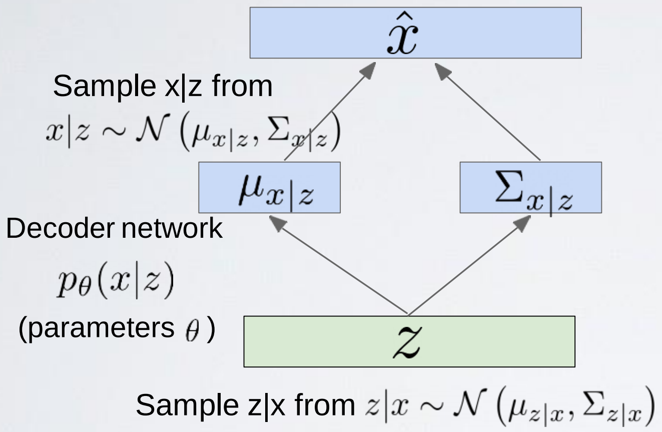
Use decoder network. Now sample z from prior!

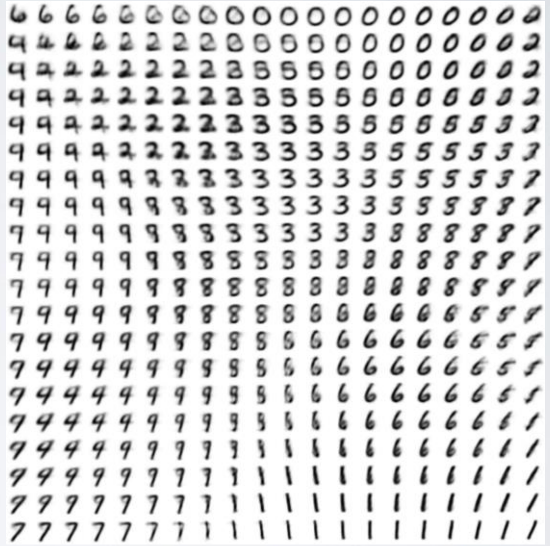


Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE: Generate data

Use decoder network. Now sample z from prior!

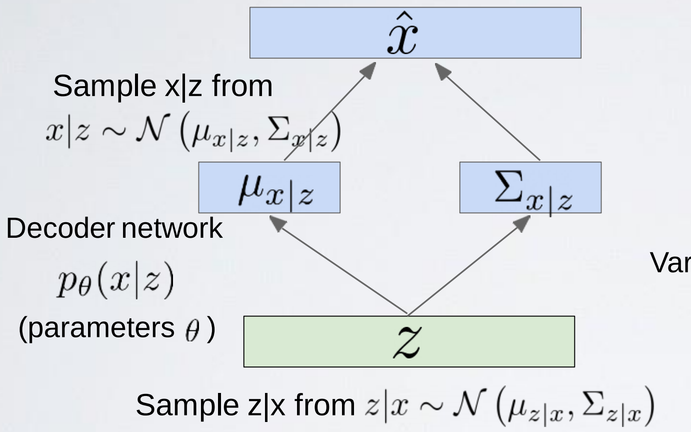




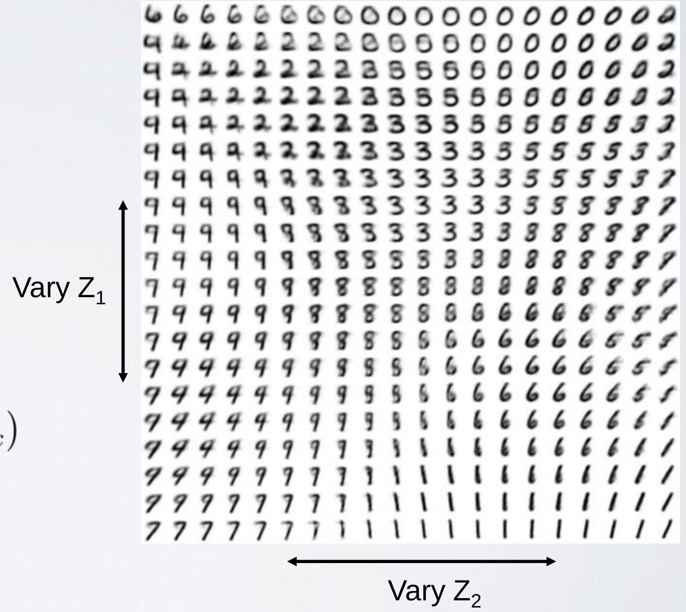
Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE: Generate data

Use decoder network. Now sample z from prior!



Data manifold for 2-d **z**



Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE: Generate data

Use decoder network. Now sample z from prior!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

Vary Z_1
Degree of smile

Vary Z_2 Head pose

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

VAE: Generate data

Use decoder network. Now sample z from prior!

Diagonal prior on **z** => independent latent variables

Different dimensions of **z** encode interpretable factors of variation

Also good feature representation that can be computed using $q_\phi(z|x)$

Vary Z_1
Degree of smile

Vary Z_2 Head pose

Kingma and Welling, "Auto-Encoding Variational Bayes", ICLR 2014

Softwares

- Vae – PyPI
  - https://pypi.org/project/vae/

- Deep learning platforms such as TensorFlow and PyTorch

References

[1] Kingma, Diederik P., and Max Welling. "An introduction to variational autoencoders." Foundations and Trends® in Machine Learning 12.4 (2019): 307-392.

[2] CS 3750 Previous slides

[3] Stanford CS 231n 2017 lecture 13

[4] Kingma, Diederik P., and Max Welling. "An introduction to variational autoencoders." Foundations and Trends® in Machine Learning 12.4 (2019): 307-392.

[5] Higgins, Irina, et al. "beta-VAE: Learning Basic Visual Concepts with a Constrained Variational Framework." Iclr 2.5 (2017): 6.