Image Generation

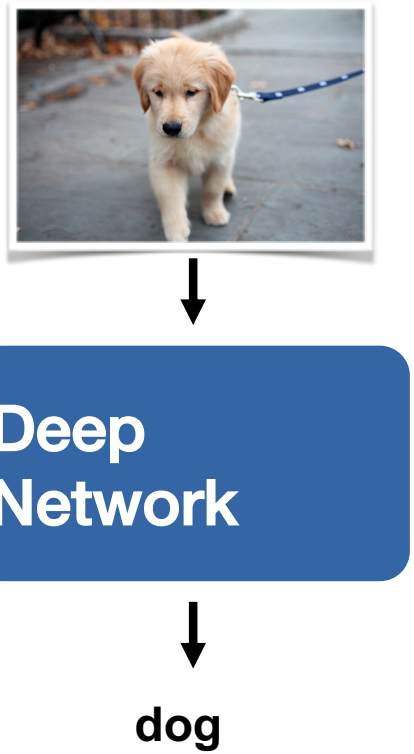
Slides adapted from S. Lazebnik, and others

Image generation

Input: Image
- High dimensional
- Structured

Output: Label
- Low dimensional

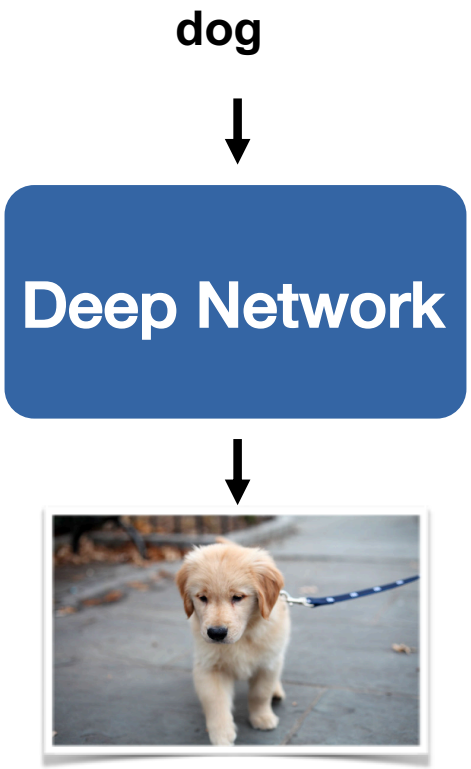
Easy



Input: Label / nothing
- Low dimensional

Output: Image
- High dimensional
- Many possibilities

Very hard



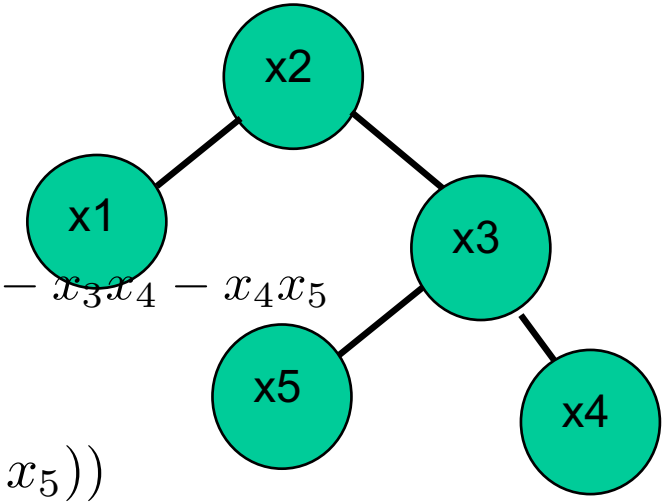
Boltzman Machines

Generative graphical models

Graphical model (Markov Random Field)

- Nodes are random variables
- Edges compatibility constraints
- Boltzman Machines – general undirected graphs
- nodes – binary random variables – edges (weights)
- Graph defines so energy function

$$E(x_1, x_2, x_3, x_4, x_5) = -\sum_{i=1}^{5} b_i x_i + \sum_i \sum_j x_i w_{ij} x_i$$



$$E(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2 + 2x_3 - x_1x_2 - x_2x_3 - x_3x_4 - x_4x_5$$

- Associated probability distribution

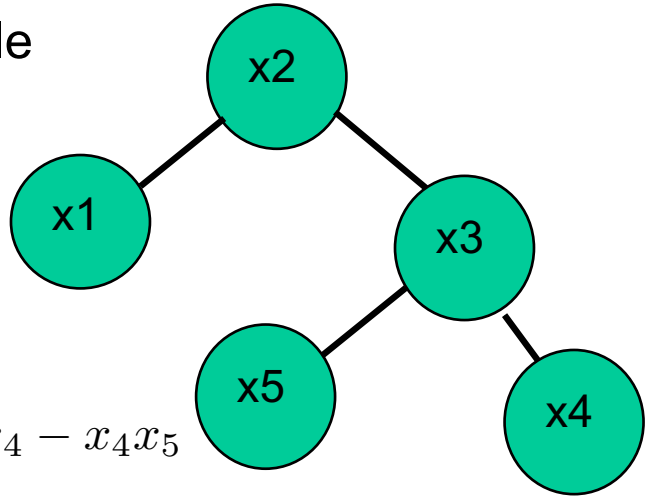
$$P(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z} \exp(-E(x_1, x_2, x_3, x_4, x_5))$$

- Structured probabilistic models

Boltzman Machine

Inference Tasks

- What is the probability of particular configuration
- What the the maximum energy
- What is the most probable configuration
- Combinatorial problem (evaluate over all possible
- 5 dim binary vectors 2^5
- use sampling



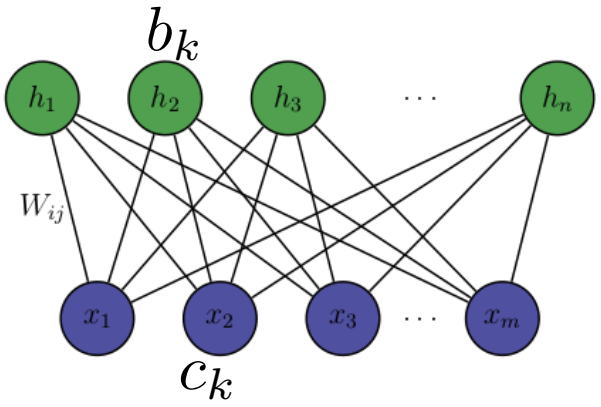
$$E(x_1, x_2, x_3, x_4, x_5) = -\sum_{i=1}^{5} b_i x_i + \sum_i \sum_j x_i w_{ij} x_i$$

$$E(x_1, x_2, x_3, x_4, x_5) = x_1 + x_2 + 2x_3 - x_1 x_2 - x_2 x_3 - x_3 x_4 - x_4 x_5$$

$$P(x_1, x_2, x_3, x_4, x_5) = \frac{1}{Z} \exp(-E(x_1, x_2, x_3, x_4, x_5))$$

Restricted Boltzman machines

- Graph is bipartite
- x set of observable nodes
- h set of hidden nodes
- c, b are bias vectors
- Define probability distribution over x and h
- Z partition function intractable



$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^T \mathbf{W} \mathbf{x} - \mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{h}$$

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

- Some intuition how parameters affect the probability of configuration

Restricted Boltzman machines

Factor view, probability as product of factors

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

$$p(\mathbf{x}, \mathbf{h}) = \exp -(-\mathbf{h}^T \mathbf{W}\mathbf{x} - \mathbf{c}^T \mathbf{x} - \mathbf{b}^T \mathbf{h})/Z$$

$$p(\mathbf{x}, \mathbf{h}) = \exp\left(\mathbf{h}^T \mathbf{W}\mathbf{x}\right) \exp\left(\mathbf{c}^T \mathbf{x}\right) \exp\left(\mathbf{b}^T \mathbf{h}\right)/Z$$

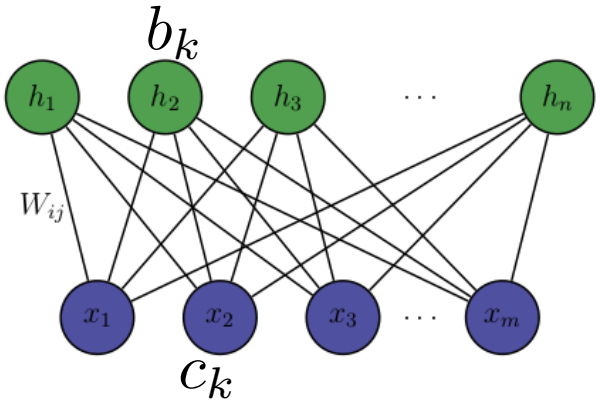
Inference in RBM's

$$p(\mathbf{x}, \mathbf{h}) = \exp\left(\mathbf{h}^T \mathbf{W} \mathbf{x}\right) \exp\left(\mathbf{c}^T \mathbf{x}\right) \exp\left(\mathbf{b}^T \mathbf{h}\right) / Z$$

Computation of probability – intractable (approximate)

Conditional inferences (conditional probabilities) possible

- similar to outputs on NN



$$p(\mathbf{h}|\mathbf{x}) = \Pi_j p(h_j|\mathbf{x})$$

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

$$p(\mathbf{x}|\mathbf{h}) = \Pi_k p(x_k|\mathbf{h})$$

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

This can be derived from

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{h'} p(x,\mathbf{h}')}$$

Partition function cancels out

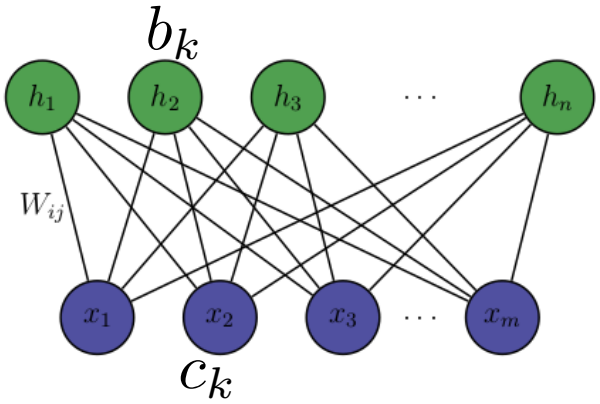
RBM's

$$p(\mathbf{x}, \mathbf{h}) = \exp\left(\mathbf{h}^T \mathbf{W} \mathbf{x}\right) \exp\left(\mathbf{c}^T \mathbf{x}\right) \exp\left(\mathbf{b}^T \mathbf{h}\right) / Z$$

Free energy

How to model

$$p(\mathbf{x}) = \sum_{\mathbf{h} \in \{0,1\}}^{H} p(\mathbf{x}, \mathbf{h})$$



… after some derivations

$$= \exp\left(\mathbf{c}^T \mathbf{x} + \sum_{i=1}^{H} \log\left(1 + \exp\left(b_j + \mathbf{W}_j \mathbf{x}\right)\right)\right) / Z$$

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

This can be derived from

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x}, \mathbf{h})}{\sum_{h'} p(x, \mathbf{h}')}$$

Partition function cancels out

Training RBM's

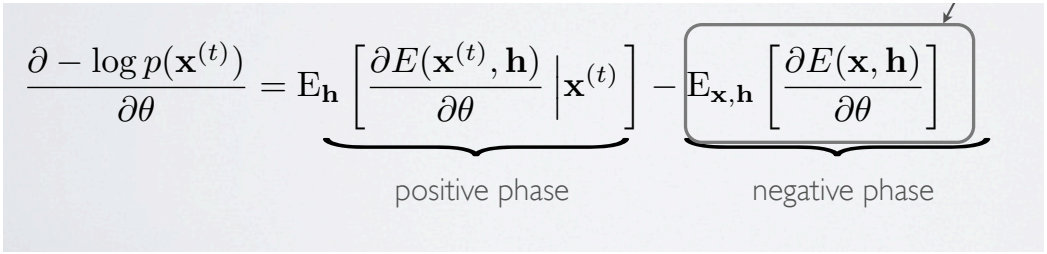
$$p(\mathbf{x}, \mathbf{h}) = \exp\left(\mathbf{h}^T \mathbf{W} \mathbf{x}\right) \exp\left(\mathbf{c}^T \mathbf{x}\right) \exp\left(\mathbf{b}^T \mathbf{h}\right) / Z$$

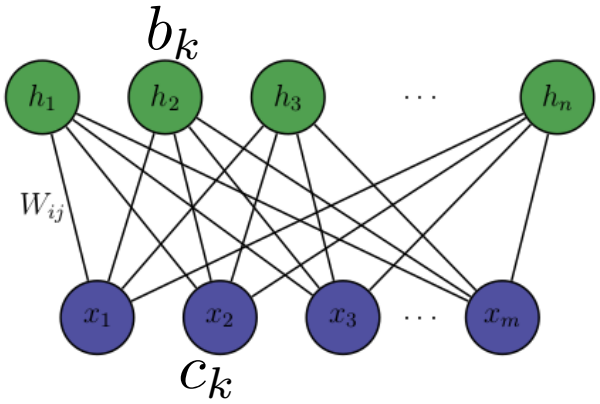
$$p(\mathbf{x}) = \exp\left(-F(\mathbf{x})\right) / Z$$

How to train RBM that models well the data

Minimize average negative log likelihood

$$\frac{1}{T} \sum_t - \log p(\mathbf{x}^{(t)})$$





This can be derived from

$$p(\mathbf{h}|\mathbf{x}) = \frac{p(\mathbf{x},\mathbf{h})}{\sum_{h'} p(x,\mathbf{h}')}$$

Partition function cancels out

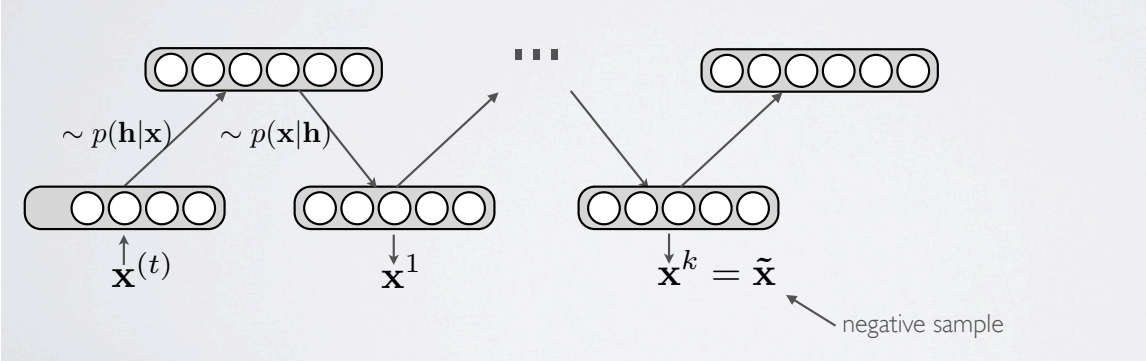
Dahl, Adams, Larochelle Training Restricted Boltzmann Machines on Word Observations, ICML 2012

Slides from Hugo Larochelle
http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Contrastive Divergence

Idea

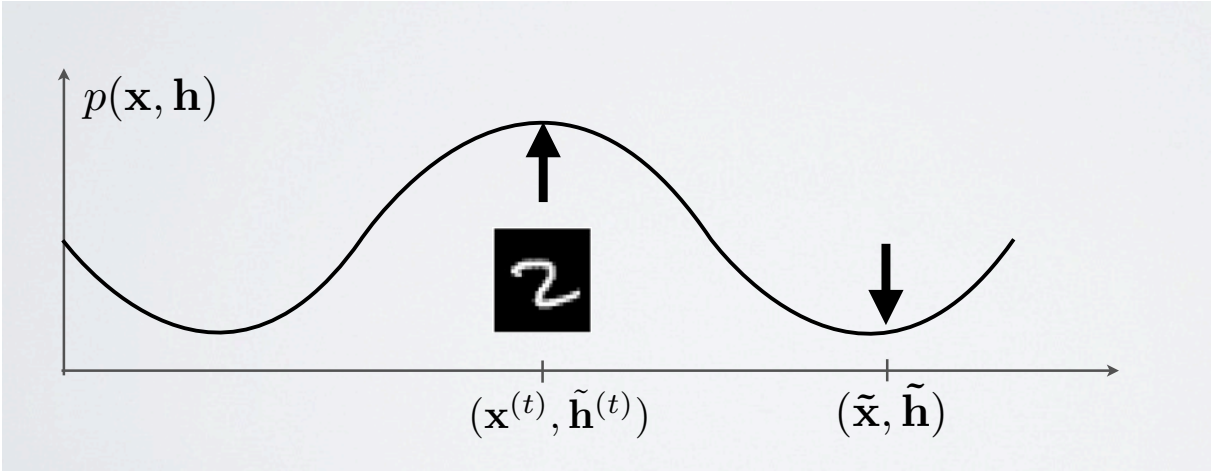
1. Replace expectation by point estimate $\tilde{\mathbf{x}}$
2. Obtain the point estimate by Gibbs sampling $\tilde{\mathbf{x}}$
3. Start sampling chain at $\mathbf{x}^{(t)}$

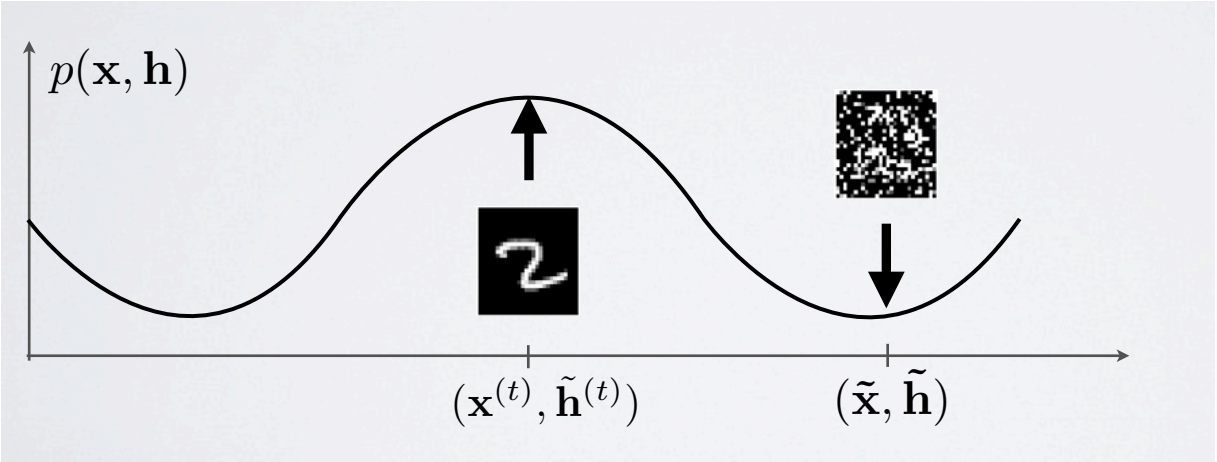


Slides from Hugo Larochelle
http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Contrastive divergence training

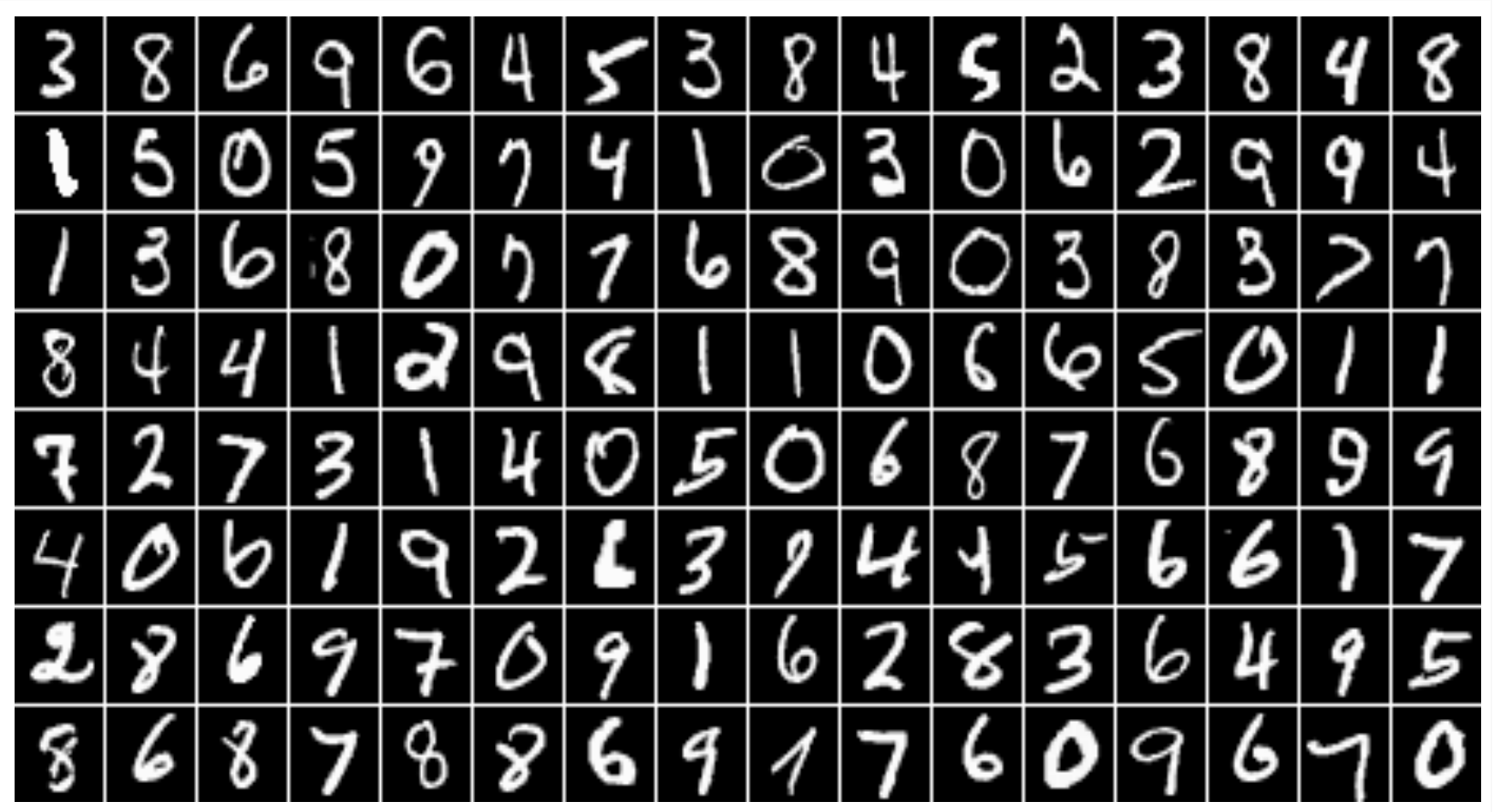
As you proceed sampled values are going to look more
Training examples





Slides from Hugo Larochelle
http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

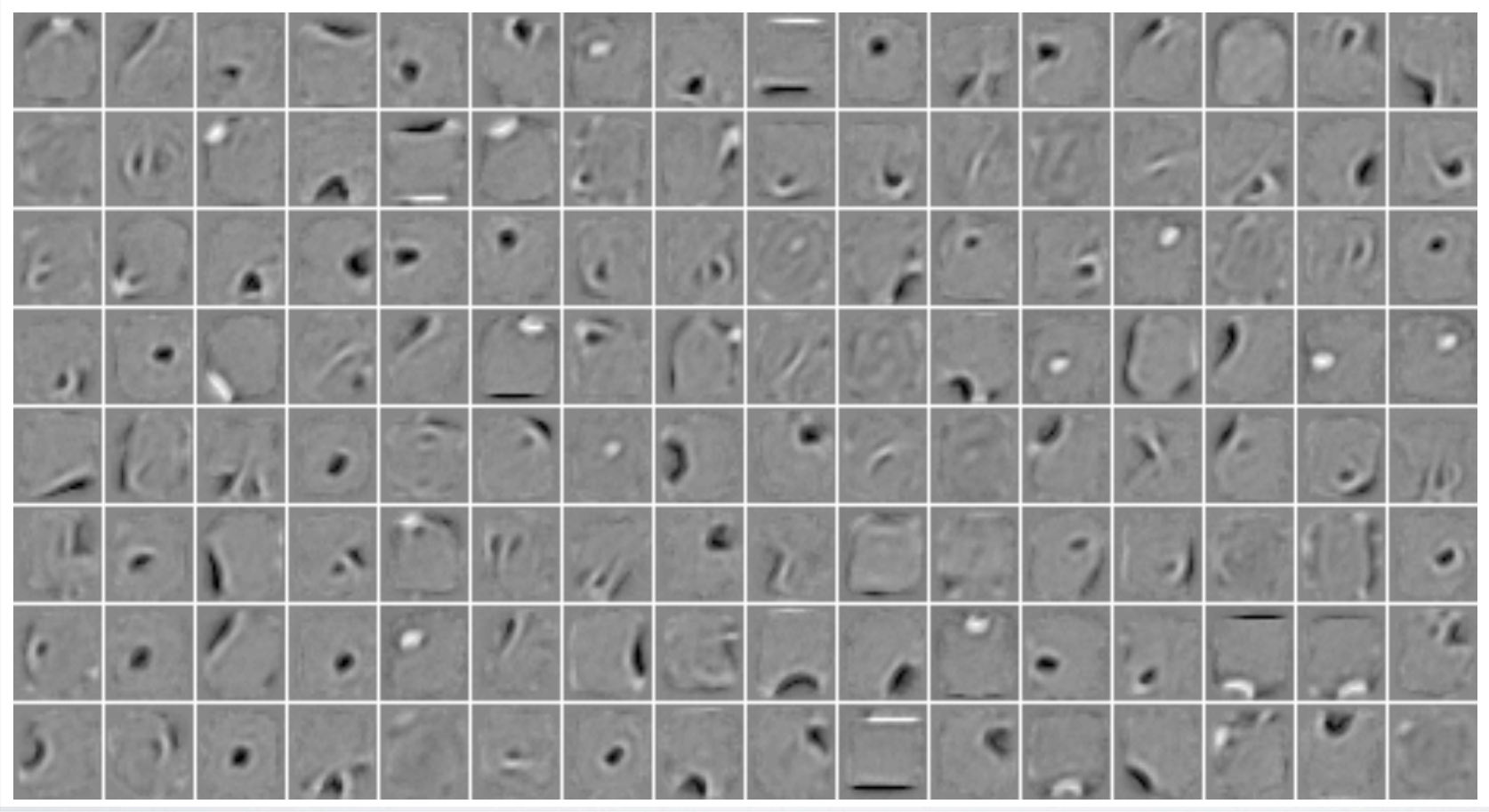
Example



Larochelle et al JMLR 2009

Slides from Hugo Larochelle
http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Learned filters

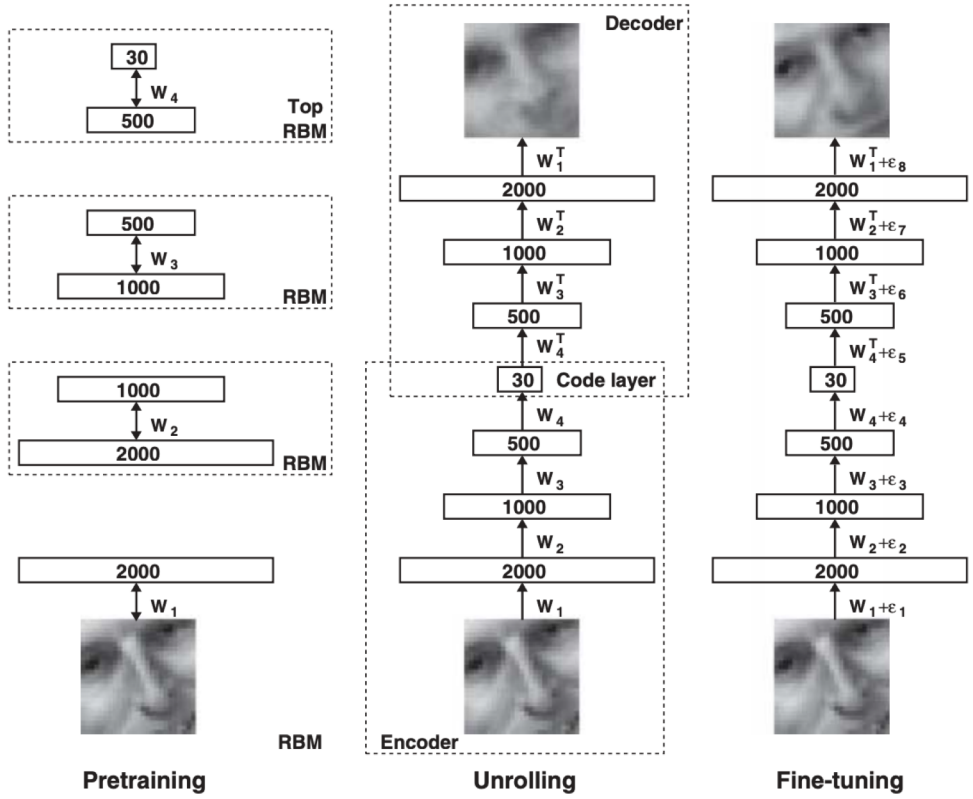


Larochelle et al JMLR 2009

Nonlinear dimensionality reduction

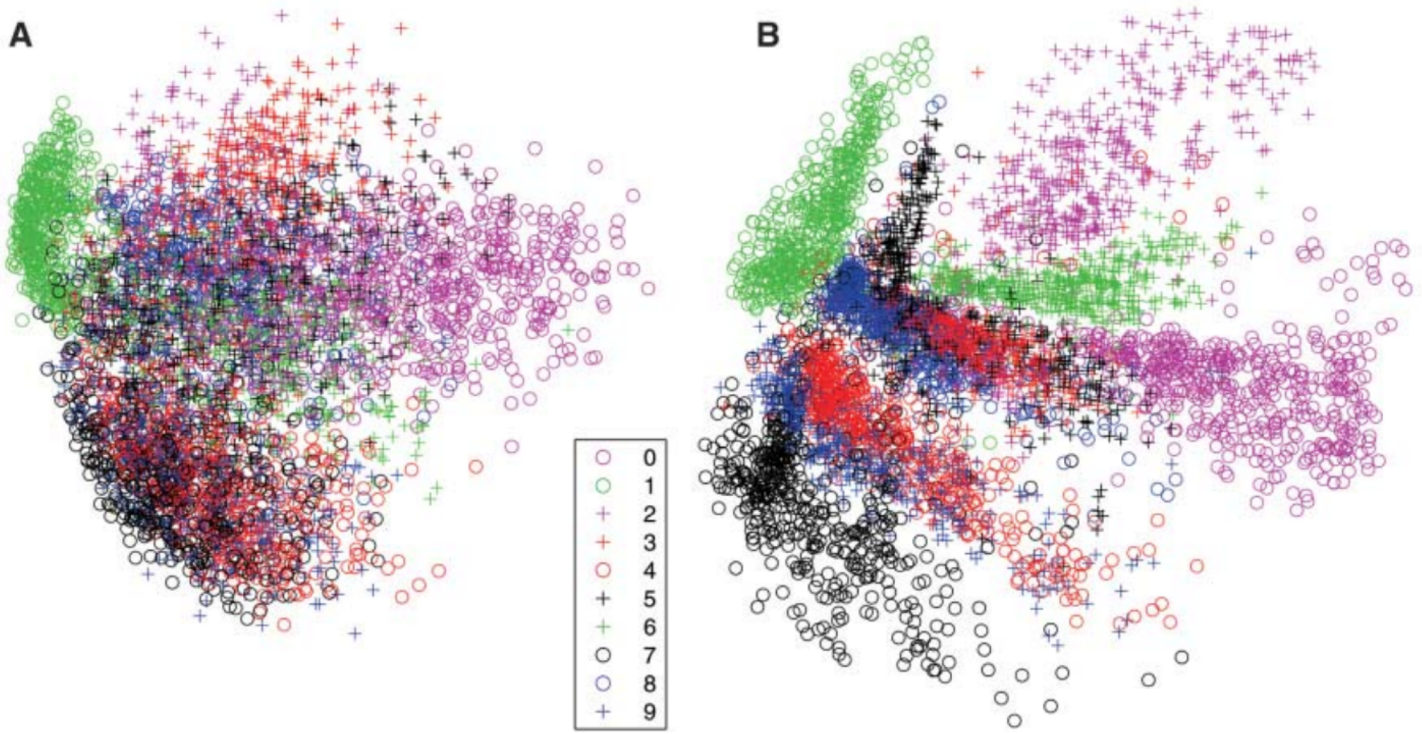
- extensions to discrete random variables
- deeper models, stack of RBM's

Reducing the Dimensionality of Data with Neural Networks,
Hinton and Salakhutdinov, Science, 2006



Nonlinear Dimensionality reduction

Fig. 3. (**A**) The two-dimensional codes for 500 digits of each class produced by taking the first two principal components of all 60,000 training images. (**B**) The two-dimensional codes found by a 784-1000-500-250-2 autoencoder. For an alternative visualization, see (*8*).



Reducing the Dimensionality of Data with Neural Networks,
Hinton and Salakhutdinov, Science, 2006

Autoencoder

Feedforward NN

Tries to reconstruct input

The dim of output is the same as input

For binary inputs sigm

Single layer NN encoder

$$\mathbf{h}(\mathbf{x}) = \text{sigm}(\mathbf{b} + \mathbf{W}\mathbf{x})$$

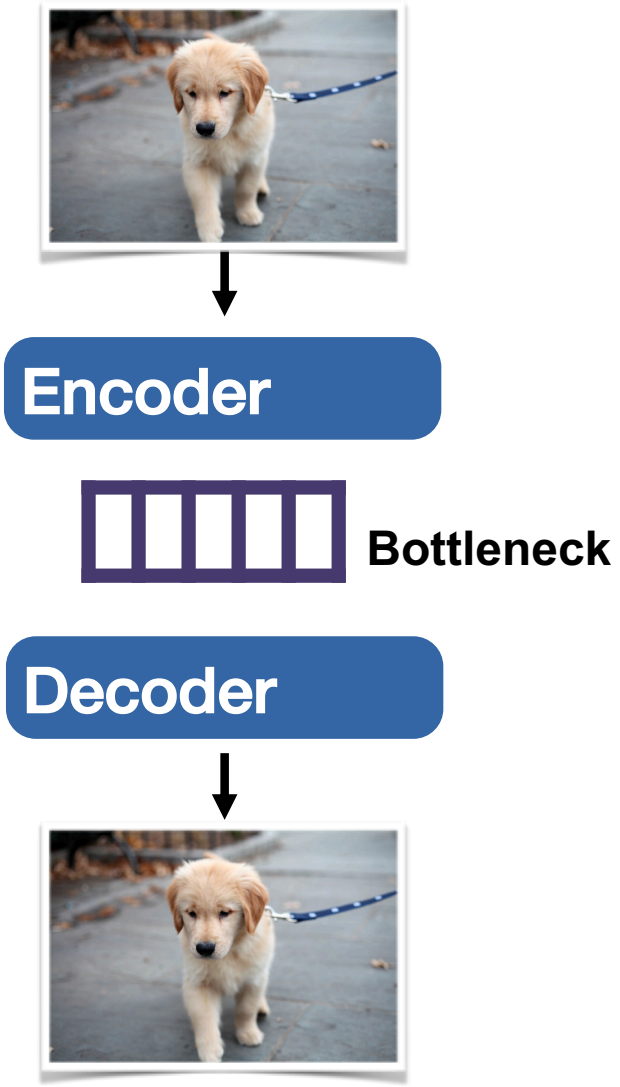
Decoder

$$\hat{\mathbf{x}} = \text{sigm}(\mathbf{c} + \mathbf{W}^*\mathbf{h}(\mathbf{x}))$$

Sometimes weights are tied

$$\mathbf{W}^* = \mathbf{W}^T$$

Hidden representation maintains everything about input – idea compression of data



Loss function

For binary inputs

$$l(f(\mathbf{x})) = -\sum_k (x_k \log(\hat{x_k}) + (1 - x_k) \log(1 - \hat{x_k})$$

For real valued inputs

$$l(f(\mathbf{x})) = \frac{1}{2} \sum_k (\hat{x_k} - x_k)^2$$

Gradient simple form – regular backprop

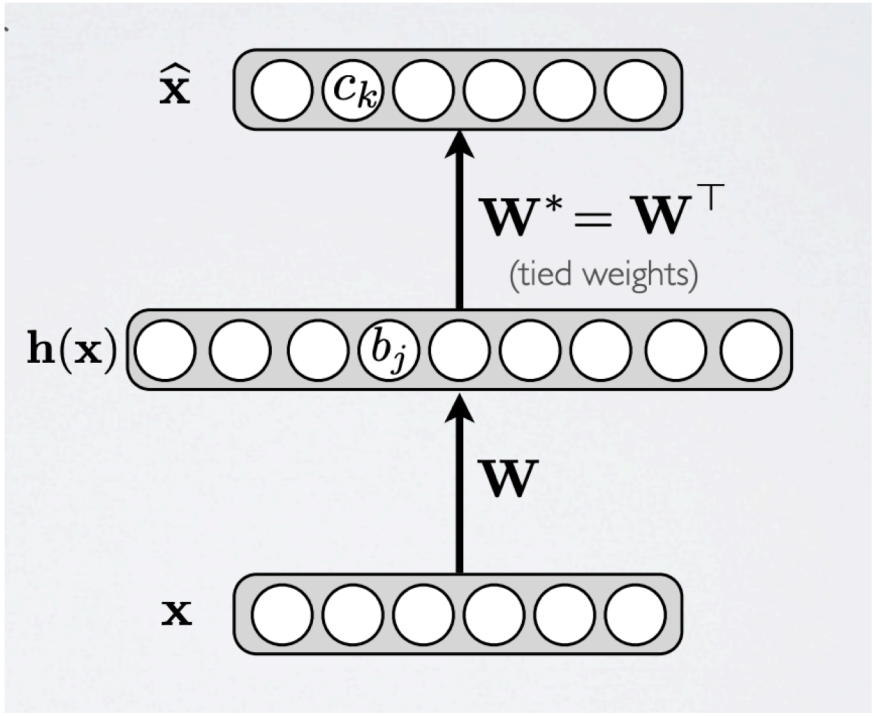
Choices

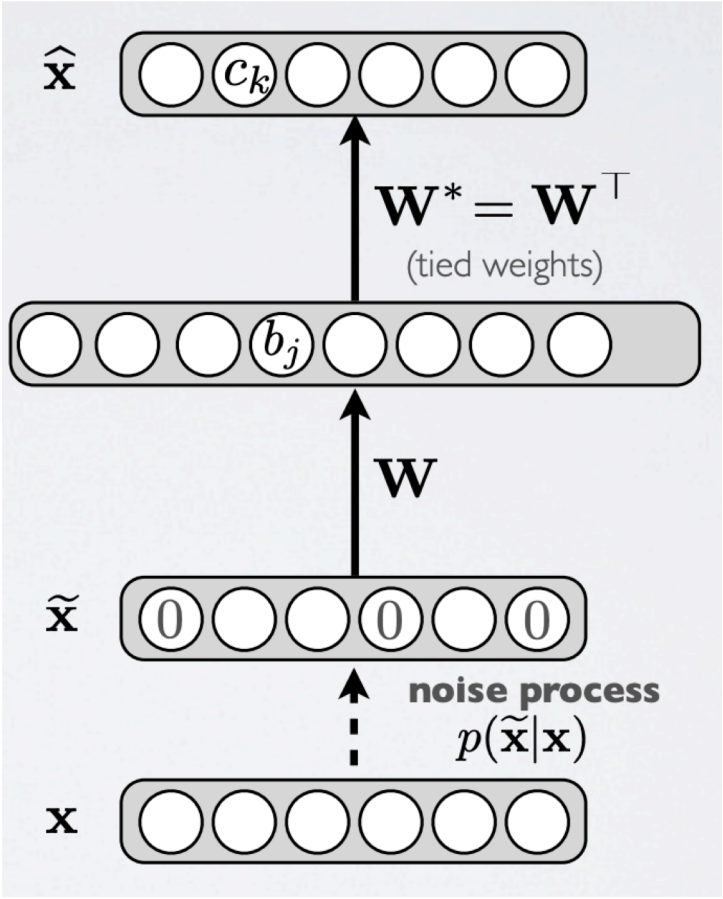
- hidden layers undercomplete, compression for the training distribution, not so good for other examples
- hidden layers overcompelete, does not extract useful representation

Denoising Autoencoder

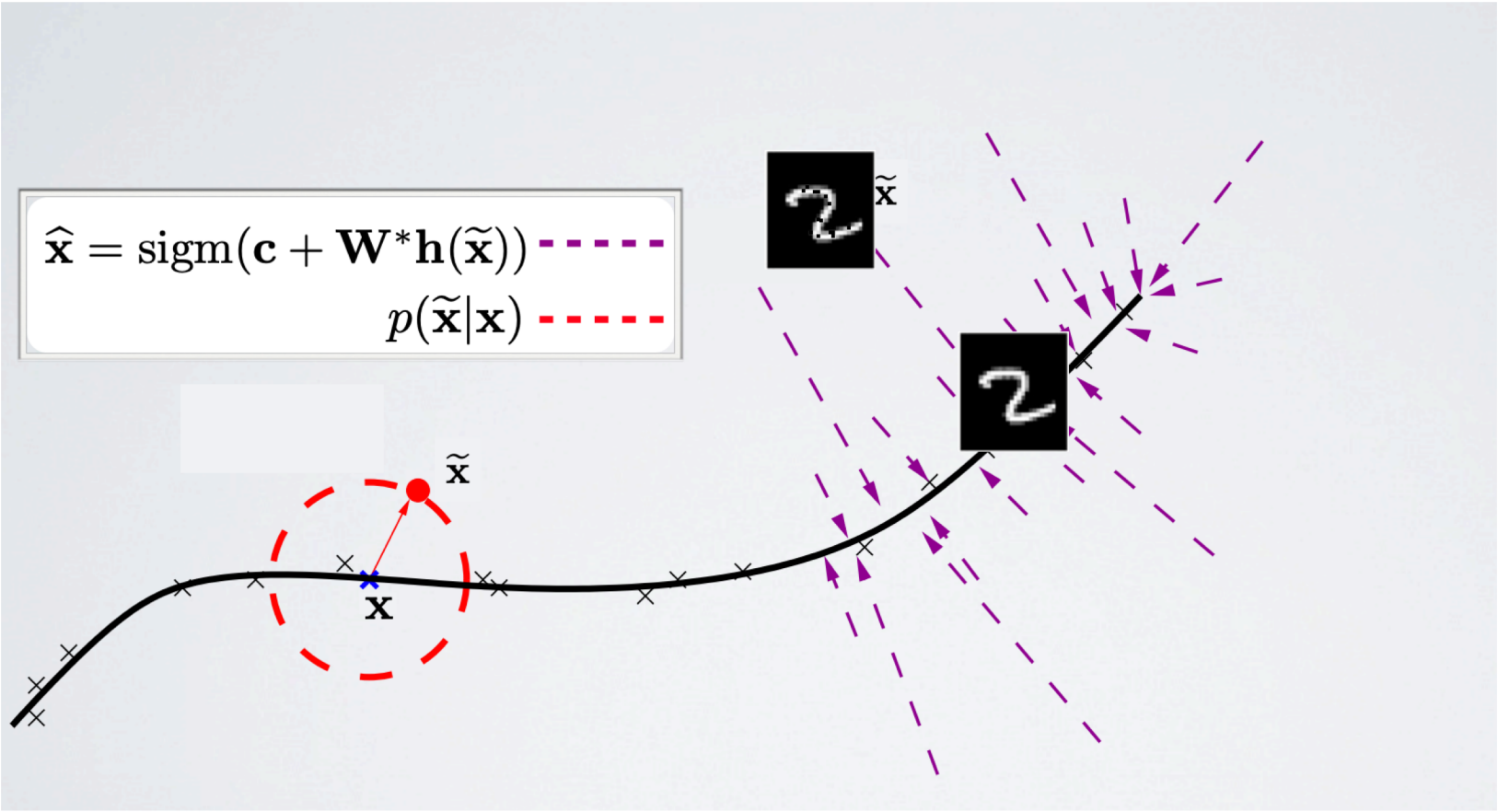
Idea: train with the noise corrupted samples –

Learn how to denoise, learn how to denoise





Denoising autoencoder

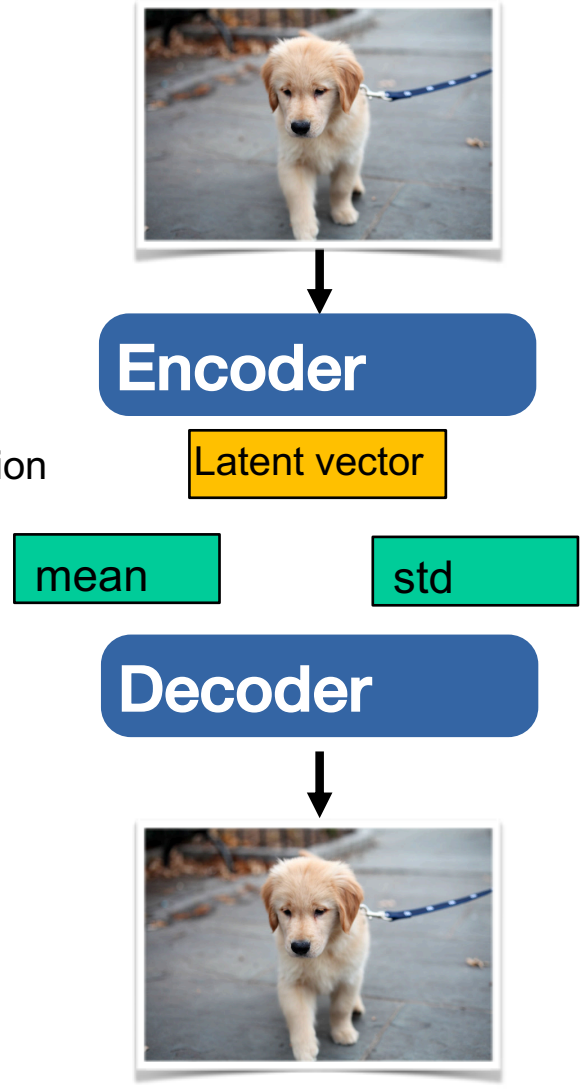


Slides from Hugo Larochelle
http://info.usherbrooke.ca/hlarochelle/neural_networks/content.html

Variational Autoencoder

- Encoder $\boldsymbol{P(x) = \int P(x|z)P(z)dz}$
  - $q(\mathbf{z}\,|\,\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; \mu_\theta(\mathbf{x}), \sigma^2_\theta(\mathbf{x})\mathbf{I}\right)$
- Sampling $\mathbf{f} \sim q(\mathbf{z}\,|\,\mathbf{x})$
- Decoder
  - $P(\mathbf{x}\,|\,\mathbf{f})$
- Approximately learns $P(\mathbf{x})$
  - Variational lower bound

Sample latent vector from distribution



Variational Autoencoder

- Encoder

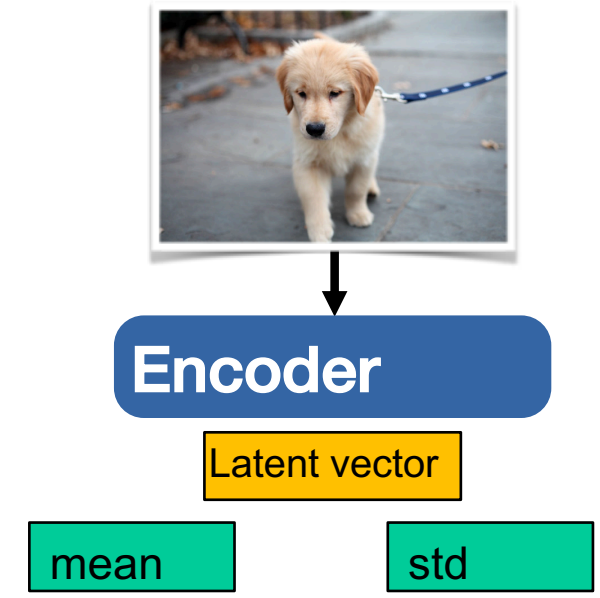
$$P(x) = \int P(x|z)P(z)dz$$

  - $q(\mathbf{z}|\mathbf{x}) = \mathcal{N}\left(\mathbf{z}; \mu_\theta(\mathbf{x}), \sigma_\theta^2(\mathbf{x})\mathbf{I}\right)$

- Sampling $\mathbf{f} \sim q(\mathbf{z}|\mathbf{x})$

- Decoder

  - $P(\mathbf{x}|\mathbf{f})$



Loss function

$$l_i(\theta, \phi) = -\mathbb{E}_{z \sim q_\theta(z|x_i)}[\log p_\phi(x_i \mid z)] + \mathbb{KL}(q_\theta(z \mid x_i) \,||\, p(z))$$

Reconstruction loss and KL divergence
p(z) is N(0,1)

Variational Autoencoder

Instead of sampling, from the distribution
To get the latent vector
use reparametrization trick

Make the latent vector mean
And add std and then corrupt std
with N(0,1) gaussian

Encoder

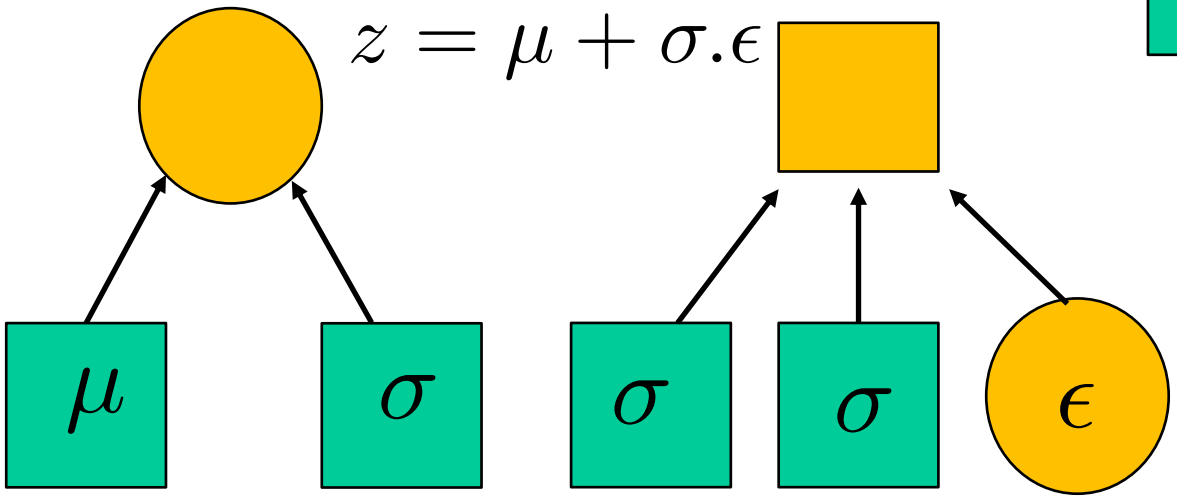
Latent vector

Sample latent vector from distribution

mean

std

$$z = \mu + \sigma . \epsilon$$



Decoder

VAE faces

Alec Radford Face Manifold from variational autoencoder