Northeastern

CS 7140:
ADVANCED MACHINE LEARNING

Recap: Boltzmann Machine

- Compute activity $a_i(t) = \sum_j w_{ij} x_j(t)$

- Set $x_i = +1$ with probability $\dfrac{1}{1 + e^{-2a_i}}$
  Else set $x_i = -1$

- Sample from a Boltzmann Machine with Gibbs sampling for probability distribution
  $P(x \mid \beta, W) = \dfrac{1}{Z} \exp[-\beta E(x; W)]$

Recap: BM Learning

- Gradient of the log likelihood

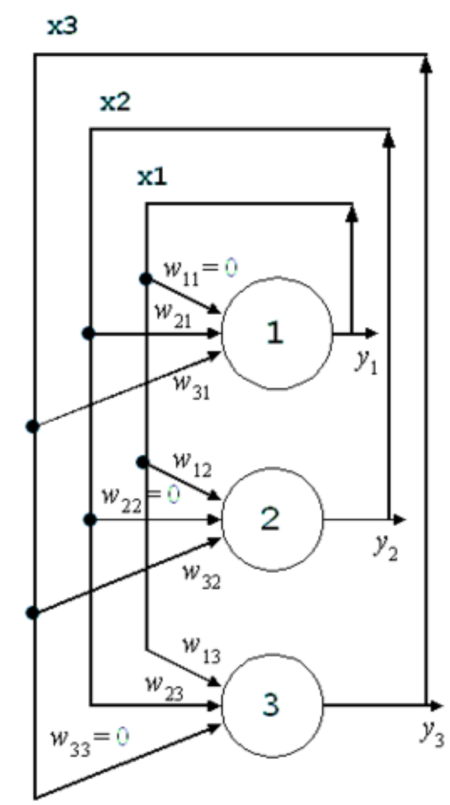
$$\frac{\partial}{\partial w_{ij}} \log P(\{\mathbf{x}^{(n)}\} \mid \mathbf{W}) = \sum_{n}^{N} \Big[\underbrace{\langle x_i^{(n)} x_j^{(n)} \rangle}_{\boxed{\text{data correlation}}} - \underbrace{\langle x_i x_j \rangle_{P(\mathbf{x}^{(n)})}}_{\boxed{\text{model correlation}}} \Big]$$

- $\langle x_i x_j \rangle_{\text{Data}} = \frac{1}{N} \sum_{n} x_i^{(n)} x_j^{(n)} \quad \langle x_i x_j \rangle_{P(\mathbf{x}^{(n)})} = \sum_{\mathbf{x}} x_i x_j P(\mathbf{x} \mid \mathbf{W})$

- Data correlation is easy but model correlation is hard — estimate by Monte Carlo

Recap:Binary Hopfield Network

- **Weights** w_{ij} denotes the connection from neuron i to neuron j
- **Architecture** symmetric, bidirectional connections $w_{ij} = w_{ji}$, no self-connections $w_{ii} = 0$



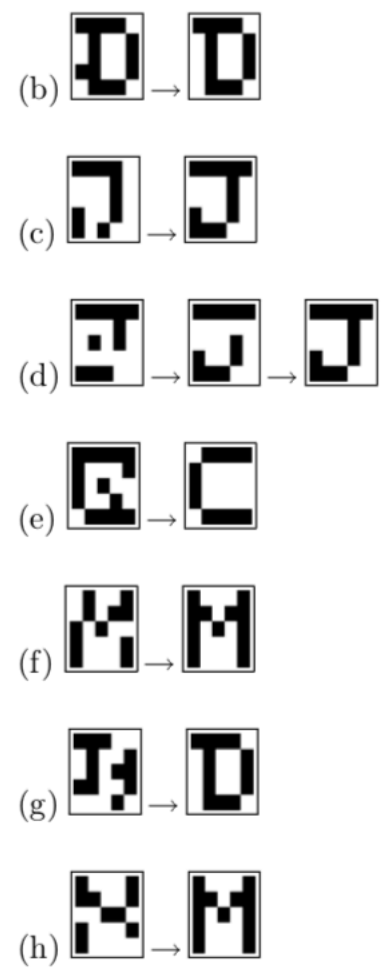
- **Activity rule** single neuron update

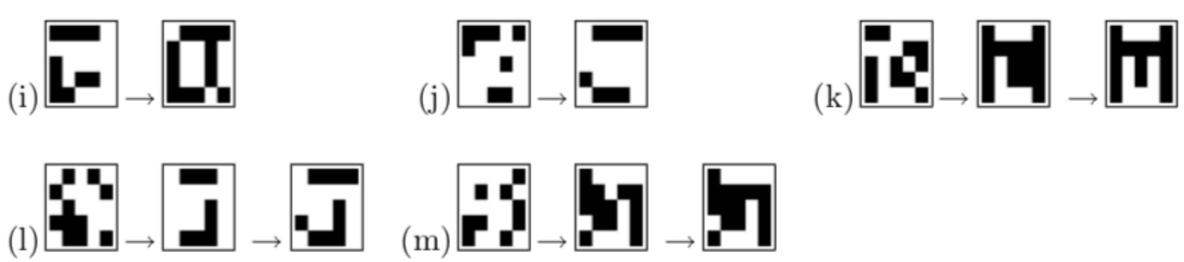
$$x(a) = \begin{cases} 1 & a \geq 0 \\ -1 & a < 0 \end{cases}$$

- **Updates** activations $a_i = \sum_j w_{ij} x_j$
- **Learning rule** make a set of desired memory $\{x^{(n)}\}$ be stable states, set weights $w_{ij} = \eta \sum_{ij} x_i^{(n)} x_j^{(n)}$

Recap:Associative Memory

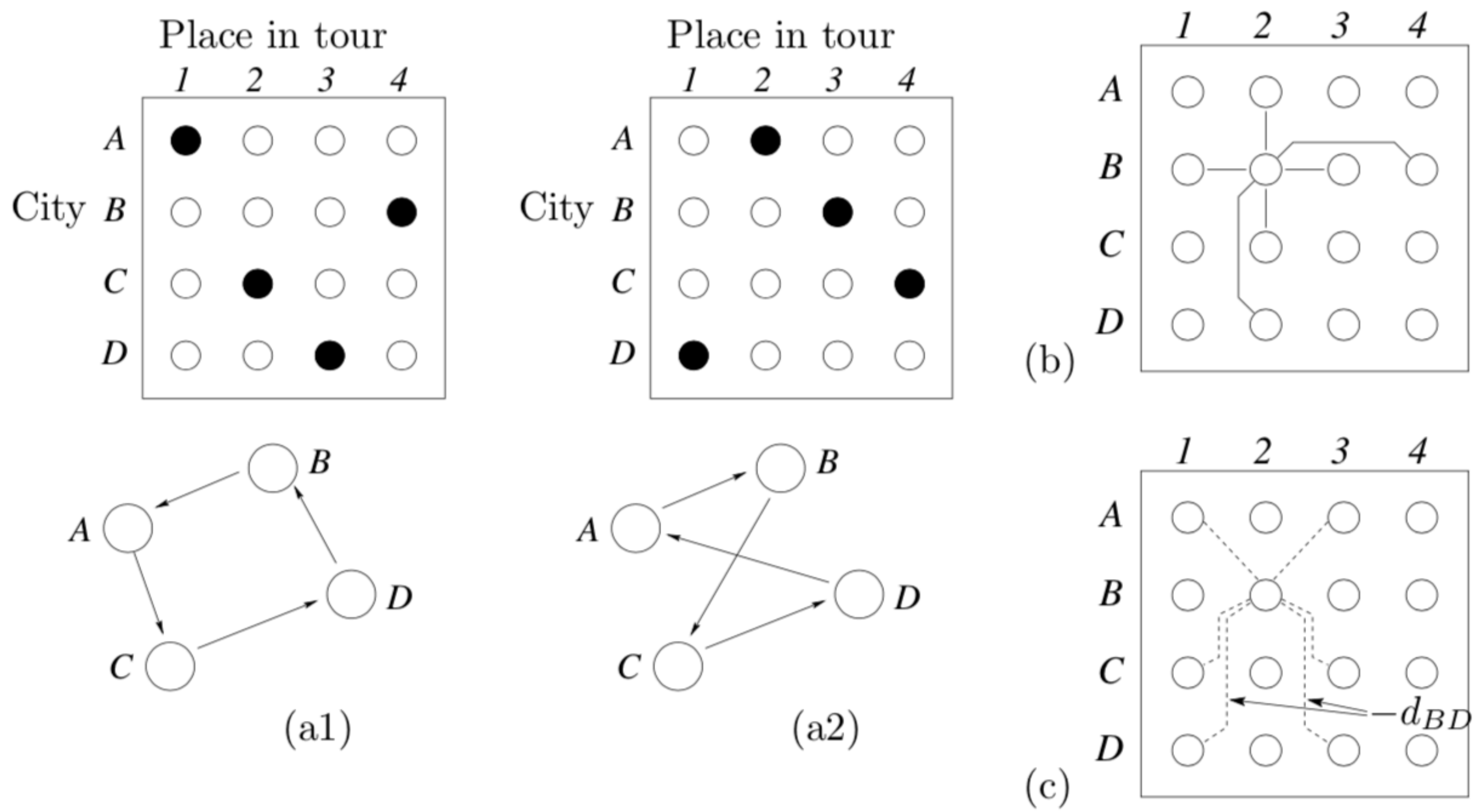
.	0	0	0	0	-2	2	-2	2	2	-2	0	0	0	2	0	0	-2	0	2	2	0	0	-2	-2
0	.	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	.	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	4	4	.	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	4	2	-2
0	0	0	0	.	2	-2	-2	2	-2	2	-4	0	0	-2	4	-4	-2	0	-2	2	0	0	-2	2
-2	-2	-2	-2	2	.	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
2	-2	-2	-2	-2	0	.	0	0	4	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	-2	0	0	.	0	0	0	2	2	2	0	-2	2	4	2	0	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	.	0	0	-2	2	2	0	2	-2	0	2	0	4	-2	-2	-4	0
2	-2	-2	-2	-2	0	4	0	0	.	0	2	2	-2	4	-2	2	0	-2	4	0	-2	-2	0	0
-2	-2	-2	-2	2	4	0	0	0	0	.	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	4
0	0	0	0	-4	-2	2	2	-2	2	-2	.	0	0	2	-4	4	2	0	2	-2	0	0	2	-2
0	-4	-4	-4	0	2	2	2	2	2	2	0	.	0	2	0	0	2	0	2	2	-4	-4	-2	2
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	.	-2	0	0	2	4	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	.	-2	2	0	-2	4	0	-2	-2	0	0
0	0	0	0	4	2	-2	-2	2	-2	2	-4	0	0	-2	.	-4	-2	0	-2	2	0	0	-2	2
0	0	0	0	-4	-2	2	2	-2	2	-2	4	0	0	2	-4	.	2	0	2	-2	0	0	2	-2
-2	-2	-2	-2	-2	0	0	4	0	0	0	2	2	2	0	-2	2	.	2	0	0	-2	-2	0	0
0	0	0	0	0	-2	-2	2	2	-2	-2	0	0	4	-2	0	0	2	.	-2	2	0	0	-2	-2
2	-2	-2	-2	-2	0	4	0	0	4	0	2	2	-2	4	-2	2	0	-2	.	0	-2	-2	0	0
2	-2	-2	-2	2	0	0	0	4	0	0	-2	2	2	0	2	-2	0	2	0	.	-2	-2	-4	0
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	.	4	2	-2
0	4	4	4	0	-2	-2	-2	-2	-2	-2	0	-4	0	-2	0	0	-2	0	-2	-2	4	.	2	-2
-2	2	2	2	-2	0	0	0	-4	0	0	2	-2	-2	0	-2	2	0	-2	0	-4	2	2	.	0
-2	-2	-2	-2	2	4	0	0	0	0	4	-2	2	-2	0	2	-2	0	-2	0	0	-2	-2	0	.





- 25 -unit binary Hopfield network
- (a) four patterns as 5x5 binary images
- (b)- (m) evolution of state of the network

Recap:TSP on Hopfield Network

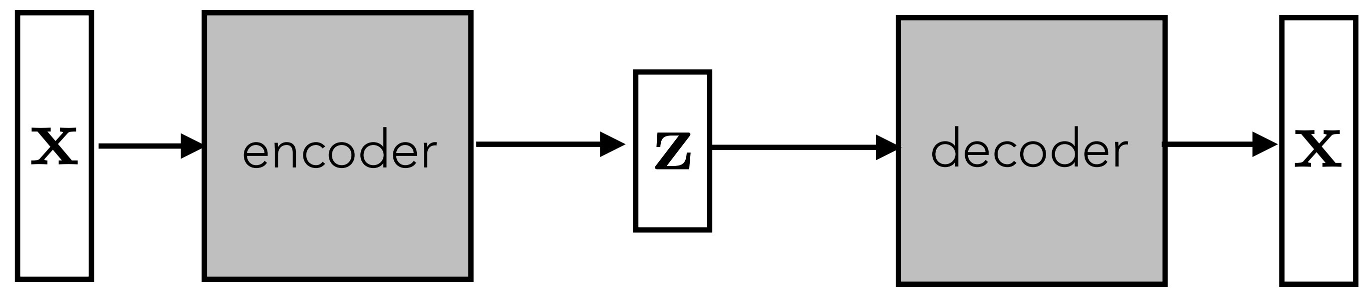


- **States have exactly one `1` in every row/column**: putting large negative weights between any pair of neurons in the same row/column
- **Weights encode the total distance**: putting negative weights proportional to the distance between the nodes

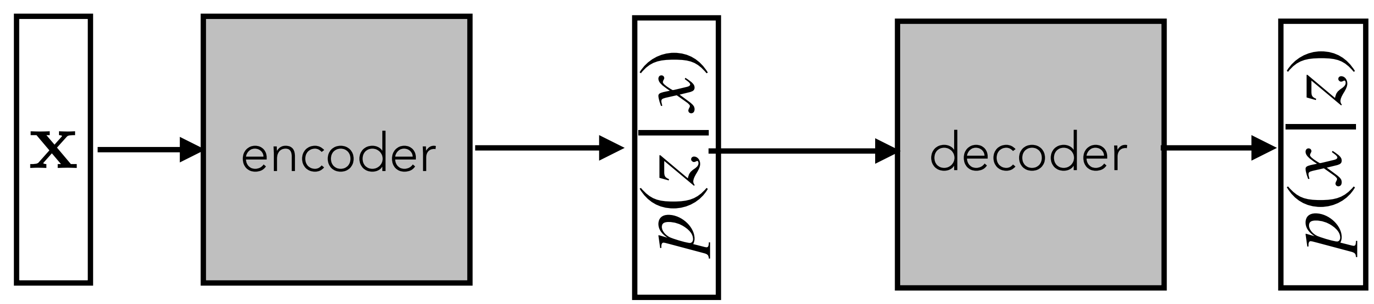
AUTOENCODERS

Autoencoder

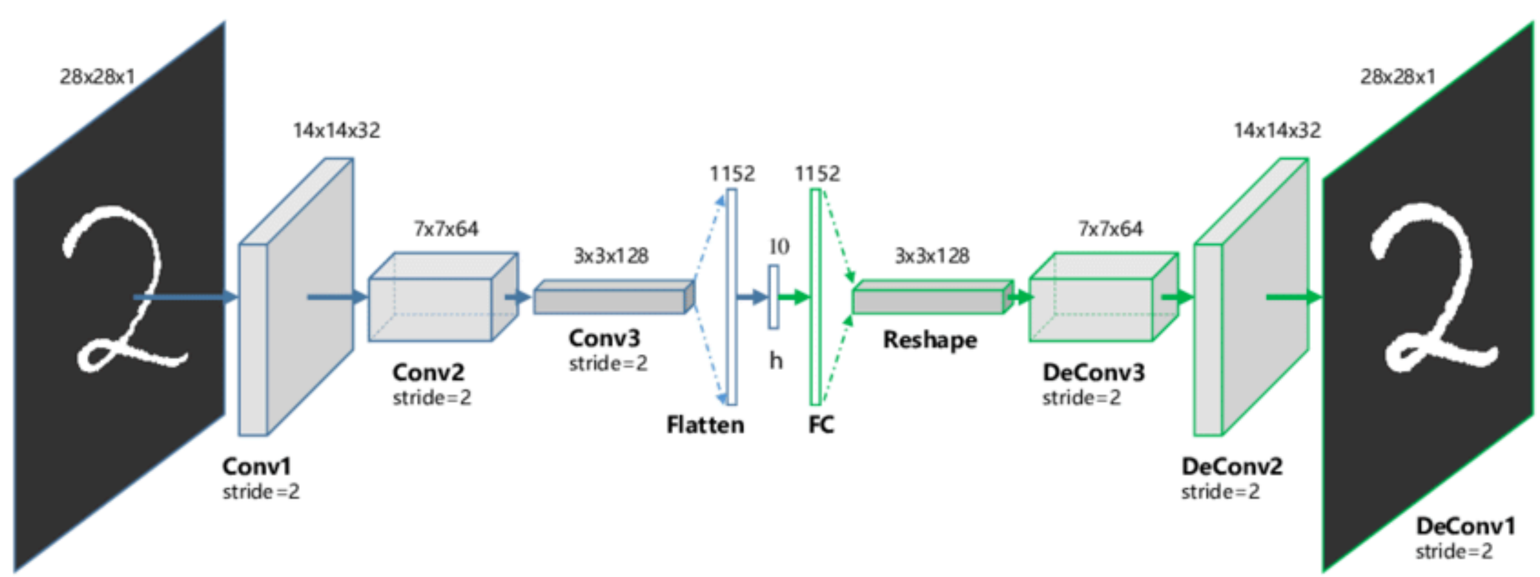
- Deterministic: dimension reduction/feature learning



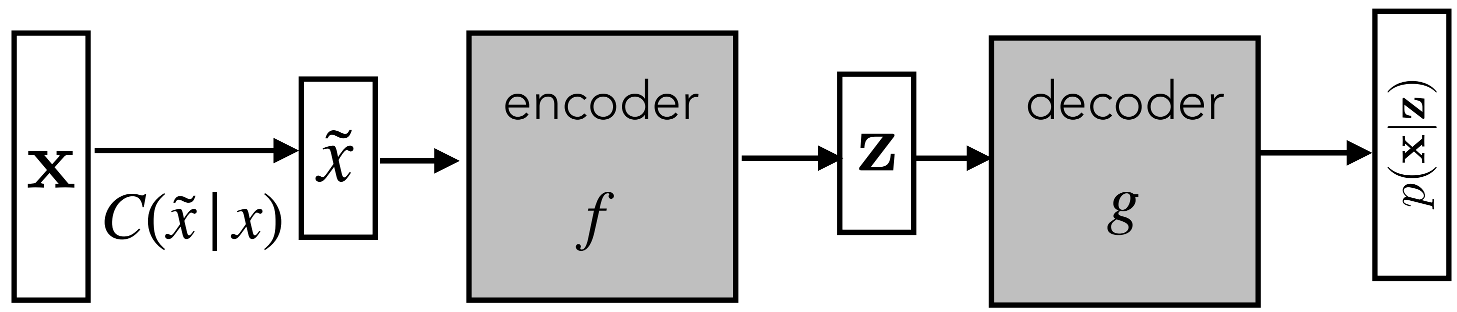
- Stochastic:



- Copy input to output



Denoising Autoencoder



- Clean data x, corrupted data $\tilde{x}$, corruption process $C(\tilde{x} \mid x)$, reconstruction distribution $p(x \mid \tilde{x})$

- Estimate using gradient-based optimization to minimize $\|g(f(\tilde{x})) - x\|^2$

- Equivalent to performing SGD on $-\mathbb{E}_{x\sim p(x)}\mathbb{E}_{\tilde{x}\sim C(\tilde{x}|x)} \log p(x \mid z)$

VARIATIONAL AUTOENCODER

Approximate Inference

- A method for approximating a complex distribution
- Gibbs inequality

$$D_{KL}(Q\,||\,P) = \sum_x Q(x) \log \frac{Q(x)}{P(x)} \geq 0$$

  - relative entropy is always nonnegative
  - non symmetric: $D_{KL}(P\,||\,Q) \neq D_{KL}(Q\,||\,P)$
- Applications in statistical physics, machine learning and data science

Variational Free Energy

- Approximating the **complex** $P(x)$ with a **simple** $Q(x;\theta)$
- Probability distribution $P(x) = \frac{1}{Z}P^{\star}(x) = \frac{1}{Z}\prod_{m=1}^{M}\phi(x_m)$
- By Gibbs inequality

$$D_{KL}(Q\,||\,P) = \log Z - \sum_m \mathbb{E}_Q[\log\phi] - H_Q$$

$$= \log Z + F[P^{\star}, Q] \geq 0$$

variational free energy

- Minimizing the relative entropy is equivalent to minimizing the variational free energy

ELBO

- Energy functional is a *lower bound* of the partition function $\log Z \geq -F[P^\star, Q] = \sum_m \mathbb{E}_Q[\log \phi] + H_Q$

$$\log p(x) = \log \int_z p(x,z)\frac{q(z)}{q(z)}dz$$

$$= \log \mathbb{E}_q\left[\frac{p(x,z)}{q(z)}\right]$$

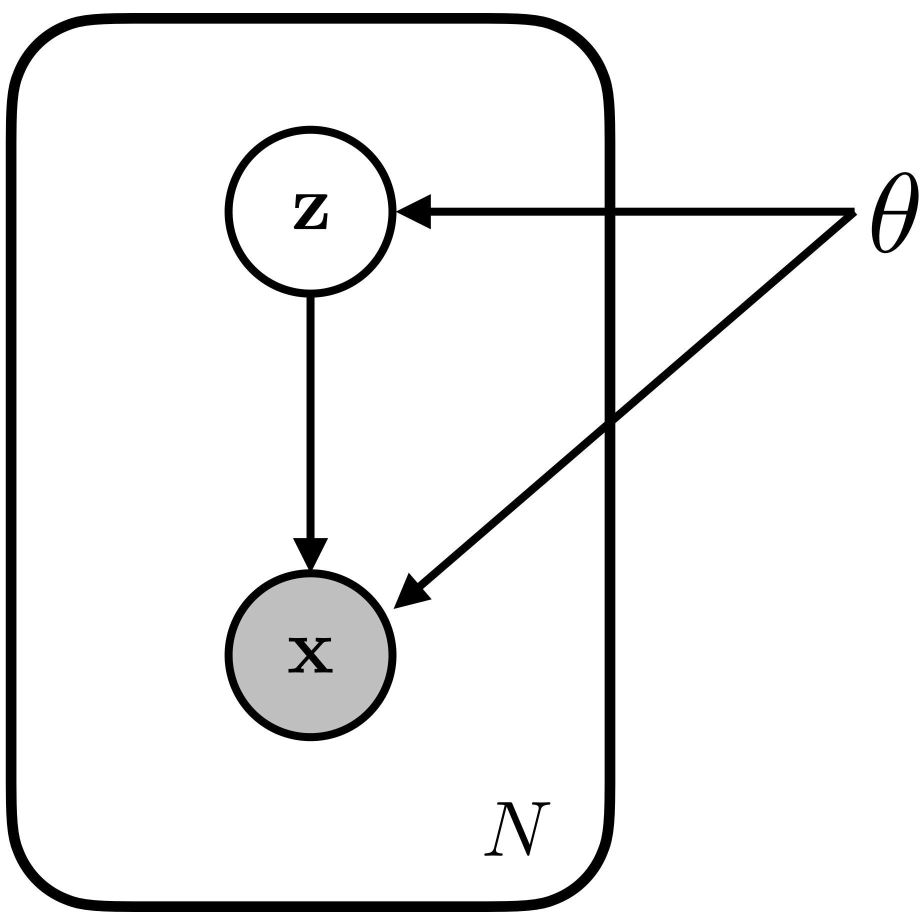
$$\geq \mathbb{E}_q\left[\log \frac{p(x,z)}{q(z)}\right]$$

$$= \underbrace{\mathbb{E}_q[\log p(x,z)] - \mathbb{E}_q[\log q(z)]}_{\text{Evidence Lower bound (ELBO)}}$$

$\mathcal{L}$ provides a lower bound on $\log p(\mathbf{x})$, so we can use $\mathcal{L}$ to (approximately) fit the model

$$\mathscr{L}(x, q; \theta) = \mathbb{E}_{z\sim q}[\log p(x,z) - \log q(z)]$$

Variational Autoencoder (VAE)



Optimize the ELBO

$$\log p(x) \geq \mathbb{E}_q[\log p(x, z)] - \mathbb{E}_q[\log q(z \mid x)]$$

$$= \mathbb{E}_q[\log p(x \mid z) q(z)] - \mathbb{E}_q[\log q(z \mid x)]$$

$$= \mathbb{E}_q[\log p(x \mid z)] - \mathbb{E}_{q(z|x)}[\log(\frac{q(z \mid x)}{q(z)})]$$

$$= \mathbb{E}_q[\log p(x \mid z)] - \mathrm{KL}(q(z \mid x) \,||\, p(z))$$

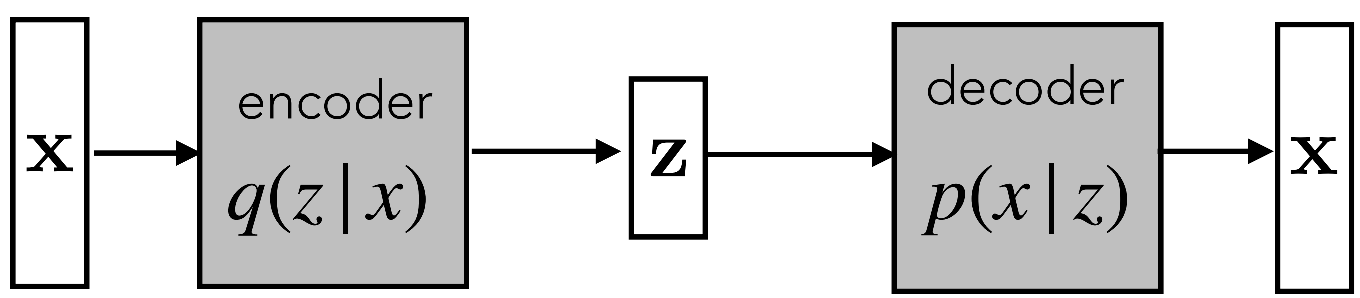
Variational Autoencoder (VAE)

- Optimize the ELBO

$$\log p(x) \geq \mathbb{E}_q[\log p(x|z)] - \mathrm{KL}(q(z|x)||p(z))$$

generative net (decoder) ***inference net (encoder)*** ***prior***

- Approximate the distributions with neural networks



How to backprop through a stochastic random variable?

Reparameterization Trick

- Back-propagation through *stochastic* transformation

example: stochastic backpropagation via "reparameterization trick" Kingma & Welling, 2014; Rezende et al., 2014

$$z \sim \mathcal{N}(\mu, \sigma) \longrightarrow z = \mu + \sigma\epsilon \quad \epsilon \sim \mathcal{N}(0, I)$$

stochastic gradients get backpropagated to σ

More generally

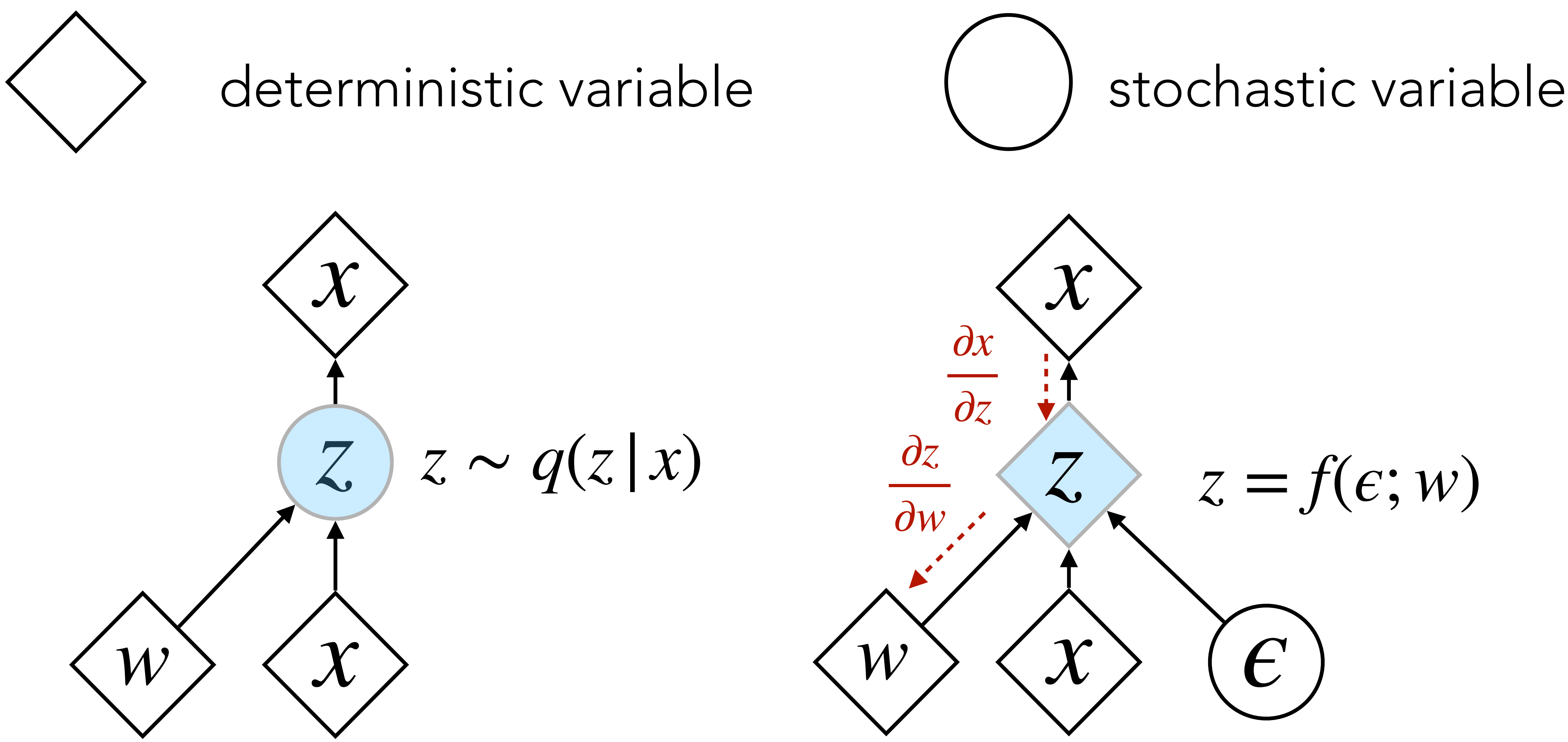
$$\boxed{z = f(\epsilon; \omega)} \qquad \omega = (x, \mu, \sigma)$$

source of randomness

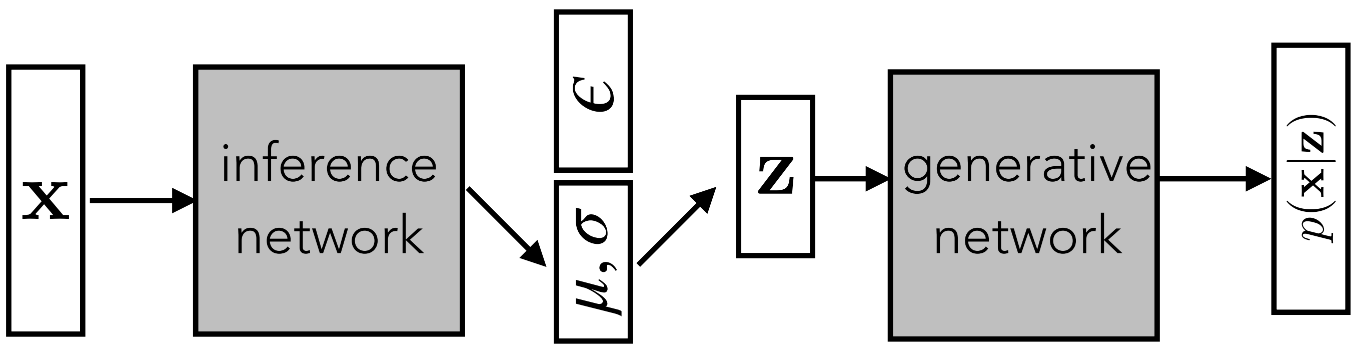
contain inputs and parameters

Reparameterization Trick

- Back-propagation through *stochastic* transformation
- Introduce a new random variable ϵ



Variational Autoencoder (VAE)



- Applications:

Generate samples

Dimension reduction

- Issues: blurring images

$$z = \mu + \sigma\epsilon \qquad \epsilon \sim \mathcal{N}(0, I)$$

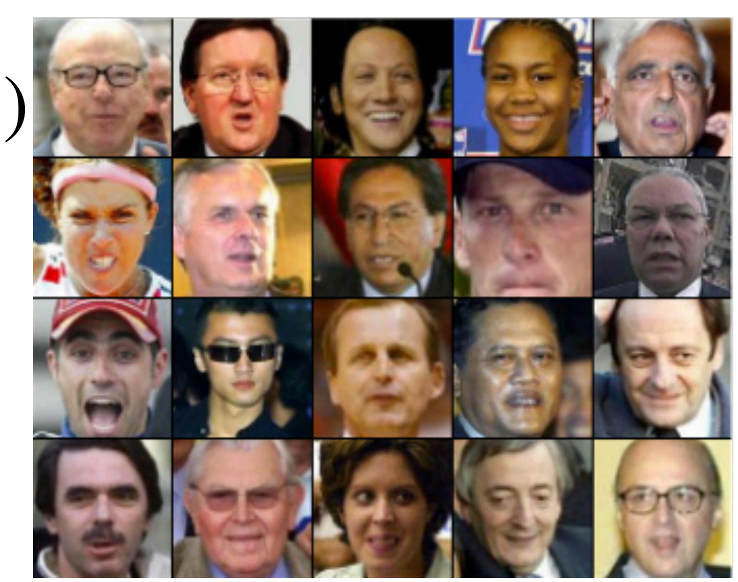
$$\mathbb{E}_q[\log p(x\,|\,z)] - \mathrm{KL}(q(z\,|\,x)\,||\,p(z))$$

$$\downarrow$$

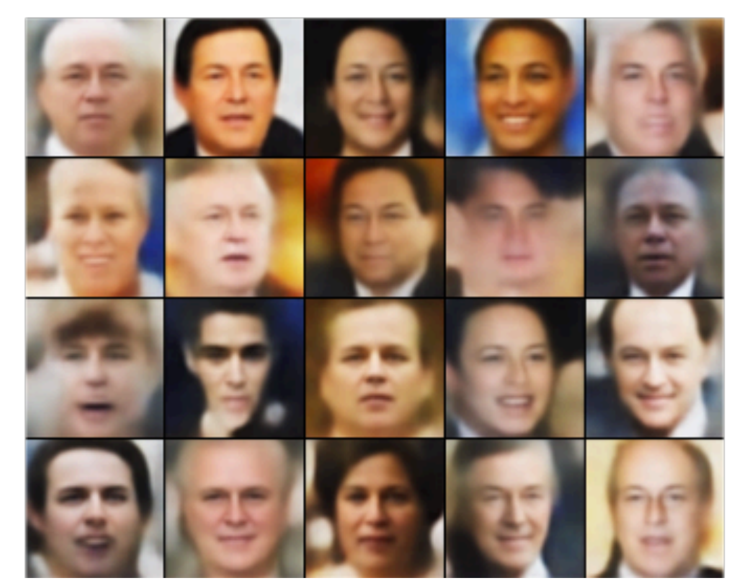
$$||\tilde{x} - x||$$

Euclidean distance is not a good measure

Input



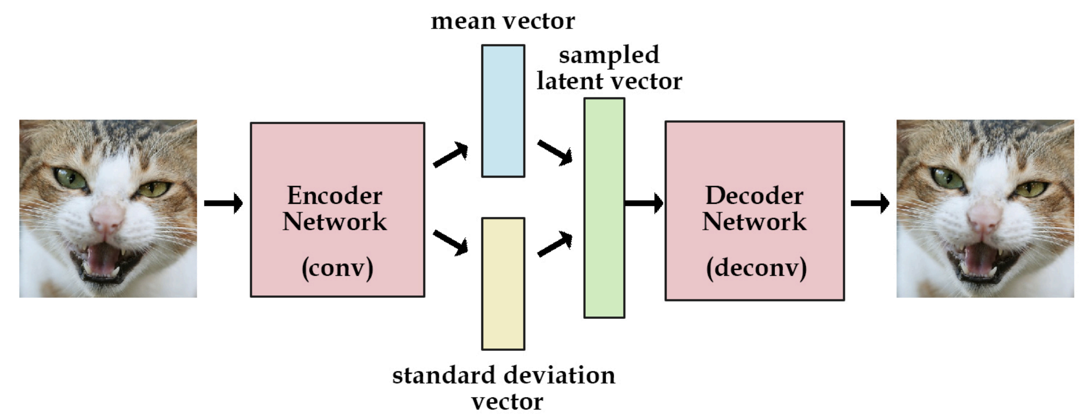
VAE reconstruction



Disentangled Representation

- $\mathbb{E}_q[\log p(x \mid z)] - \mathrm{KL}(q(z \mid x) \mid\mid p(z))$

  reconstruction　　disentanglement



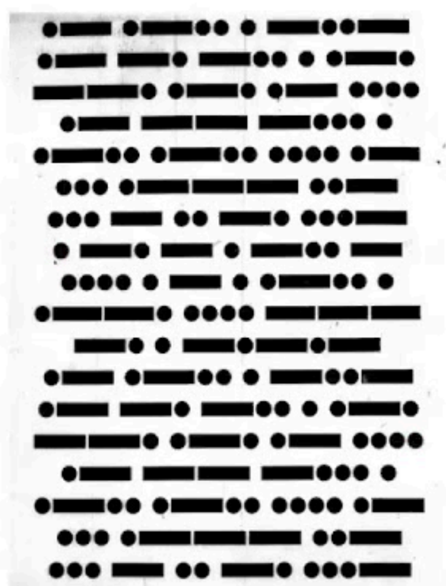
- Vanilla VAE encourages posteriors $q(z \mid x)$ to be close to Gaussian

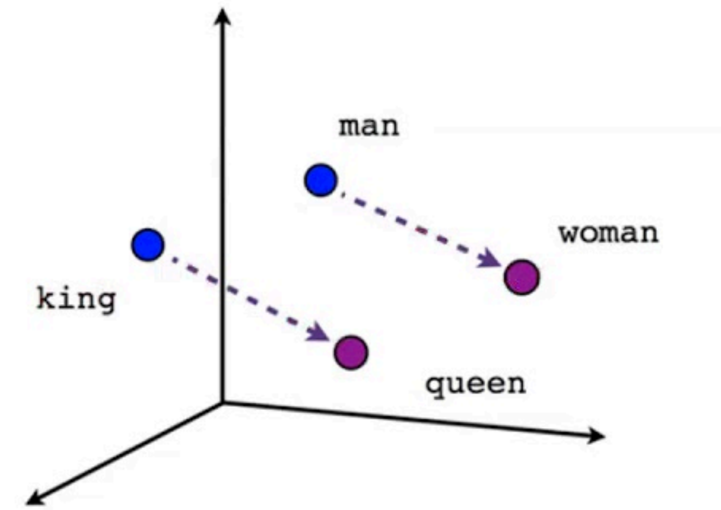
- Promotes disentanglement, but also want reconstruction

$$\mathbb{E}_{q(z|X)}[\log p(X|z)] - \beta D_{KL}[q(z|X)||p(z)]$$

Disentangle latent variables, each image represents a sample from changing the latent vector z.

Discrete Stochastic Operation



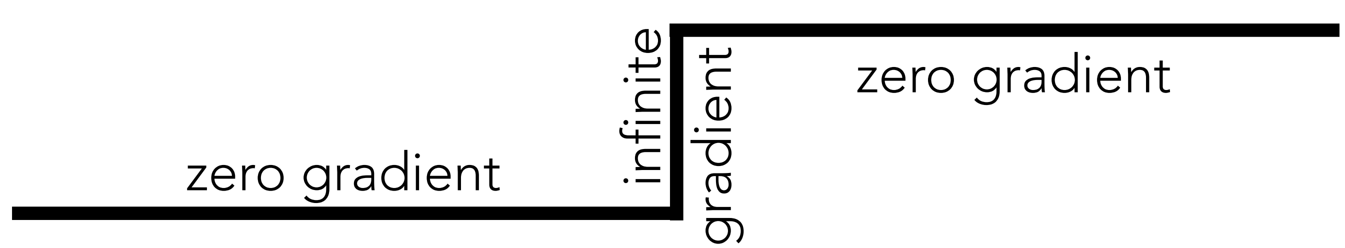


Reparameterization trick does not work for discrete variable

$$z = f(\epsilon; \omega)$$

discrete ***step function***

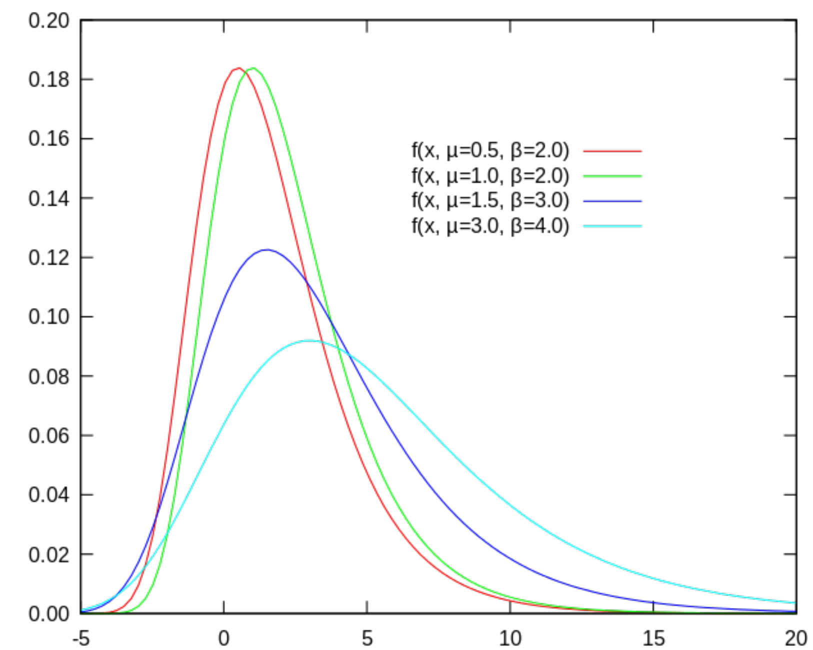
step functions are *non-differentiable*



Gumbel-Soft max

Gumbel distribution

PDF: $\frac{1}{\beta} e^{z+e^{-z}}$ $z = \frac{x-\mu}{\beta}$



Gumbel random variable

$U \sim \text{uniform}(0,1)$ $G = -\log(-\log U)$

Re-parametrize discrete variable

$P(Z=k) = \pi_k$ $Z = \text{argmax}_k(\log \pi_k + G_k)$

Softmax

$$\sigma(\pi) = \frac{e^{\pi_k}}{\sum_k e^{\pi_k}}$$

Gumbel-Soft max

Re-parametrize discrete variable

$$P(Z = k) = \pi_k \qquad Z = \text{argmax}_k(\log \pi_k + G_k)$$

Gumbel distribution

temperature

$$\boxed{Z^\tau = \frac{e^{(\log \pi_k + G_k)/\tau}}{\sum_k e^{(\log \pi_k + G_k)/\tau}}}$$