Generative Models

Third School on Data Science and Machine Learning

Renato Vicente Dec 5, 2024

Our Team

Renato Vicente

Giovanni Tavares

Pedro Pimenta

Bayes Theorem

$q(H) = \theta?$ → $D \in \{H, T\}^n$

$$p(\theta \mid D, I) = \frac{p(D \mid \theta, I)p(\theta \mid I)}{p(D \mid I)}$$

posterior: $p(\theta \mid D, I)$

data-generating distribution: $p(D \mid \theta, I)$

prior: $p(\theta \mid I)$

evidence: $p(D \mid I)$

$$L(\theta) = p(D \mid \theta, I)$$

likelihood: $L(\theta)$

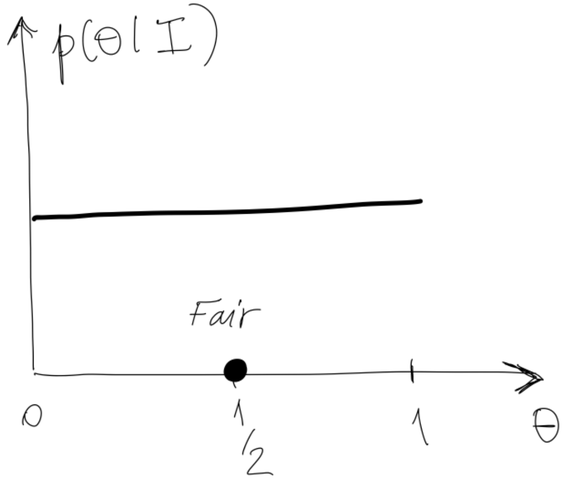
Bayesian Inference

$$p(k\,|\,\theta, I) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

$q(H) = \theta?$ → random variable $K(D)$

Random Experiment:

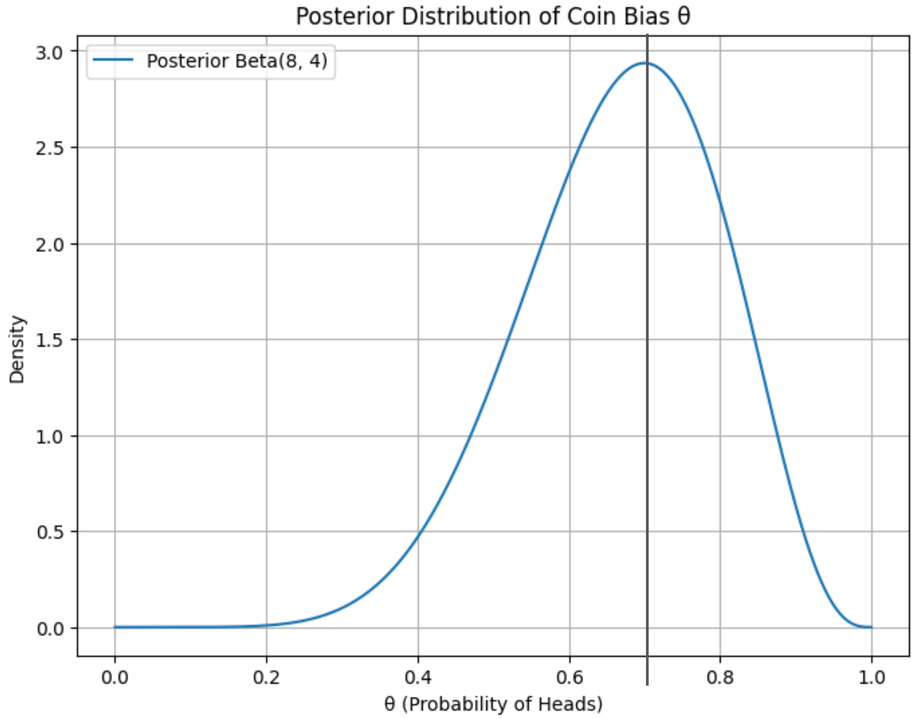
# of Heads in n independent coin tosses



Bayesian Inference

$$p(\theta \mid k, I) \propto \theta^k (1-\theta)^{(n-k)} \qquad 0 \leq \theta \leq 1$$

$q(H) = \theta?$ → random variable $K(HHHHHHHHTTT) = 7$



$$\hat{\theta} = \text{argmax}\, p(\theta \mid K(D), I)$$

Energy Picture

$$p(\theta \mid D, I) = \frac{p(D \mid \theta, I) p(\theta \mid I)}{p(D \mid I)}$$

$$\log p(\theta \mid D, I) = \log p(D \mid \theta, I) + \log p(\theta \mid I) - \log p(D \mid I)$$

$D = \{z_\ell\}$ Independent and identically distributed

$$\log p(\theta \mid D, I) = \log \prod_\ell p(z_\ell \mid \theta, I) + \log p(\theta \mid I) - \log p(D \mid I)$$

Energy Picture

Loss function $$E(\theta) := -\sum_{\ell} \log p(z_\ell \,|\, \theta, I) + \log p(D \,|\, I)$$

Regularisation $$\lambda R(\theta) := -\log p(\theta \,|\, I)$$

Log- Posterior $$L(\theta) := -\log p(\theta \,|\, D, I) = E(\theta) + \lambda R(\theta)$$

$$\hat{\theta} = \operatorname{argmin} L(\theta)$$

Energy Picture: example

$$E(W) = \frac{1}{2\sigma^2} \sum_{\ell=1}^{n} (z_\ell - \mu(x_\ell \,|\, W))^2 - \frac{n}{2} \log 2\pi\sigma^2$$

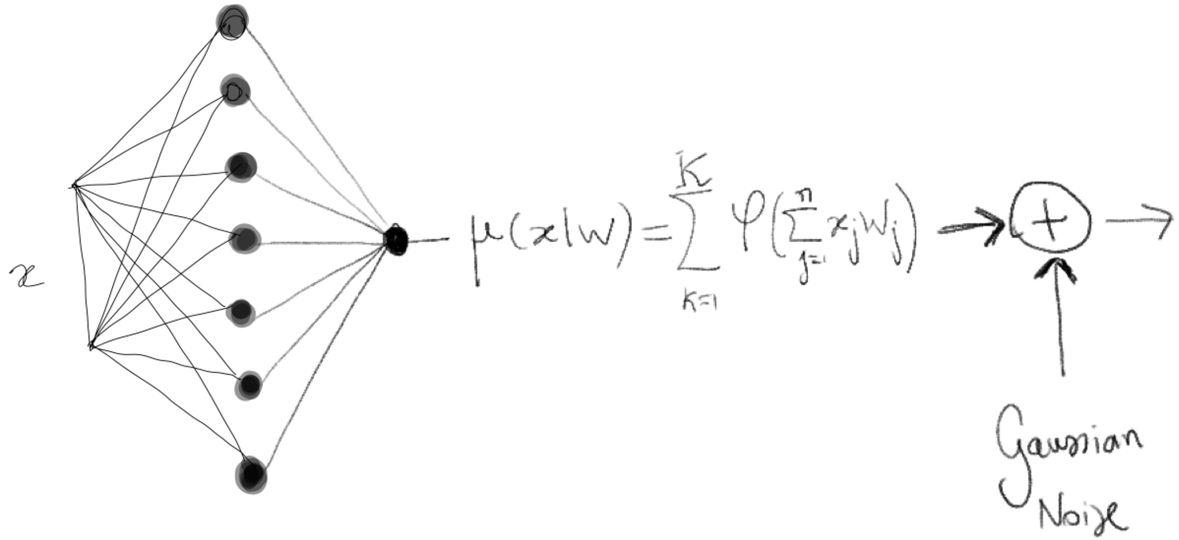
$$\lambda R(W) := -\log \exp \left[-\lambda \sum_j |\, W_j \,| \right]$$

$$L(W) = \frac{1}{2\sigma^2} \sum_{\ell=1}^{n} (z_\ell - \mu(x_\ell \,|\, W))^2 + \lambda \sum_j |\, W_j \,| - \cancel{\frac{n}{2} \log 2\pi\sigma^2}$$

irrelevant

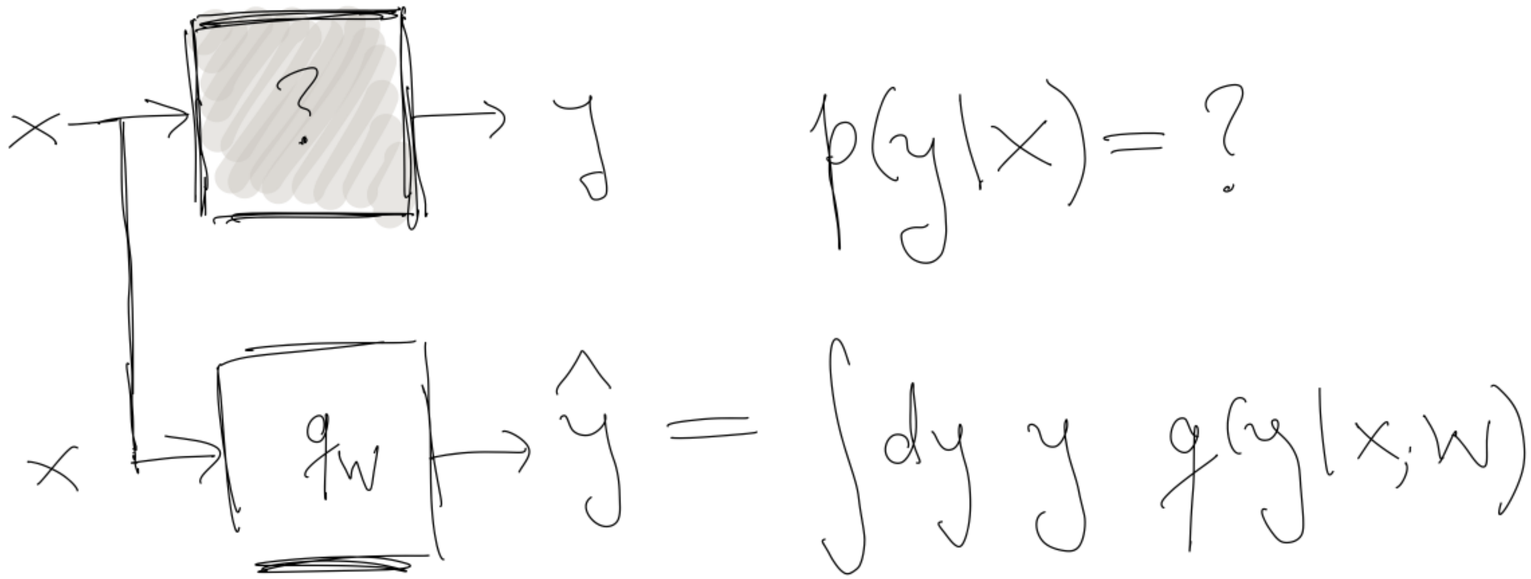
Energy Picture: example

$$L(W) = \frac{1}{2\sigma^2} \sum_{\ell=1}^{n} (z_\ell - \mu(x_\ell \mid W))^2 + \lambda \sum_j |W_j|$$



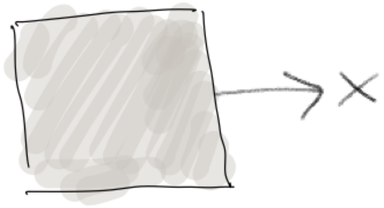
Types of Inference Problems

Supervised Learning



Types of Inference Problems

Unsupervised Learning: Clustering

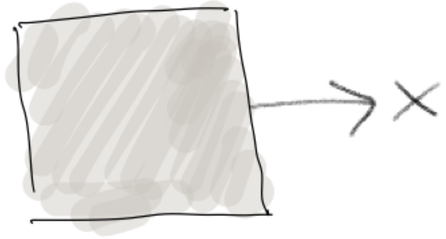


$$p(x) = ?$$

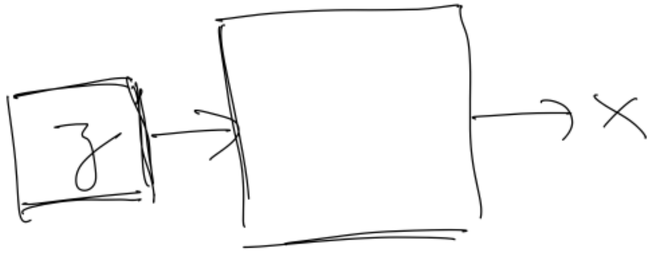
$$q(x|w) = \sum_c \alpha_c \, q(x|c,w)$$

Types of Inference Problems

Unsupervised Learning: Dimensional Reduction



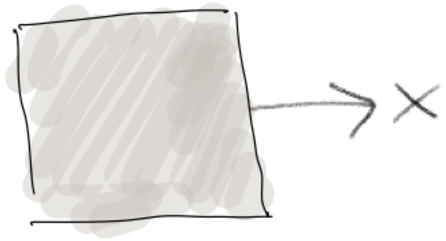
$$p(x) = ?$$



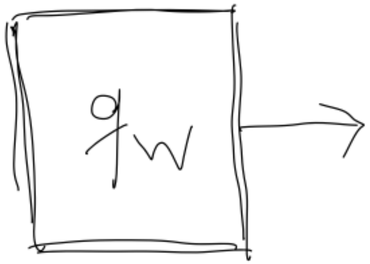
$$q(x|w) = \int dz \; q(x|z,w)\, p(z)$$

Types of Inference Problems

Unsupervised Learning: Sampling



$$p(x) = ?$$



$$\hat{x} \sim q(x|w)$$

1. Learning as Bayesian Inference

Bayes Theorem

$q(H) = \theta?$ → $D \in \{H, T\}^n$

$$p(\theta \mid D, I) = \frac{p(D \mid \theta, I)p(\theta \mid I)}{p(D \mid I)}$$

posterior: $p(\theta \mid D, I)$

data-generating distribution: $p(D \mid \theta, I)$

prior: $p(\theta \mid I)$

evidence: $p(D \mid I)$

$$L(\theta) = p(D \mid \theta, I)$$

likelihood: $L(\theta)$

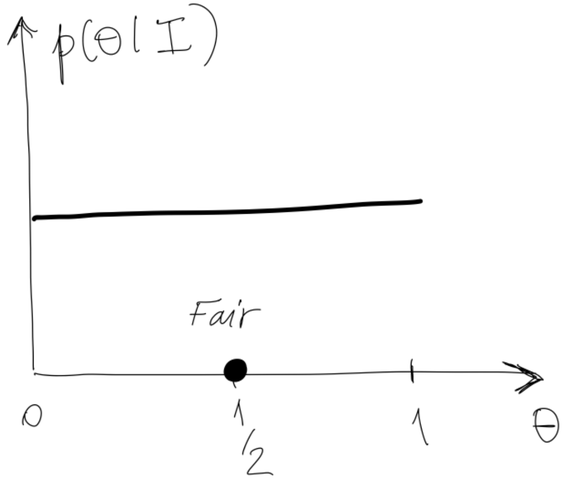
Bayesian Inference

$$p(k\,|\,\theta, I) = \binom{n}{k} \theta^k (1-\theta)^{n-k}$$

$q(H) = \theta?$ → random variable $K(D)$

Random Experiment:

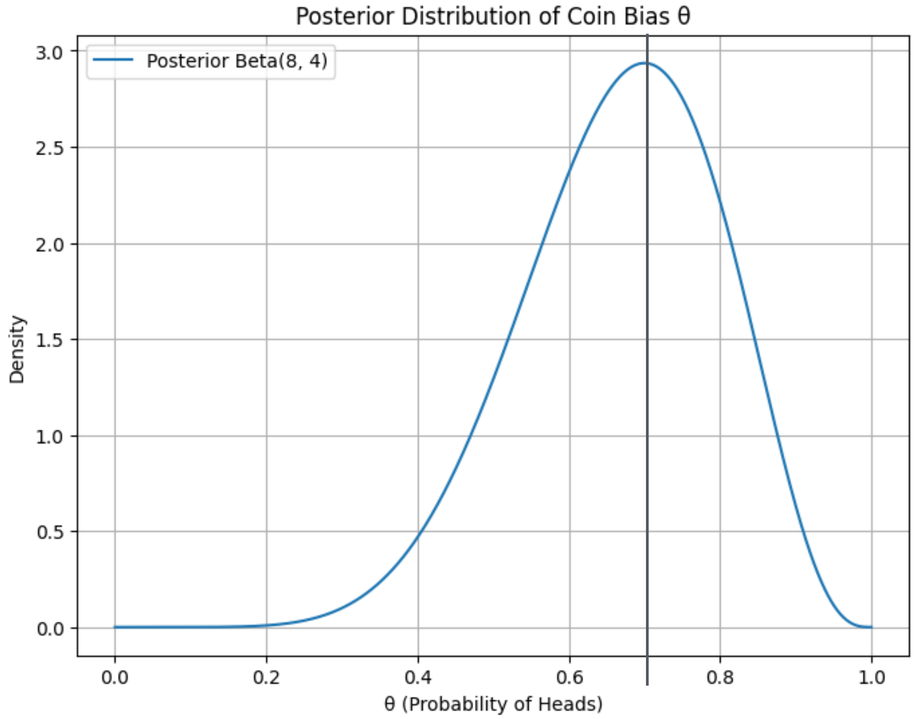
# of Heads in n independent coin tosses



Bayesian Inference

$$p(\theta \mid k, I) \propto \theta^k (1-\theta)^{(n-k)} \qquad 0 \leq \theta \leq 1$$

$q(H) = \theta?$ → random variable $K(HHHHHHHHTTT) = 7$



$$\hat{\theta} = \text{argmax}\, p(\theta \mid K(D), I)$$

Energy Picture

$$p(\theta \mid D, I) = \frac{p(D \mid \theta, I) p(\theta \mid I)}{p(D \mid I)}$$

$$\log p(\theta \mid D, I) = \log p(D \mid \theta, I) + \log p(\theta \mid I) - \log p(D \mid I)$$

$D = \{z_\ell\}$ Independent and identically distributed

$$\log p(\theta \mid D, I) = \log \prod_\ell p(z_\ell \mid \theta, I) + \log p(\theta \mid I) - \log p(D \mid I)$$

Energy Picture

Loss function $$E(\theta) := -\sum_{\ell} \log p(z_\ell \,|\, \theta, I) + \log p(D \,|\, I)$$

Regularisation $$\lambda R(\theta) := -\log p(\theta \,|\, I)$$

Log- Posterior $$L(\theta) := -\log p(\theta \,|\, D, I) = E(\theta) + \lambda R(\theta)$$

$$\hat{\theta} = \operatorname{argmin} L(\theta)$$

Energy Picture: example

$$E(W) = \frac{1}{2\sigma^2} \sum_{\ell=1}^{n} (z_\ell - \mu(x_\ell \,|\, W))^2 - \frac{n}{2} \log 2\pi\sigma^2$$

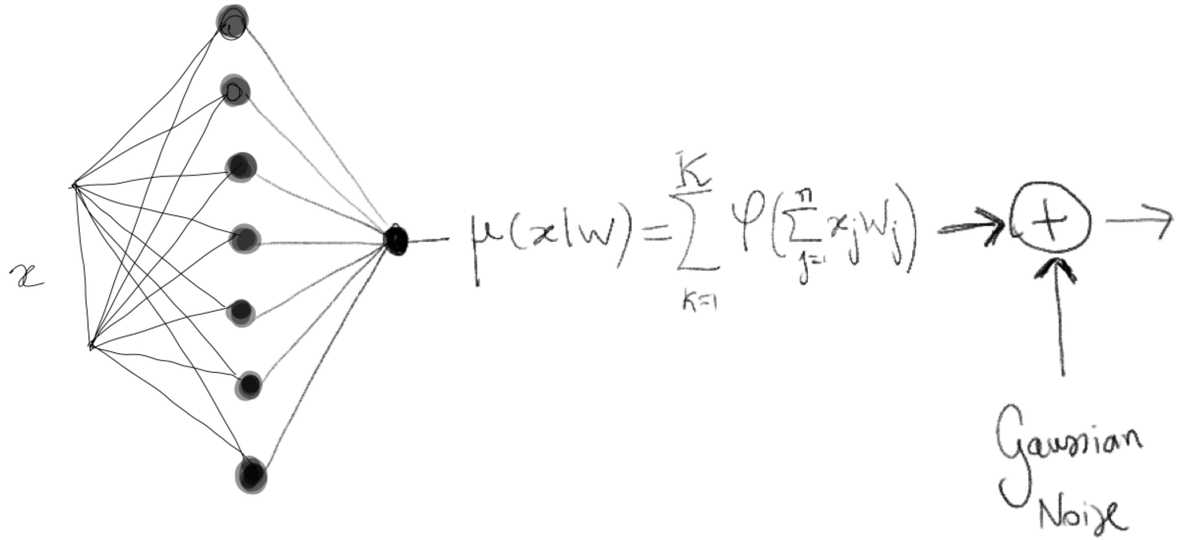
$$\lambda R(W) := -\log \exp \left[-\lambda \sum_j |\, W_j \,| \right]$$

$$L(W) = \frac{1}{2\sigma^2} \sum_{\ell=1}^{n} (z_\ell - \mu(x_\ell \,|\, W))^2 + \lambda \sum_j |\, W_j \,| - \cancel{\frac{n}{2} \log 2\pi\sigma^2}$$

irrelevant

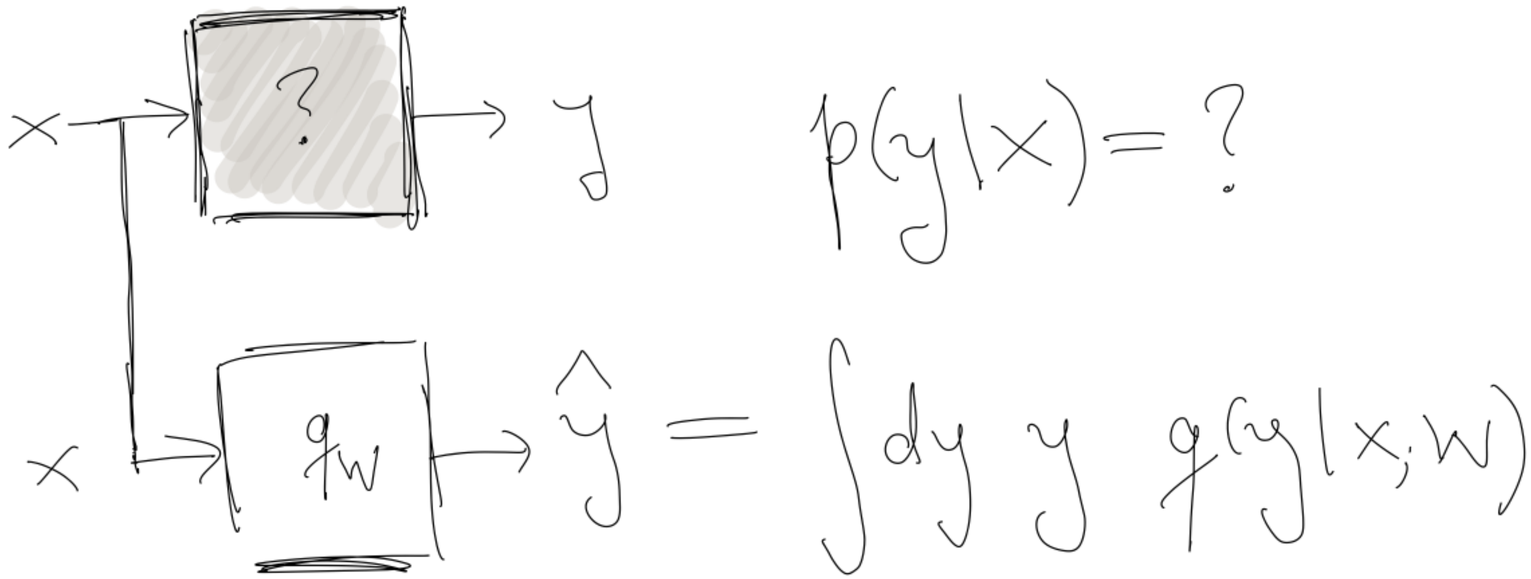
Energy Picture: example

$$L(W) = \frac{1}{2\sigma^2} \sum_{\ell=1}^{n} (z_\ell - \mu(x_\ell \mid W))^2 + \lambda \sum_j |W_j|$$



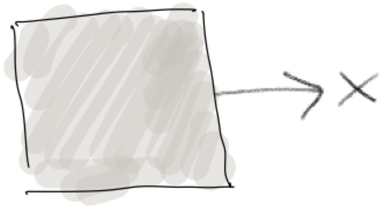
Types of Inference Problems

Supervised Learning



Types of Inference Problems

Unsupervised Learning: Clustering

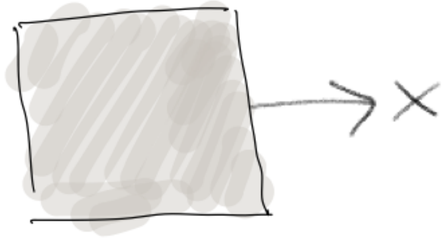


$$p(x) = ?$$

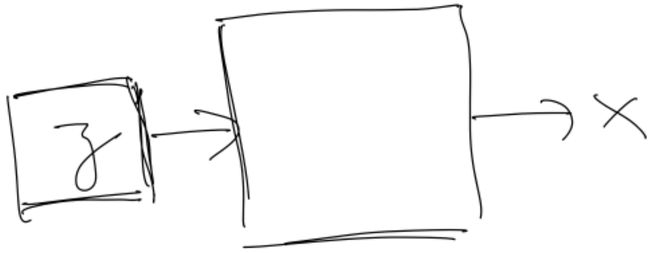
$$q(x|w) = \sum_c \alpha_c \, q(x|c,w)$$

Types of Inference Problems

Unsupervised Learning: Dimensional Reduction



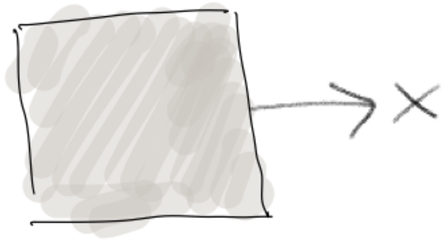
$$p(x) = ?$$



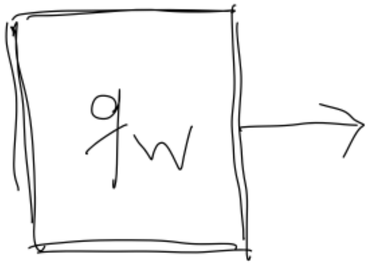
$$q(x|w) = \int dz \; q(x|z,w)\, p(z)$$

Types of Inference Problems

Unsupervised Learning: Sampling



$$p(x) = ?$$

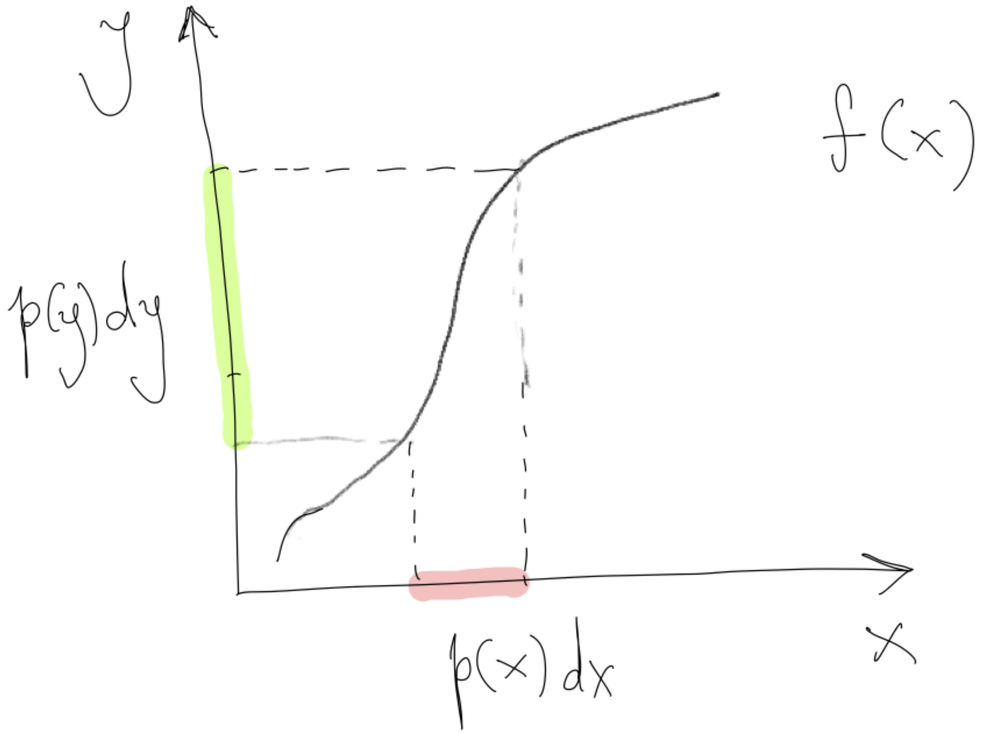


$$\hat{x} \sim q(x|w)$$

2. Sampling

Transforming Variables

1 dimension



$$p(y) = p(x)\left|\frac{dx}{dy}\right|$$

something we know how to sample

1. $x \sim p(x)$
2. $y = f(x)$

Transforming Variables

n dimensions

$$p(y_1, y_2, \cdots, y_n) = p(x_1, x_2, \cdots, x_n) \left| \frac{\partial(x_1, x_2, \cdots, x_n)}{\partial(y_1, y_2, \cdots, y_n)} \right|$$

Jacobian determinant

$$J = \frac{\partial(x_1, x_2, \ldots, x_n)}{\partial(y_1, y_2, \ldots, y_n)} = \begin{pmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} & \cdots & \frac{\partial x_1}{\partial y_n} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} & \cdots & \frac{\partial x_2}{\partial y_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial x_n}{\partial y_1} & \frac{\partial x_n}{\partial y_2} & \cdots & \frac{\partial x_n}{\partial y_n} \end{pmatrix}$$

1. $\mathbf{x} \sim p(\mathbf{x})$
2. $\mathbf{y} = \mathbf{F}(\mathbf{x})$

Markov Chain Monte Carlo

$$p(x_1, x_2, \cdots, x_n) = \frac{\tilde{p}(x_1, x_2, \cdots, x_n)}{Z}$$

$$\tilde{p}(x_1, x_2, \cdots, x_n) = e^{-E(x_1, x_2, \cdots, x_n)}$$

Easy!

$$Z = \sum_{x_1, x_2, \cdots, x_n} e^{-E(x_1, x_2, \cdots, x_n)}$$

Difficult!

Markov Chain Monte Carlo

$$p(x_1, x_2, \cdots, x_n) = \frac{\tilde{p}(x_1, x_2, \cdots, x_n)}{Z}$$

$$\tilde{p}(x_1, x_2, \cdots, x_n) = e^{-E(x_1, x_2, \cdots, x_n)}$$

Easy!

$$Z = \sum_{x_1, x_2, \cdots, x_n} e^{-E(x_1, x_2, \cdots, x_n)}$$

Difficult!

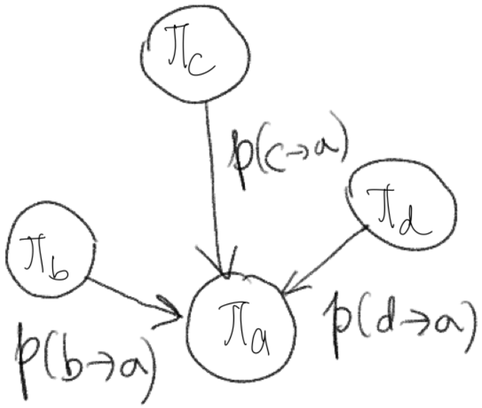
partition function

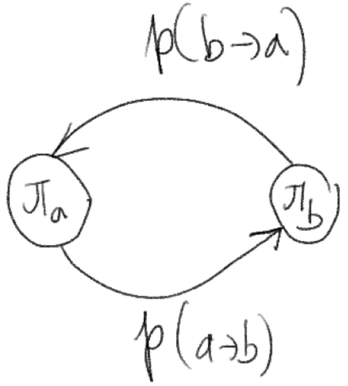
Markov Chain Monte Carlo

Detailed Balance

$$a = (a_1, a_2, \cdots, a_n)$$ microstate a

$$\pi_{t+1}(a) = \sum_b p(b \to a)\pi_t(b)$$





The equilibrium distribution satisfies:

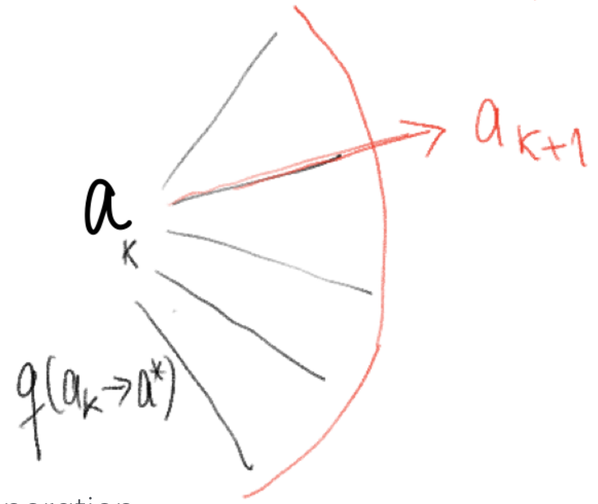
$$\pi(a)p(a \to b) = \pi(b)p(b \to a)$$

Markov Chain Monte Carlo

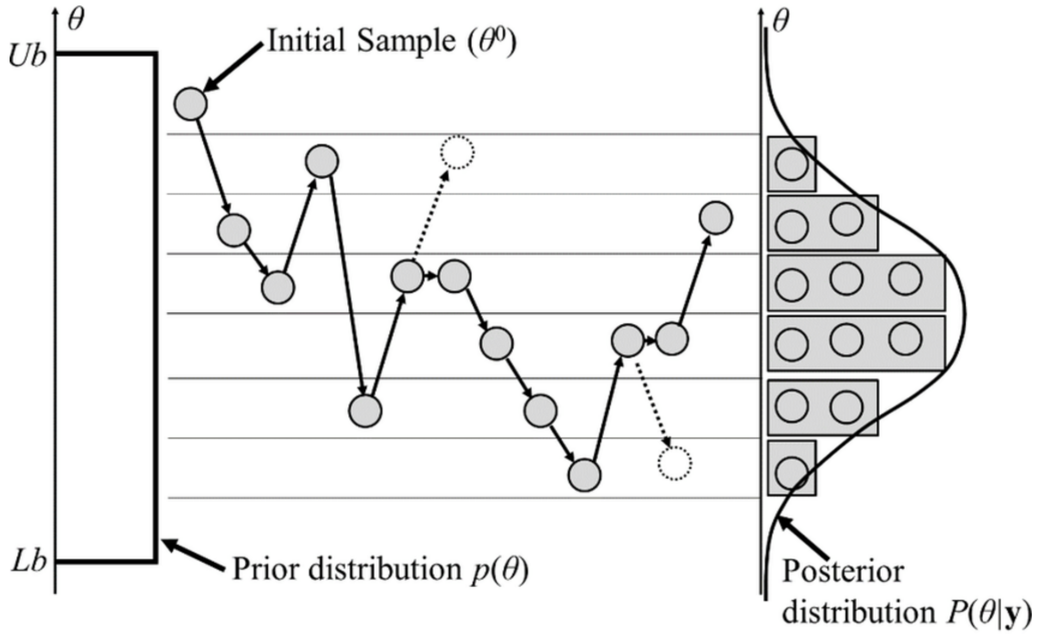
Metropolis-Hastings

acceptance probability

$$A(a^*, a_k) = \min\left\{1, \frac{\tilde{p}(a^*)\, q(a_k \to a^*)}{\tilde{p}(a_k)\, q(a^* \to a_k)}\right\}$$



candidate generation

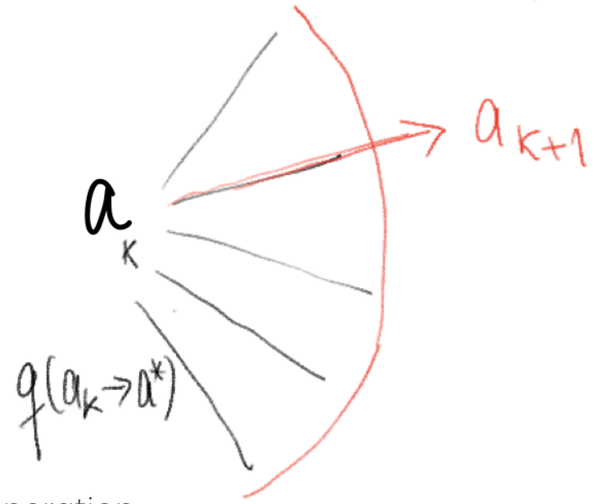


Markov Chain Monte Carlo

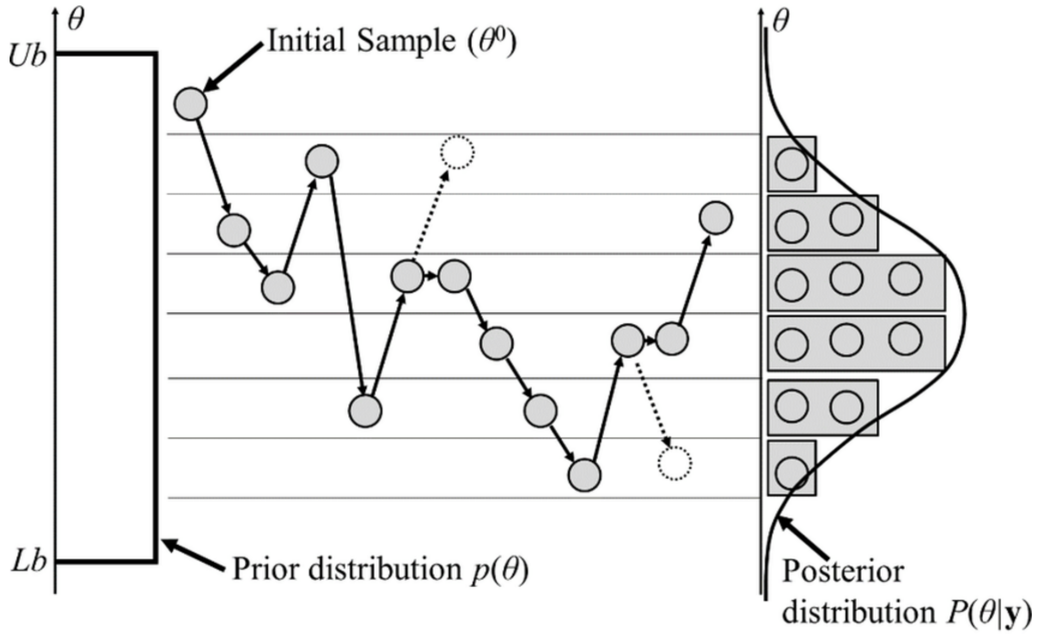
Metropolis-Hastings

acceptance probability

$$A(a^*, a_k) = \min\left\{1, \frac{\tilde{p}(a^*)\, q(a_k \to a^*)}{\tilde{p}(a_k)\, q(a^* \to a_k)}\right\}$$



candidate generation



Gibbs Sampling

Input: Initial values $\{z_i : i \in 1, \ldots, M\}$

Conditional distributions $\{p(z_i|\{z_{j\neq i}\}) : i \in 1, \ldots, M\}$

Number of iterations T

Output: Final values $\{z_i : i \in 1, \ldots, M\}$

---

for $\tau \in \{1, \ldots, T\}$ **do**

  for $i \in \{1, \ldots, M\}$ **do**

    $z_i \sim p(z_i|\{z_{j\neq i}\})$

  end for

end for

return $\{z_i : i \in 1, \ldots, M\}$

Langevin Sampling

Input: Initial value $\mathbf{x}^{(0)}$

Probability density $p(\mathbf{x}, \mathbf{w})$

Learning rate parameter η

Number of iterations T

Output: Final value $\mathbf{x}^{(T)}$

$\mathbf{x} \leftarrow \mathbf{x}_0$

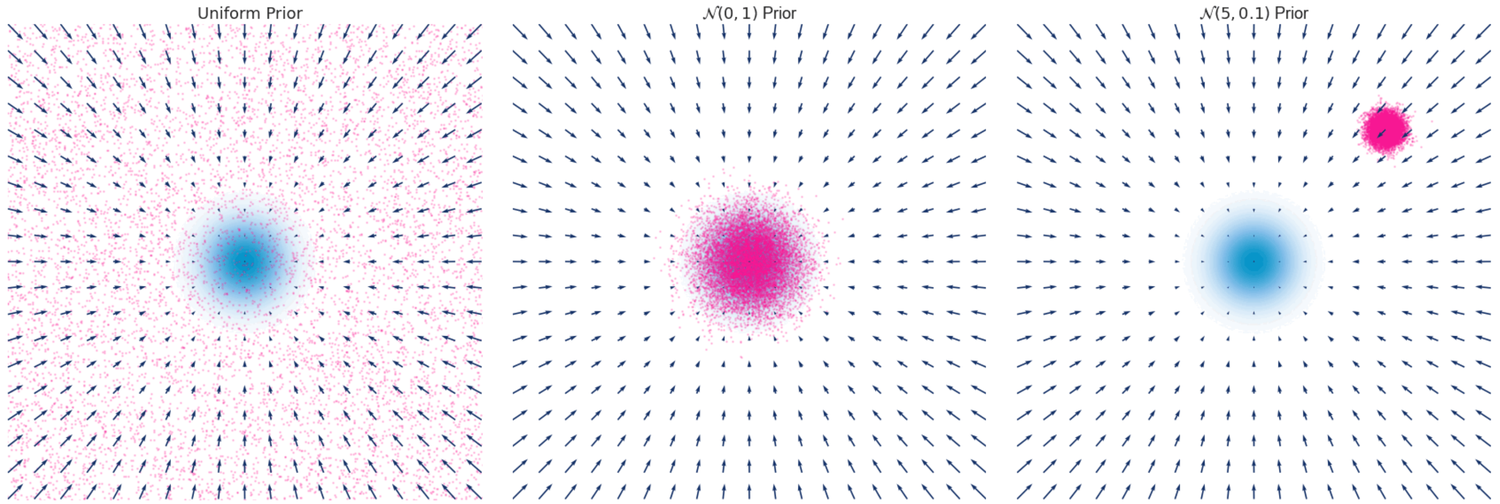
for $\tau \in \{1, \ldots, T\}$ **do**

  $\boldsymbol{\epsilon} \sim \mathcal{N}(\boldsymbol{\epsilon}|\mathbf{0}, \mathbf{I})$

  $\mathbf{x} \leftarrow \mathbf{x} + \eta \nabla_{\mathbf{x}} \ln p(\mathbf{x}, \mathbf{w}) + \sqrt{2\eta}\boldsymbol{\epsilon}$

end for

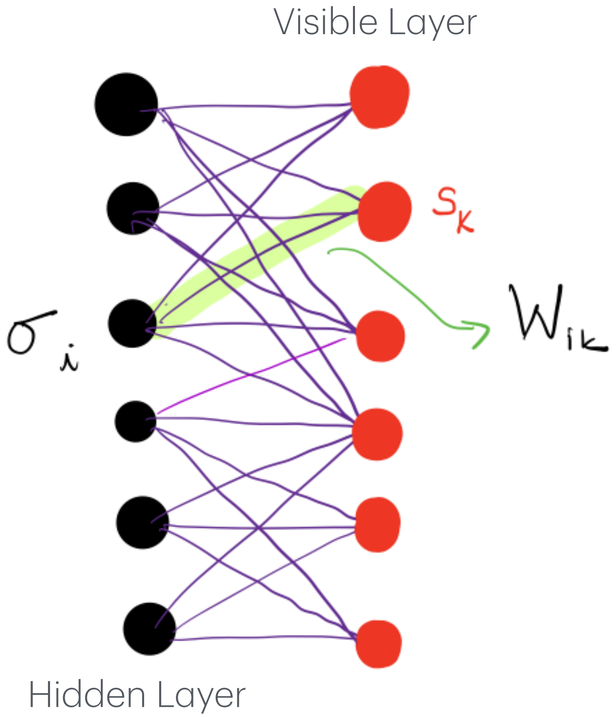
return $\mathbf{x}$ `// Final value` $\mathbf{x}^{(T)}$



https://perceptron.blog/langevin-dynamics/

3. Energy Based Models

Restricted Boltzmann Machines



$$q(\mathbf{S} \mid W) = \sum_{\sigma} q(\mathbf{S}, \sigma \mid W)$$

$$q(S, \sigma \mid W) = \frac{e^{-\beta \sum_{i,k} S_k W_{ki} \sigma_i}}{Z}$$

Task: Find W from independent samples of an unknown distribution $D = \{\mathbf{S}_\ell\}_{\ell=1}^{m} \sim p^m$

Kullback-Leibler Divergence

$$KL(p||q) = \mathbb{E}_{X\sim p}\left[\log\left(\frac{p(X)}{q(X)}\right)\right]$$

$$= \sum_x p(x)\log\frac{1}{q(x)} - \sum_x p(x)\log\frac{1}{p(x)}$$

$$= H[p,q] - H[p]$$

Information inefficiency of using q as a model of a process generated by p

Training RBMs

$$\Delta W = -\eta \frac{\partial}{\partial W} KL(p \,||\, q(W))$$

$$= \eta \sum_S \frac{p(S)}{q(S \,|\, W)} \frac{\partial q(S \,|\, W)}{\partial W}$$

with a couple of simple algebra steps:

$$\Delta W_{ik} = \eta\beta \left[\sum_{\sigma S} \sigma_i S_k q(\sigma \,|\, S; W) p(S) - \sum_{\sigma S} \sigma_i S_k q(\sigma, S \,|\, W) \right]$$

Contrastive Divergence

Training RBMs

$$\Delta W_{ik} = \eta\beta \left[\mathbb{E}_{data}[\sigma_i S_k] - \mathbb{E}_{model}[\sigma_i S_k]\right]$$

sample $q(\sigma \mid S_\ell; W)$

sample $q(\sigma, S \mid W)$

using Gibbs sampling

Training RBMs

$$\Delta W_{ik} = \eta\beta \left[\mathbb{E}_{data}[\sigma_i S_k] - \mathbb{E}_{model}[\sigma_i S_k]\right]$$

sample $q(\sigma \mid S_\ell; W)$

sample $q(\sigma, S \mid W)$

using Gibbs sampling

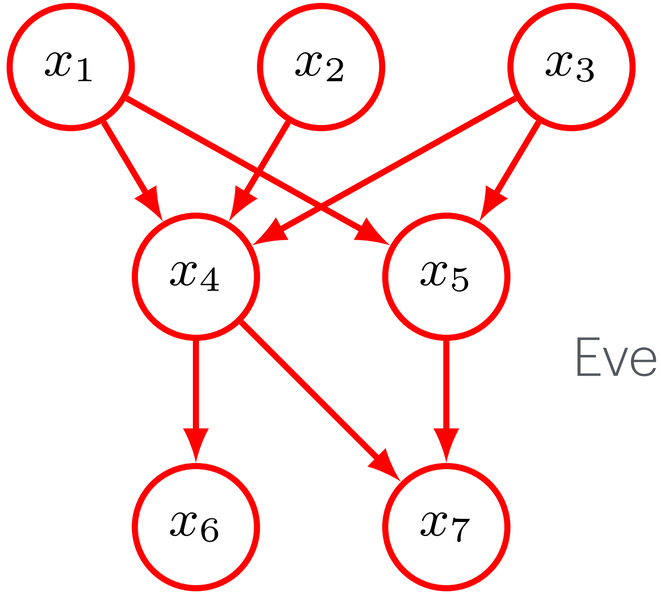
4. Autoregressive Models

Joint Distribution Factorisation

$$p(x_1, x_2, \cdots, x_K) = p(x_k \,|\, x_1, \cdots, x_{K-1}) \cdots p(x_2 \,|\, x_1) p(x_1)$$

A joint distribution can be represented by products of conditional distributions, factorisation. There are K! possible factorisations of this kind.

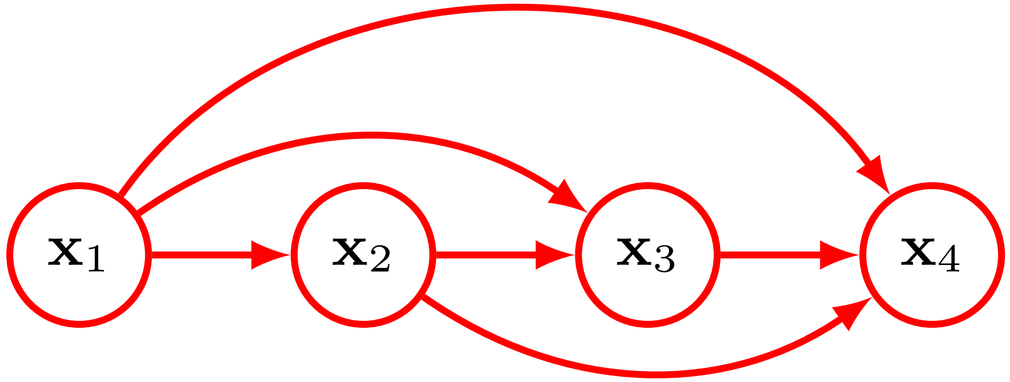
Graphical Models



Every factorisation can be uniquely represented by a directed graph.

$$p(x_1, x_2, \cdots, x_7) = p(x_1)p(x_2)p(x_3)p(x_4 \mid x_1, x_2, x_3)p(x_5 \mid x_1, x_3)p(x_6 \mid x_4)p(x_7 \mid x_4, x_5)$$

Autoregressive Models

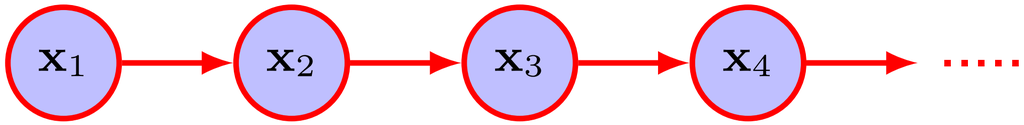


An autoregressive model is structured to distinguish between “past" and “future”.

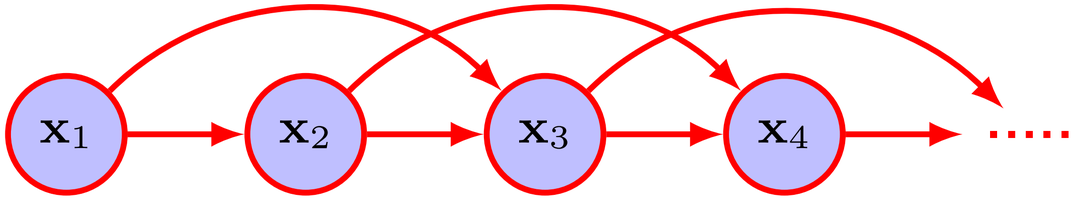
Indices may be interpreted as “time”, and causality requires that a variable indexed k depends only on variables with indices j<k.

$$p(x_1, x_2, \cdots, x_K) = \prod_{\ell=1}^{K} p(x_\ell \,|\, x_1, \cdots, x_{K-1})$$

Markov Chains



$$p(x_1, x_2, \cdots, x_K) = \prod_{\ell=1}^{K} p(x_\ell \,|\, x_{\ell-1})$$

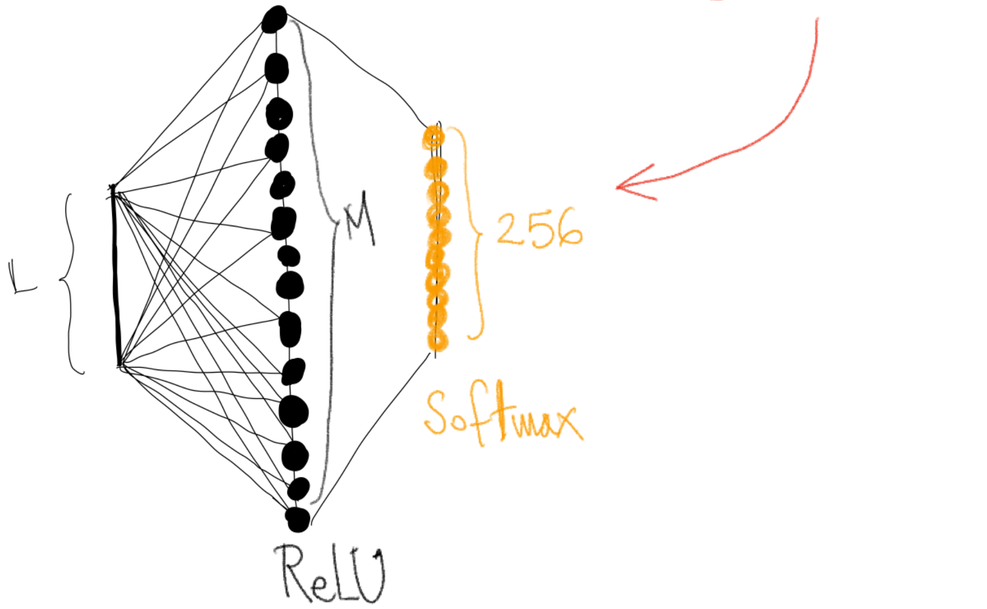


$$p(x_1, x_2, \cdots, x_K) = \prod_{\ell=1}^{K} p(x_\ell \,|\, x_{\ell-1}, x_{\ell-2})$$

Order-k Markov Chains are instances of Finite Memory Autoregressive Models.

Modelling Conditionals with a NN

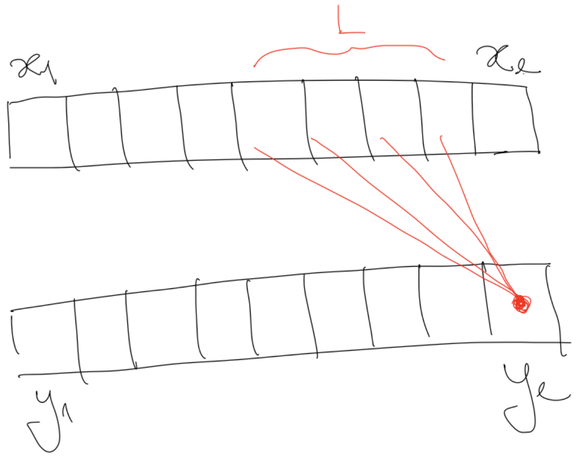
$$q(\mathbf{x}) = q(x_1)\, q(x_2|x_1) \cdots q(x_L|x_{L-1}, \cdots, x_1) \prod_{\ell=L+1}^{m} q(x_\ell | x_{\ell-1}, \cdots x_{\ell-L})$$



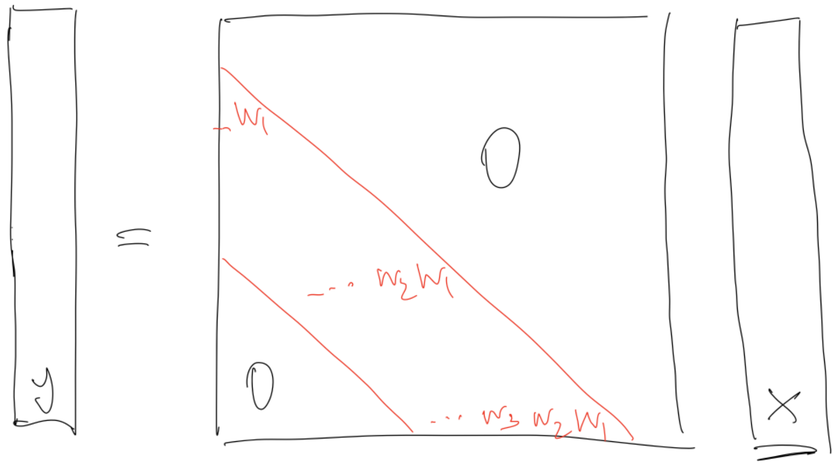
$$[x_{\ell-1}, \cdots, x_{\ell-L}] \rightarrow \text{Linear}(L, M) \rightarrow \text{ReLU} \rightarrow \text{Linear}(M, 256) \rightarrow \text{Softmax}$$

An example architecture designed to learn a joint distribution of discrete variables with 256 levels

Causal Convolution Layer 1D



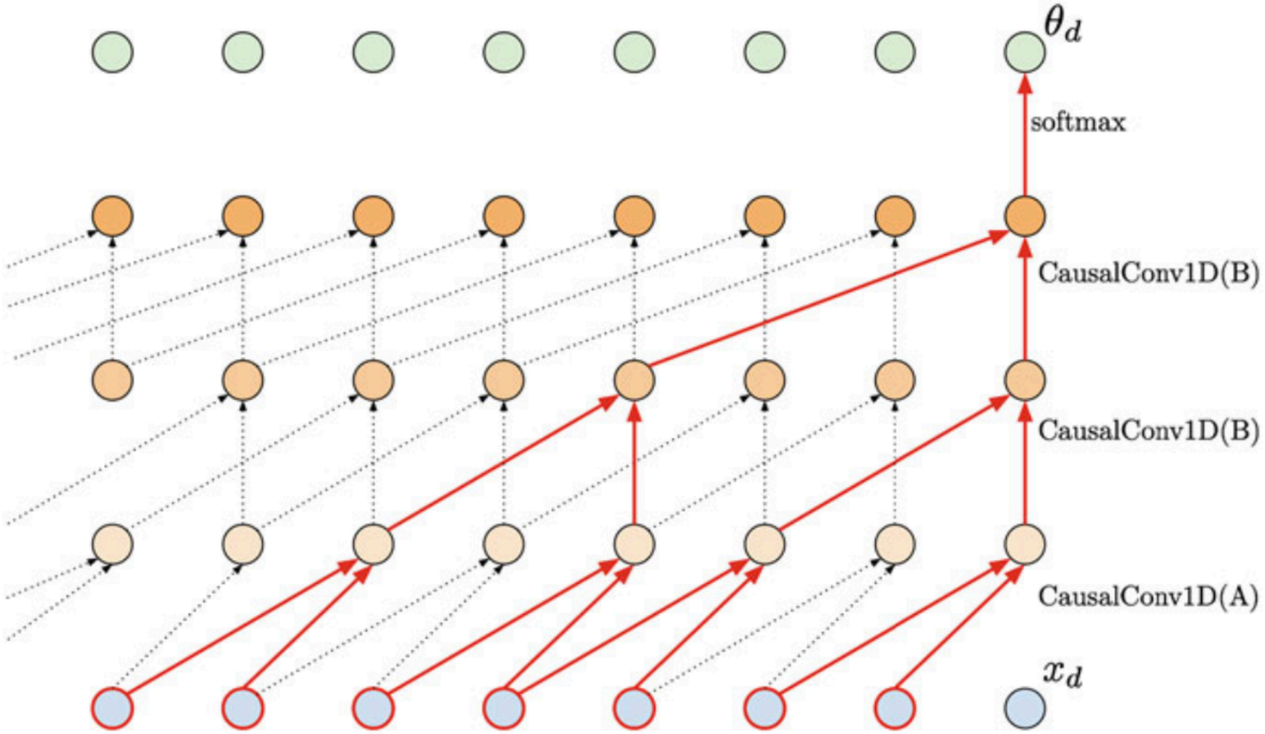
$$y_L = w_1 x_{L-1} + w_2 x_{L-2} + \cdots + w_L x_{L-L}$$



y= CausalConv1D(x)

Conditionals with CausalConv1D

$$q(\mathbf{x}) = q(x_1)\, q(x_2|x_1) \cdots q(x_L|x_{L-1}, \ldots, x_1) \prod_{\ell=L+1}^{n} q(x_\ell | x_{\ell-1}, \ldots x_{\ell-L})$$



Again, an example architecture designed to learn a joint distribution of discrete variables with 256 levels

Likelihood

$$p(x_d|\mathbf{x}_{<d}) = \text{Categorical}\,(x_d|\theta_d\,(\mathbf{x}_{<d}))$$

$$= \prod_{l=1}^{L} \left(\theta_{d,l}\right)^{[x_d=l]},$$

CausalConv network

$$\ln p(\mathcal{D}) = \ln \prod_n p(\mathbf{x}_n)$$

$$= \sum_n \ln p(\mathbf{x}_n)$$

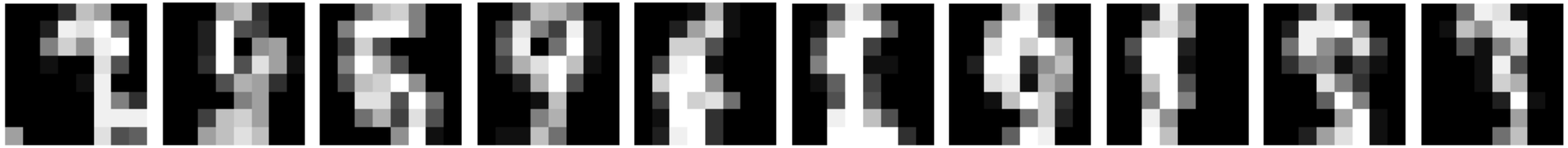
$$= \sum_n \ln \prod_d p(x_{n,d}|\mathbf{x}_{n,<d})$$

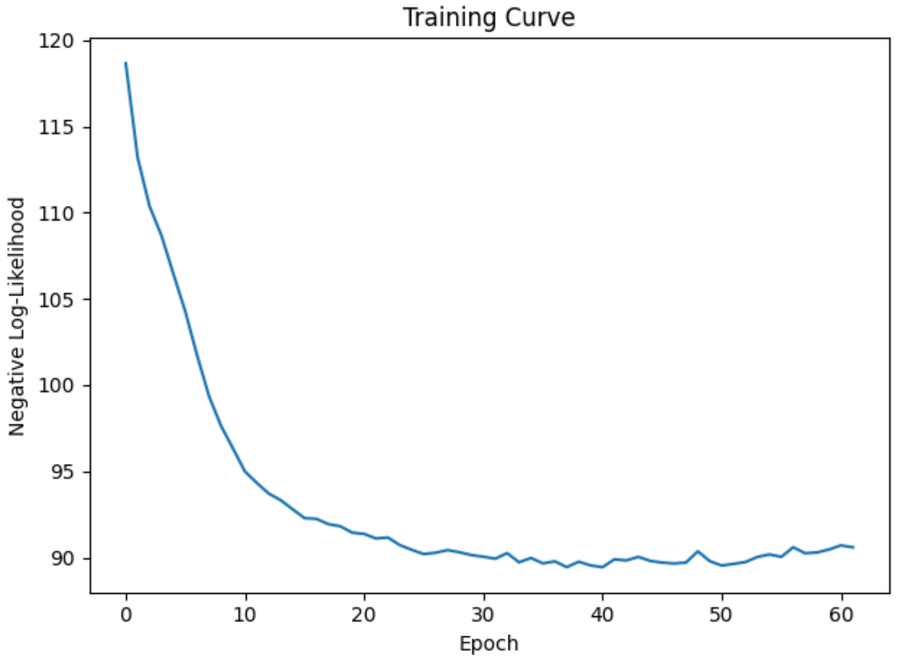
$$= \sum_n \left(\sum_d \ln p(x_{n,d}|\mathbf{x}_{n,<d}) \right)$$

$$= \sum_n \left(\sum_d \ln \text{Categorical}\,(x_d|\theta_d\,(\mathbf{x}_{<d})) \right)$$

$$= \sum_n \left(\sum_d \left(\sum_{l=1}^{L} [x_d = l] \ln \theta_d\,(\mathbf{x}_{<d}) \right) \right).$$

Results





Training with 1500 images of size 8x8 for 60 epochs.

5. Flow Based Models

Variables Transformation Revisited

$$p(x) = \frac{p\left(z = f^{-1}(x)\right)}{\left|J_f(x)\right|}$$

We can start with a simple distribution: $\pi(z_0) = N(z_0 \,|\, 0; \mathbb{I})$

Then successively transform variables: $f(z_0) = f_K \circ f_{K-1} \circ f_{K-2} \circ \cdots f_1(z_0)$

The right transformation can be learned from data.

Likelihood

$$f(z_0) = f_K \circ f_{K-1} \circ f_{K-2} \circ \cdots f_1(z_0)$$

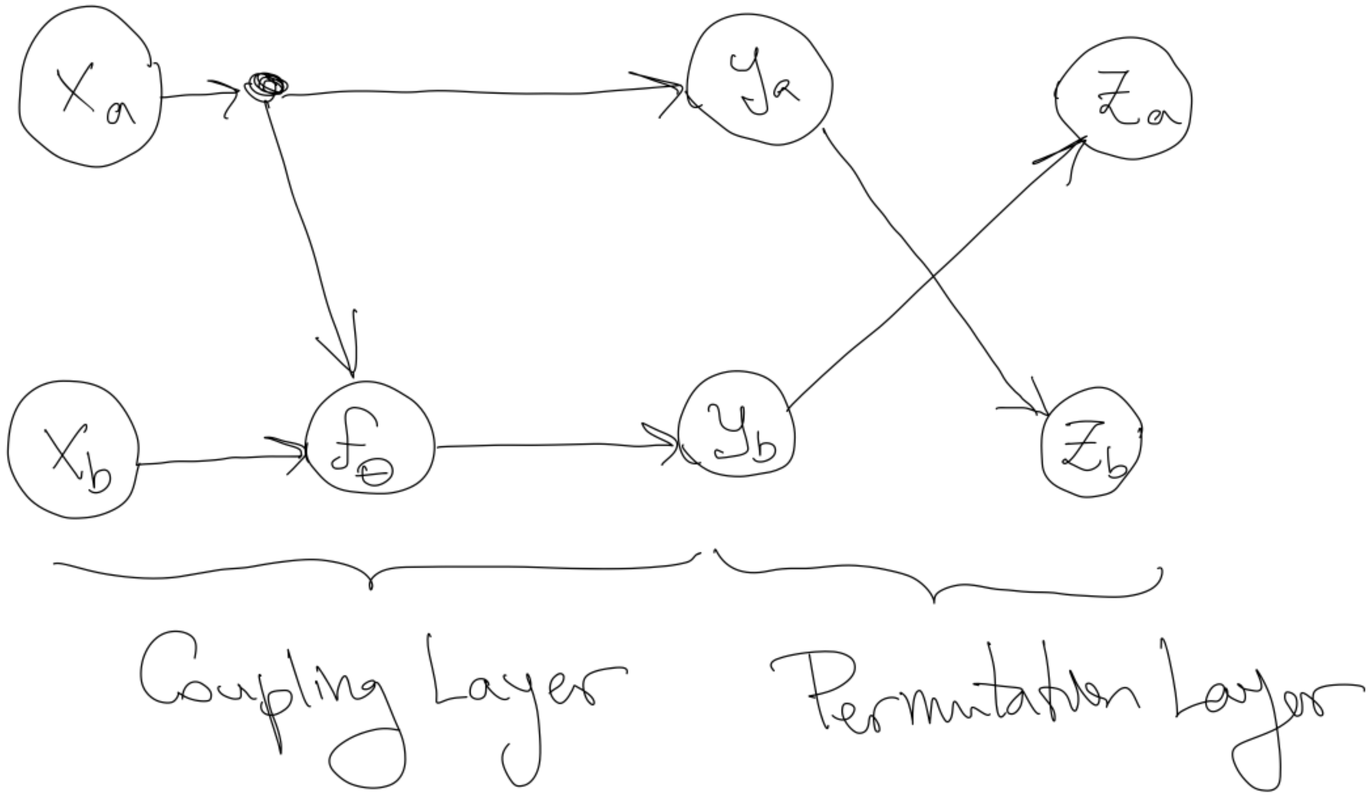
$$q(x) = \pi(z_0 = f^{-1}(x)) \prod_{\ell=1}^{K} \left| \text{Det} \frac{\partial f_\ell(z_{\ell-1})}{\partial z_{\ell-1}} \right|^{-1}$$

The Log-likelihood can be written as (making model parameterisation explicit):

$$\log q(x \,|\, \theta) = -\frac{1}{2} \left\| f^{-1}(x \,|\, \theta) \right\|^2 + \text{const} - \sum_{\ell=1}^{K} \log \left| \text{Det} \frac{\partial f_\ell(z_{\ell-1} \,|\, \theta)}{\partial z_{\ell-1}} \right|$$

We seek invertible transformations with a tractable Jacobean.

Real-Valued Non-volume Preserving Flows



$$x = [x_a, x_b]$$

$$y_a = x_a$$

$$y_b = e^{S(x_a)} \odot x_b + T(x_a)$$

Scaling

Translation

Real-Valued Non-volume Preserving Flows

1 Invertible by Design

$$x_a = y_a$$

$$x_b = \left[y_b - T(y_a)\right] \odot e^{-S(x_a)}$$

2 With a tractable Jacobean

$$\mathbf{J} = \begin{bmatrix} \mathbf{I}_{d\times d} & \mathbf{0}_{d\times(D-d)} \\ \frac{\partial \mathbf{y}_b}{\partial \mathbf{x}_a} & \operatorname{diag}\left(\exp\left(s\left(\mathbf{x}_a\right)\right)\right) \end{bmatrix}$$

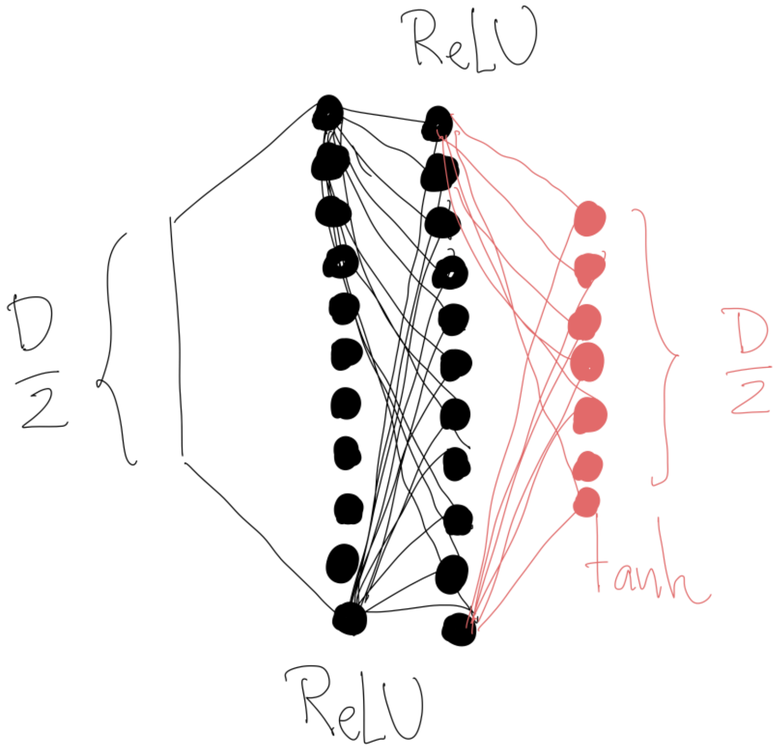
$$\det(\mathbf{J}) = \prod_{j=1}^{D-d} \exp\left(s\left(\mathbf{x}_a\right)\right)_j = \exp\left(\sum_{j=1}^{D-d} s\left(\mathbf{x}_a\right)_j\right)$$

RealNVP Likelihood

Considering k stacked layers:

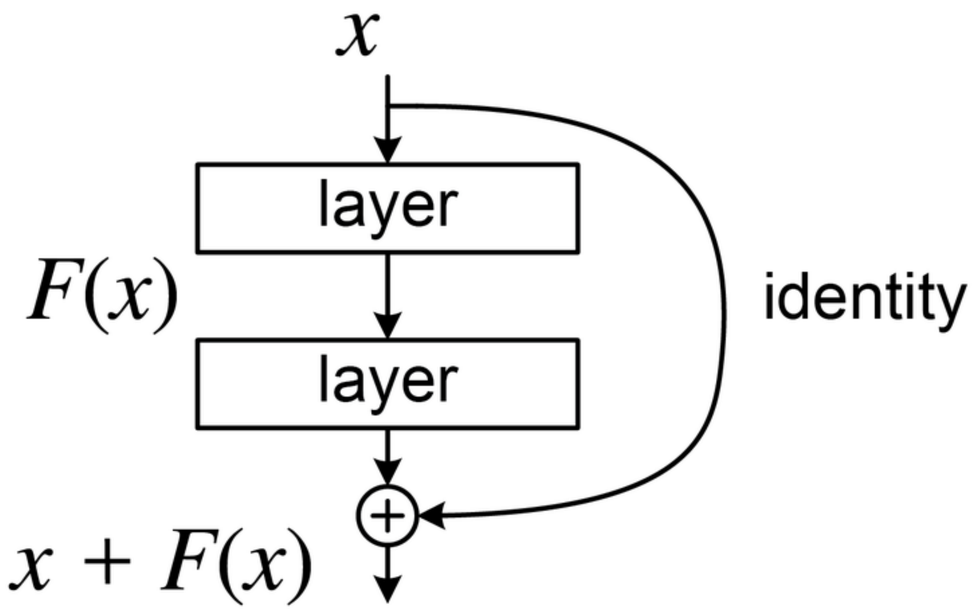
$$\log q(x\,|\,\theta) = -\frac{1}{2}\,\left\| f^{-1}(x\,|\,\theta) \right\|^{2} + \text{const} - \sum_{\ell=1}^{K}\sum_{j=1}^{D-d} S_k(x_a^k\,|\,\theta)_j$$

S and T could be, for example, feedforward NNs like:



ResNet Flows

Transformations don’t need to be explicitly invertible (https://arxiv.org/abs/1811.00995).



A Residual Block adds an identity operation that mitigates vanishing gradient issues.

ResNet Flows

Theorem: Let y(x) = x+F(x) be a residual block. Then y(x) is invertible if Lip(F(x)) <1. Where

$$\|F(x) - F(z)\| \leq Lip(F)\|x - z\|$$

Banach fixed-point theorem guarantees the existence and unicity of a fixed point for the map:

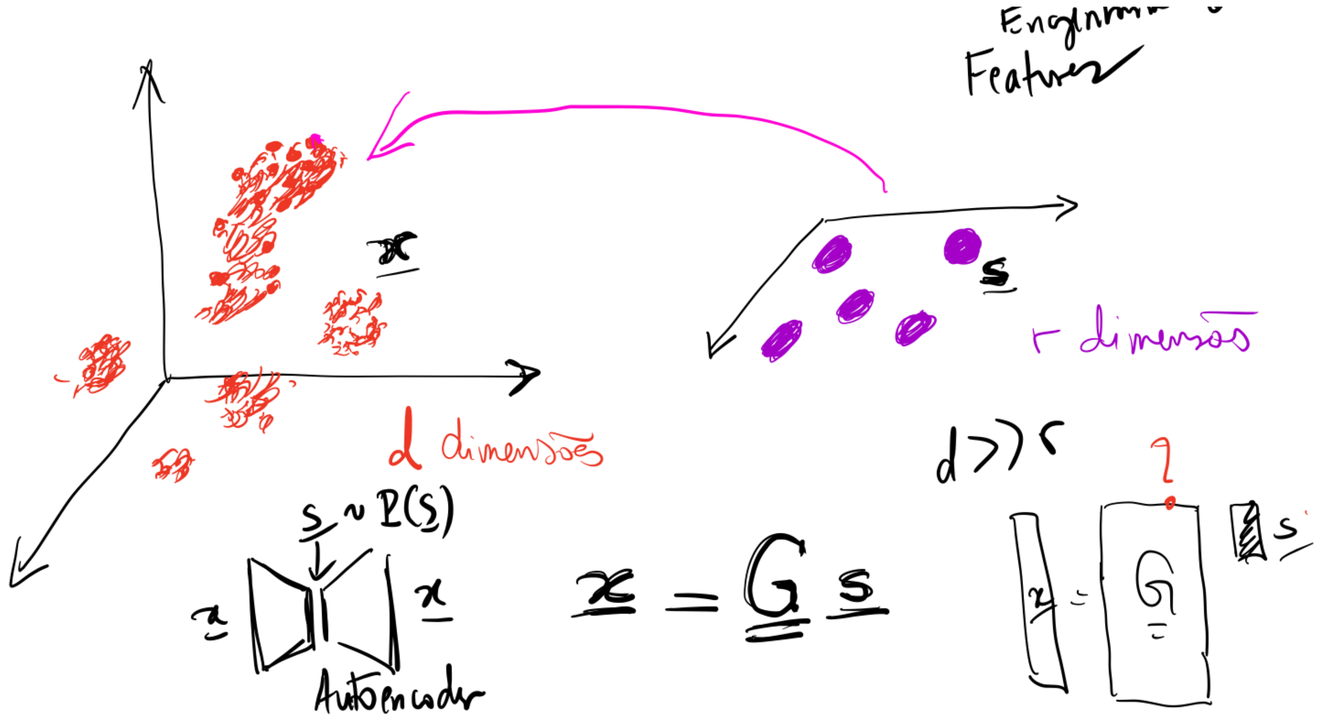
$$x^{(i+1)} = z - F(x^{(i)})$$

This fixed point yields $y^{-1}(z)$

It is possible to guarantee Lip(F(x)) <1 for many relevant architectures like ConvNN or DNNs

6. Auto-Encoders

Linear Auto-encoders



In a dimensionality reduction scenario, we would like to find a low dimensional representation for x, that can be decompressed with minimum loss.

Linear Auto-encoders: ICA

By supposing independence of the components of the latent vector $\mathbf{z}$ we devise the method known as Independent Component Analysis (ICA):

$$p(\mathbf{s}) = \prod_{\ell=1}^{m} p(s_\ell)$$

We also know that

$$\mathbf{x} = \mathbf{G}\mathbf{s}$$

Where $\mathbf{G}$ is a rectangular $d \times m$ matrix.

Linear Auto-encoders: ICA

We can use Bayesian inference:

$$\mathbf{D} = \{\underline{x}^{(n)}\}_{n=1}^{N}$$

$$P(\underline{G}|\mathbf{D},\mathcal{H}) = \frac{P(\mathbf{D}|\underline{G},\mathcal{H})\,P(\underline{G}|\mathcal{H})}{P(\mathbf{D}|\mathcal{H})} = \text{cte}$$

Bayes

$$\arg\max P(\underline{G}|\mathbf{D},\mathcal{H}) = \arg\max P(\mathbf{D}|\underline{G},\mathcal{H})$$

max posterior — max verossimilhança

$$P(\mathbf{D}|\underline{G},\mathcal{H}) = \prod_{n=1}^{N} P(\underline{x}^{(n)}|\underline{G},\mathcal{H})$$

$$\ln P(\mathbf{D}|\underline{G},\mathcal{H}) = \sum_{n=1}^{N} \ln P(\underline{x}^{(n)}|\underline{G},\mathcal{H})$$

Bayesian Inference for ICA

$$P(\underline{x}^{(n)}|\underline{\underline{G}},\mathcal{H}) = \int d^r\underline{s}^{(n)}\, P(\underline{x}^{(n)}|\underline{s}^{(n)},\underline{\underline{G}},\mathcal{H})\, P(\underline{s}^{(n)}|\mathcal{H})$$

$$= \prod_{j=1}^{r} p(s_j^{(n)})$$

$$\underline{x} = \underline{\underline{G}}\,\underline{s}$$

$$P(x,y) = P(x|y)\,p(y)$$

$$P(\underline{x}^{(n)},\underline{s}^{(n)}|\underline{\underline{G}},\mathcal{H}) = P(\underline{x}^{(n)}|\underline{s}^{(n)},\underline{\underline{G}},\mathcal{H})\, P(\underline{s}^{(n)}|\mathcal{H})$$

$$= \prod_{i=1}^{d} \delta\left(x_i^{(n)} - \sum_{j=1}^{r} G_{ij}\, s_j^{(n)}\right) \prod_{j=1}^{r} p(s_j^{(n)})$$

Training ICA

$$P(\underline{x}^{(n)} | \underline{\underline{G}}, \mathcal{H}) = \int d^r \underline{s}^{(n)} \prod_{i=1}^{d} \delta\left(x_i^{(n)} - \sum_{j=1}^{r} G_{ij} s_j^{(n)}\right) \prod_{j=1}^{r} p\left(s_j^{(n)}\right)$$

$$\int ds\, \delta(x - as) f(s) = \frac{1}{a} f\left(\frac{x}{a}\right)$$

$$= \frac{1}{|\det \underline{\underline{G}}|} \prod_{j=1}^{r} p\left(G^{-1}_{ji} x_i^{(n)}\right)$$

$$\Downarrow$$

$$\ln P(\underline{x}^{(n)} | \underline{\underline{G}}, \mathcal{H}) = -\ln |\det \underline{\underline{G}}| + \sum_{j=1}^{r} \ln p\left(G^{-1}_{ji} x_i^{(n)}\right)$$

Training ICA

Introducing the encoder matrix:

$$\underline{\underline{W}} := \underline{\underline{G}}^{-1}$$

$$\underline{x} = \underline{\underline{G}}\,\underline{s}$$

$$\underline{s} = \underline{\underline{W}}\,\underline{x}$$

Using the following identities:

$$\frac{\partial}{\partial G_{ji}} \ln \det \underline{\underline{G}} = G^{-1}_{ij} = W_{ij}$$

$$\frac{\partial}{\partial G_{ji}} G^{-1}_{lm} = - G^{-1}_{lj} G^{-1}_{im} = - W_{lj} W_{im}$$

$$\frac{\partial}{\partial W_{ij}} f = - G_{jm} \left(\frac{\partial}{\partial G_{lm}} f \right) G_{li}$$

Training ICA

Defining $$z_i = \varphi_i(s_i) = \frac{d \ln p_i(s_i)}{d s_i}$$

Ex:

$$p_i(s_i) = \frac{e^{-\frac{K}{2}s_i^2}}{\sqrt{\frac{2\pi}{K}}} \Rightarrow \varphi_i(s_i) = K s_i$$

$$p_i(s_i) = \frac{A}{e^{s_i} + e^{-s_i}} \Rightarrow \varphi_i(s_i) = -\tanh(s_i)$$

Training ICA

Calculate the gradient:

$$\frac{\partial}{\partial G_{ji}} \ln P\left(\underline{x}^{(n)} \middle| \underline{G}, \mathcal{H}\right) = -W_{ij} - a_i z_e W_{ej}$$

ou

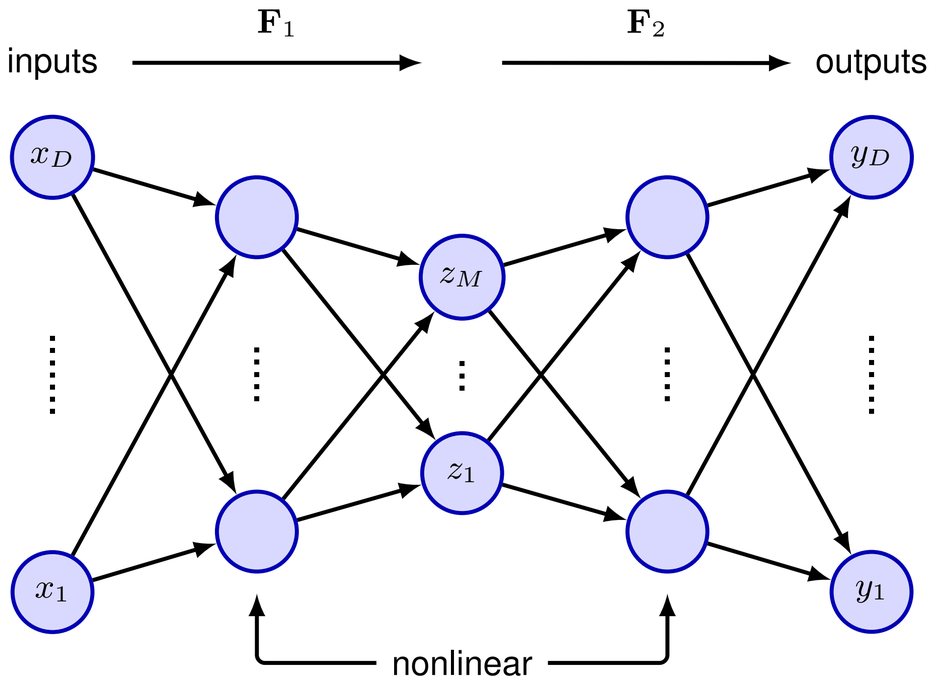
$$\frac{\partial}{\partial W_{ij}} \ln P\left(\underline{x}^{(n)} \middle| \underline{G}, \mathcal{H}\right) = G_{ji} + x_j z_i$$

We finally find:

$$\Delta W_{ij} = W_{ji}^{-1} + x_j z_i$$

$$z_i = \varphi_i\left(\sum_l W_{il} x_l\right)$$

Variational Auto-encoders



Training a generic Auto-encoder involves maximising a intractable likelihood:

$$p(x \mid w) = \int p(x \mid z, w) p(z) dz$$

Evidence Lower Bound (ELBO)

It is easy to verify that for any choice of q(z) we have:

$$\ln p(x \mid w) = \mathscr{L}(w) + \mathrm{KL}\left(q(z) \| p(z \mid x, w)\right)$$

with:

$$\mathscr{L}(w) = \int q(z) \ln \frac{p(x, z \mid w)}{q(z)} dz$$

$$\mathrm{KL}(q(z) \| p(z \mid x, w)) = -\int q(z) \ln \frac{p(z \mid x, w)}{q(z)} dz$$

Evidence Lower Bound (ELBO)

As $\mathrm{KL}(q(z)\|p(z\,|\,x,w)) \geq 0$

We have that $\ln p(x\,|\,w) \geq \mathscr{L}(w)$

ELBO consists in choosing q(z) wisely to make the optimisation of $\mathscr{L}(w)$ feasible. We choose

$$q(z\,|\,x,\phi) = \prod_{j=1}^{M} \mathcal{N}\left(z_j\,|\,\mu_j(x,\phi), \sigma_j^2(x,\phi)\right)$$

Neural nets

Evidence Lower Bound (ELBO)

We can rewrite the ELBO as

$$\mathscr{L}_n(w,\phi) = \int q(z_n \,|\, x_n, \phi) \ln \left\{ \frac{p(x_n, z_n, w)}{q(z_n \,|\, x_n, \phi)} \right\} dz_n$$

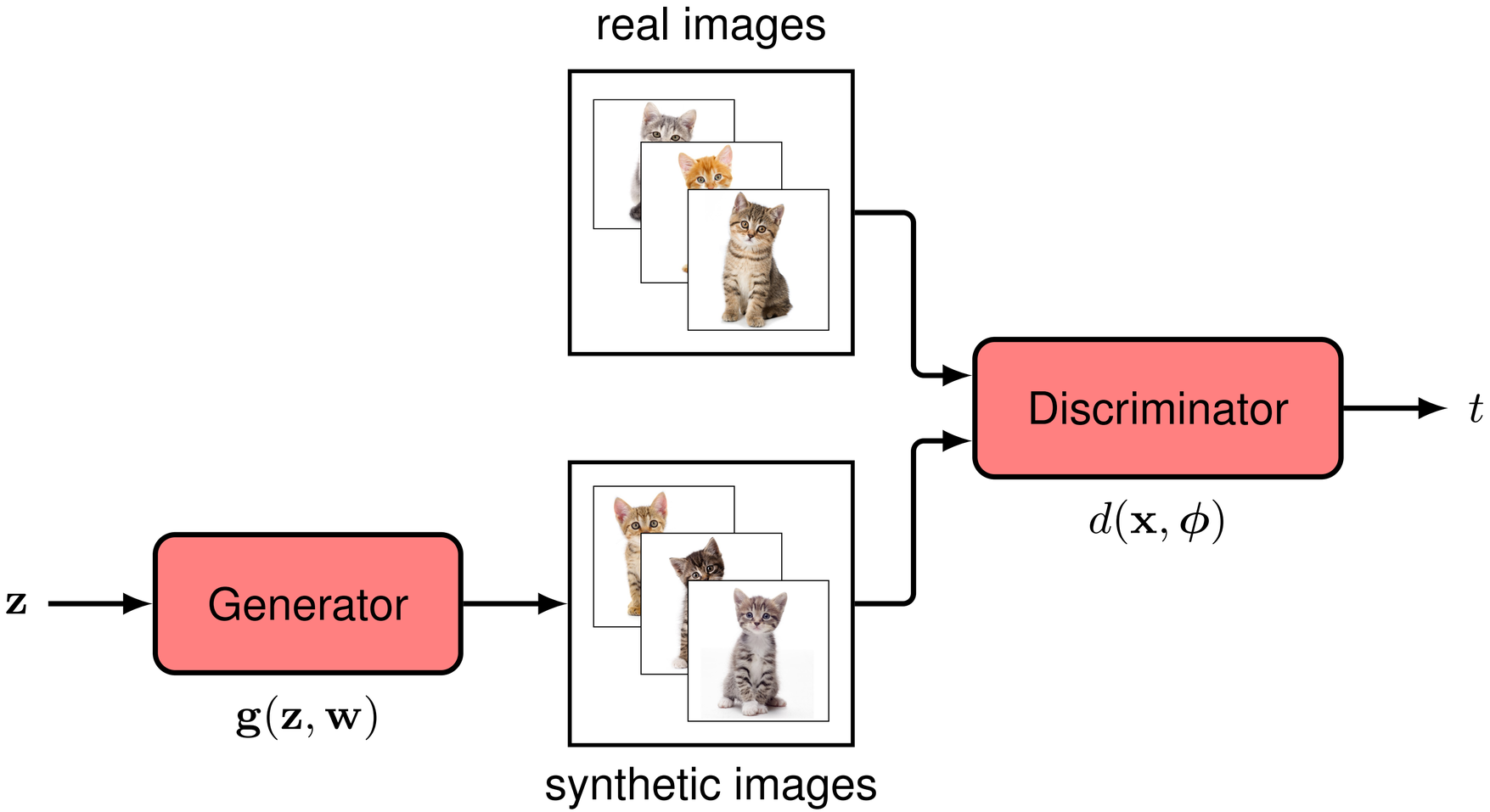
$$= \int q(z_n \,|\, x_n, \phi) \ln p(x_n \,|\, z_n, w) dz_n - \mathrm{KL}(q(z_n \,|\, x_n, \phi) \| p(z_n))$$

$$\frac{1}{L} \sum_{l=1}^{L} \ln p(x_n \,|\, z_n^{(l)}, w)$$

$$\frac{1}{2} \sum_{j=1}^{M} \left\{ 1 + \ln(\sigma_j^2(x_n, \phi)) - \mu_j^2(x_n, \phi) - \sigma_j^2(x_n, \phi) \right\}$$

7. Generative Adversarial Networks

Idea



Train a generator of data and a discriminator competitively.

Training

Real data is labeled t=1, while synthetic data is labeled t=0. The output of the discriminator is $P(t=1) = d(\mathbf{x}, \phi)$. The error function is the standard cross-entropy:

$$E(\mathbf{w}, \phi) = -\frac{1}{N}\sum_{n=1}^{N}\left[t_n \ln d_n + (1-t_n)\ln(1-d_n)\right]$$

Considering the labels for the GAN we have:

$$E_{\text{GAN}}(\mathbf{w}, \phi) = -\frac{1}{N_{\text{real}}}\sum_{n\in\text{real}} \ln d(x_n, \phi) - \frac{1}{N_{\text{synth}}}\sum_{n\in\text{synth}} \ln\left(1 - d(g(z_n, \mathbf{w}), \phi)\right)$$

Training

We have to minimize the loss in respect to ϕ, while we maximize it in respect to $\mathbf{w}$

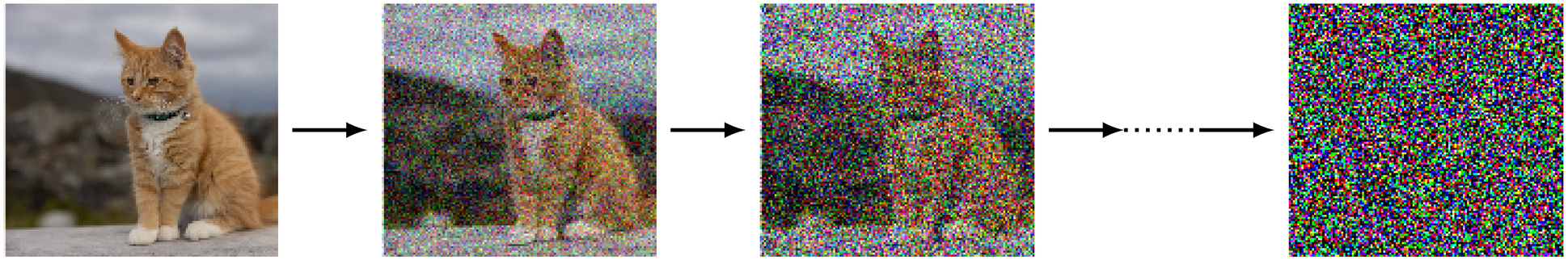
$$\Delta\phi = -\lambda \nabla_{\phi} E_n(\mathbf{w}, \phi)$$

$$\Delta\mathbf{w} = \lambda \nabla_{\mathbf{w}} E_n(\mathbf{w}, \phi)$$

where we suppose gradients over single data loss function or over minibatches of data

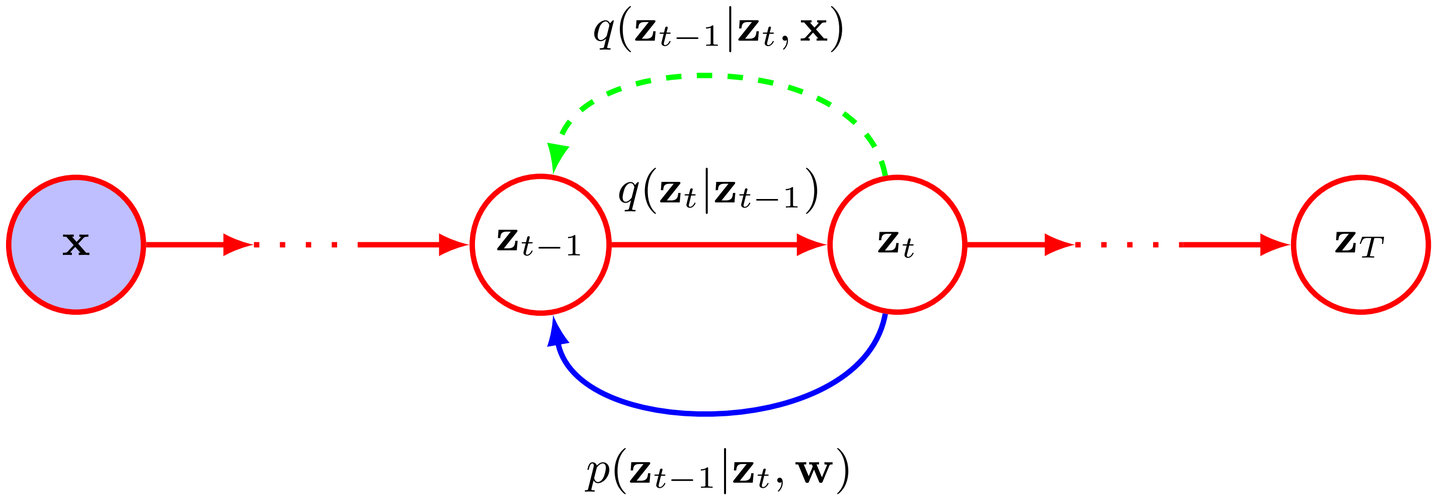
8. Diffusion Models

General Idea



A diffusion model learns to generate data by iteratively denoising samples, reversing a process that progressively added noise to the data during training.

Forward Encoder



Noise is incrementally added to the data x in the first step we have $q(z_1|x) = N(z_1|\sqrt{1-\beta_1}x, \beta_1\mathbb{I})$ with $\beta_1 < 1$

In the additional steps we have: $q(z_\ell|z_{\ell-1}) = N(z_\ell|\sqrt{1-\beta_\ell}z_{\ell-1}, \beta_\ell\mathbb{I})$

Diffusion Kernel

The whole trajectory of latent variables

$$q(z_1, \ldots, z_t \,|\, \mathbf{x}) = q(z_1 \,|\, \mathbf{x}) \prod_{\tau=2}^{t} q(z_\tau \,|\, z_{\tau-1})$$

Marginalising we get the diffusion kernel as

$$q(z_t \,|\, \mathbf{x}) = \mathcal{N}(z_t \,|\, \sqrt{\alpha_t}\mathbf{x}, (1-\alpha_t)\mathbf{I})$$

with

$$\alpha_t = \prod_{\tau=1}^{t} (1-\beta_\tau)$$

for $T \to \infty$

$$q(z_T \,|\, \mathbf{x}) = \mathcal{N}(z_T \,|\, \mathbf{0}, \mathbf{I})$$

Reverse Decoder

The reverse transformation can in principle be computed by considering Bayes' Theorem

$$q(z_{t-1} \mid z_t) = \frac{q(z_t \mid z_{t-1}) q(z_{t-1})}{q(z_t)}$$

However the part

$$q(z_{t-1}) = \int q(z_{t-1} \mid \mathbf{x}) p(\mathbf{x})\, d\mathbf{x}$$

Is intractable. The best we can do is trying to learn an approximation for $q(z_{t-1} \mid z_t)$

Reverse Decoder

We approximate the reverse transformation by the following

$$p(z_{t-1} \mid z_t, w) = \mathcal{N}(z_{t-1} \mid \mu(z_t, w, t), \beta_t \mathbf{I})$$

↑

Deep Neural Network

The denoising process becomes:

$$p(x, z_1, \ldots, z_T \mid w) = p(z_T) \left[\prod_{t=2}^{T} p(z_{t-1} \mid z_t, w) \right] p(x \mid z_1, w)$$

Training

The likelihood demands integrating over all possible latent trajectories what is obviously intractable

$$p(x \mid w) = \int \cdots \int p(x, z_1, \ldots, z_T \mid w)\, dz_1 \ \ldots \ dz_T$$

For any choice of q(z) we have:

$$\ln p(x \mid w) = \mathscr{L}(w) + \mathrm{KL}\Big(q(z) \| p(z \mid x, w)\Big)$$

with:

$$\mathscr{L}(w) = \int q(z) \ln \frac{p(x, z \mid w)}{q(z)} dz$$

Training

But $\mathrm{KL}\big(q(z)\|p(z\,|\,x,w)\big) \geq 0$

Leading to $\ln p(x\,|\,w) \geq \mathscr{L}(w)$

We can then maximize $\mathscr{L}(w)$ instead of the intractable likelihood, we just have to choose q(z) wisely. We choose

$$q(z_1,\ldots,z_t\,|\,\mathbf{x}) = q(z_1\,|\,\mathbf{x})\prod_{\tau=2}^{t} q(z_\tau\,|\,z_{\tau-1})$$

Training

With this choice we get:

$$\mathscr{L}(w) = \mathbb{E}_q \left[\log \frac{p(z_T) \left\{ \prod_{t=2}^{T} p(z_{t-1} \,|\, z_t, w) \right\} p(x \,|\, z_1, w)}{q(z_1 \,|\, x) \prod_{t=2}^{T} q(z_t \,|\, z_{t-1}, x)} \right]$$

$$= \mathbb{E}_q \left[\log p(z_T) + \sum_{t=2}^{T} \log p(z_{t-1} \,|\, z_t, w) + \log p(x \,|\, z_1, w) - \log q(z_1 \,|\, x) - \sum_{t=2}^{T} \log q(z_t \,|\, z_{t-1}, x) \right]$$

Where

$$\mathbb{E}_q[\,\cdot\,] = \int \cdots \int q(z_1 \,|\, x) \prod_{t=2}^{T} q(z_t \,|\, z_{t-1}) [\,\cdot\,] \, dz_1 \ldots dz_T$$

is an expectation over the forward process that is easy to generate

Training

Keeping only terms that depend on w and using Bayes theorem yields:

$$\mathscr{L}(w) = \mathbb{E}_q \left[\sum_{t=2}^{T} \log \left(\frac{p(z_{t-1} \mid z_t, w)}{q(z_{t-1} \mid z_t, x)} \right) + \log p(x \mid z_1, w) \right]$$

Finally we get

$$\mathscr{L}(w) = \int q(z_1 \mid x) \ln p(x \mid z_1, w) dz_1 - \sum_{t=2}^{T} \int \mathrm{KL}(q(z_{t-1} \mid z_t, x) \| p(z_{t-1} \mid z_t, w)) q(z_t \mid x) dz_t$$

$$\mathbb{E}_q[\ln p(x \mid z_1, w)] \simeq \frac{1}{L} \sum_{l=1}^{L} \ln p(x \mid z_1^{(l)}, w)$$

$$\mathrm{KL}(q(z_{t-1} \mid z_t, x) \| p(z_{t-1} \mid z_t, w)) = \frac{1}{2\beta_t} \| m_t(x, z_t) - \mu(z_t, w, t) \|^2 + \text{const}$$

9. Transformers

Language Model

Consider an alphabet Σ and a end-of-seq symbol EOS. A Language Model is a collection of conditional probability distributions $p(y \mid \mathbf{y})$, where $\mathbf{y} \in \Sigma^*$, the Kleene closure of Σ.

$$p(y_1, \cdots, y_T) = p(EOS \mid \mathbf{y}) \prod_{t=1}^{T} p(y_t \mid \mathbf{y}_{<t})$$

Embedding Function

Converts a symbol of the alphabet in a vector of $\mathbb{R}^D$: $e(y) : \Sigma \rightarrow \mathbb{R}^D$

$$\mathbf{M} \begin{bmatrix} \delta(y_1, y) \\ \\ \\ \\ \\ \\ \delta(y_{|\Sigma|}, y) \end{bmatrix} = \begin{bmatrix} e(y) \end{bmatrix}$$

one-hot encoding of symbol y

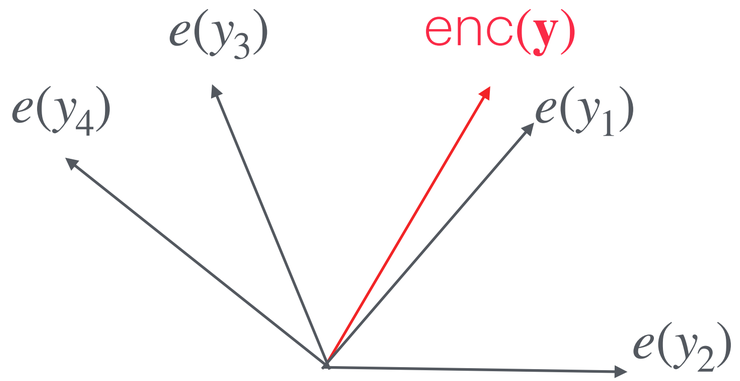
$$e_{pos}(y_e ll) = e(y_\ell) + \phi_{pos}(\ell)$$

positional encoding

Encoding Function

Converts a sequence of symbols in a vector of $\mathbb{R}^D$: $\text{enc}(\mathbf{y}) : \Sigma^* \rightarrow \mathbb{R}^D$

The internal product inside the vector space containing embeddings and encodings register meaning:



The most probable next symbol should be y_1 !

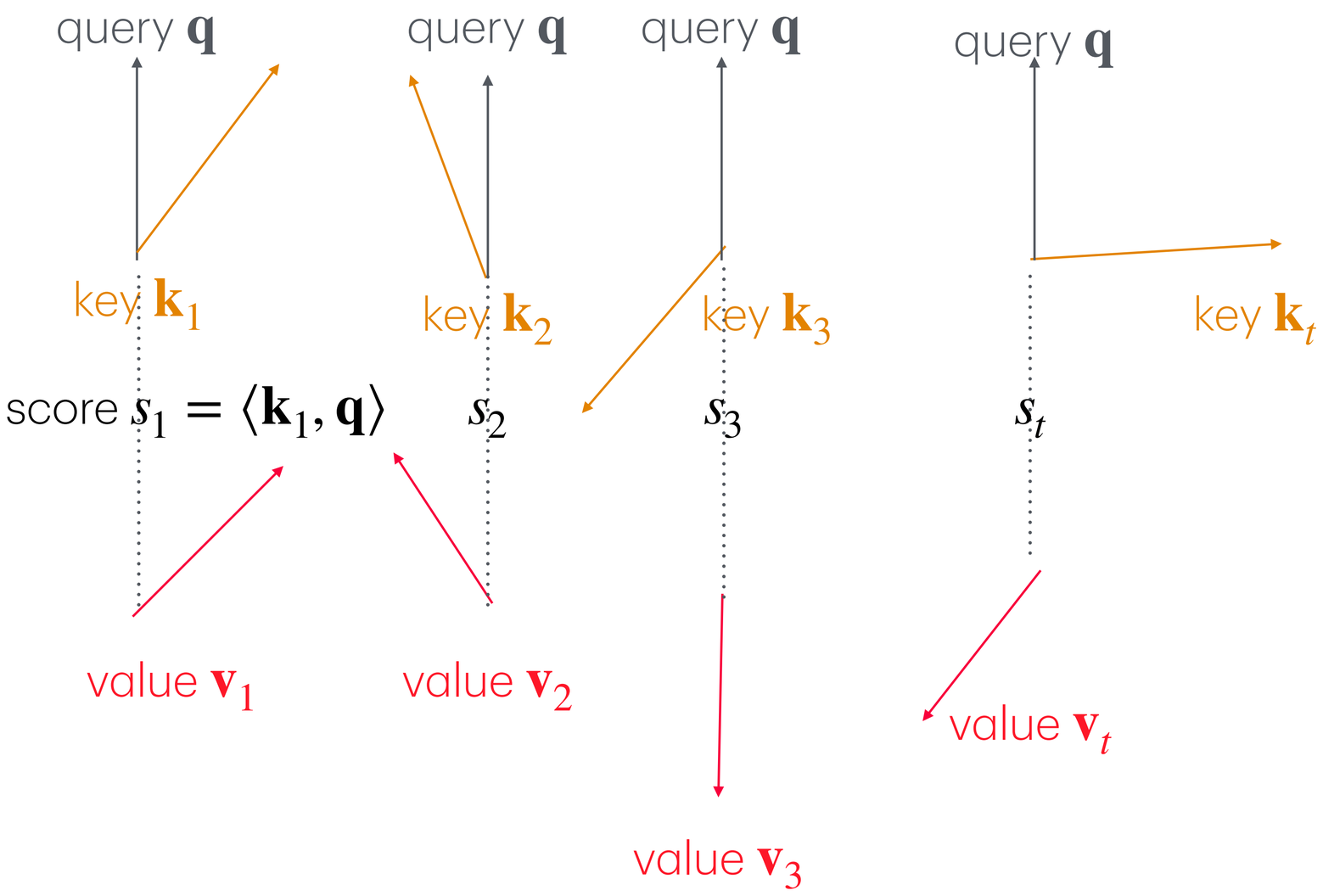
Modeling Conditionals

Converts a sequence of symbols in a vector of $\mathbb{R}^D$: $\text{enc}(\mathbf{y}) : \Sigma^* \to \mathbb{R}^D$

$$p(y \mid \mathbf{y}_{<t}) = \text{softmax}_\beta \left(\langle e(y), \text{enc}(\mathbf{y}_{<t}) \rangle \right)$$

$$\text{softmax}_\beta \left(\langle e(y), \text{enc}(\mathbf{y}_{<t}) \rangle \right) = \frac{\exp \left(\beta \langle e(y), \text{enc}(\mathbf{y}_{<t}) \rangle \right)}{\sum_{y' \in \Sigma} \exp \left(\beta \langle e(y'), \text{enc}(\mathbf{y}_{<t}) \rangle \right)}$$

Attention Mechanism

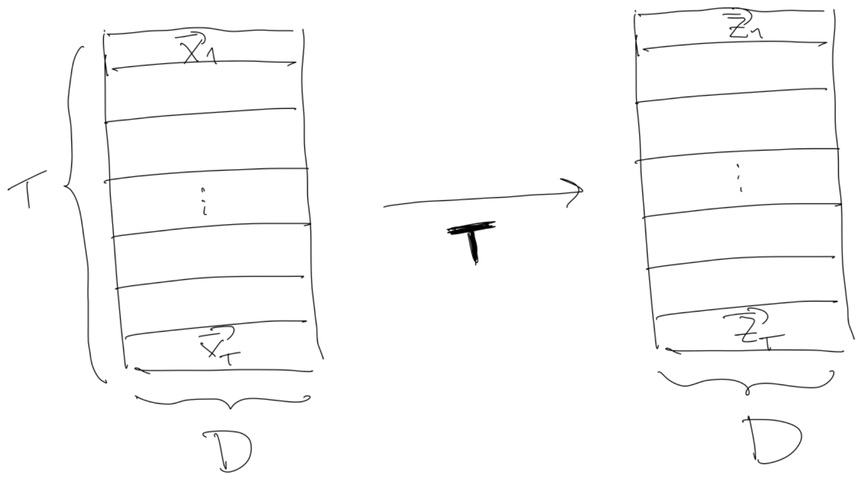


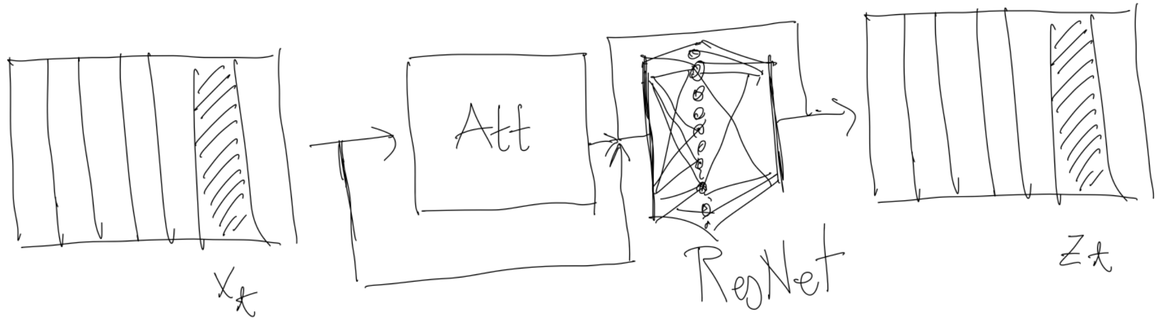
$$\mathbf{a}_t = \text{Att}(\mathbf{q}, \mathbf{K}, \mathbf{V}) = \sum_{\ell=1}^{t} s_\ell \mathbf{v}_\ell$$

Transformer Layer

$\mathbf{Q}, \mathbf{K}, \mathbf{V}, \mathbf{O}$ are parameterised functions of $\mathbb{R}^D \rightarrow \mathbb{R}^D$

$\mathbf{T} : \mathbb{R}^{T \times D} \rightarrow \mathbb{R}^{T \times D}$





$$\mathbf{a}_t = \text{Att}(\mathbf{Q}(\mathbf{x}_t), \mathbf{K}(\mathbf{x}_t), \mathbf{V}(\mathbf{x}_t)) + \mathbf{x}_t$$

$$\mathbf{z}_t = \mathbf{O}(\mathbf{a}_t) + \mathbf{a}_t$$

$$\mathbf{T}(\mathbf{X}) = \mathbf{Z}$$

Transformer Block

$$\mathbf{X}^1 = (e_{pos}(y_0), \cdots, e_{pos}(y_{t-1}))$$

$$\mathbf{Z}^\ell = \mathbf{T}_\ell(\mathbf{X}^\ell), \quad 1 \leq \ell < L$$

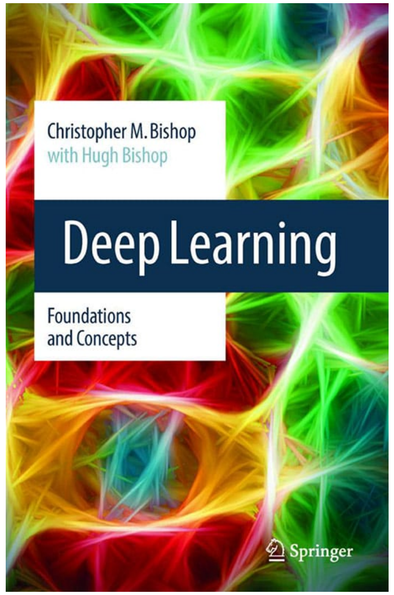
$$\mathbf{X}^{\ell+1} = \mathbf{Z}^\ell, \quad 1 \leq \ell < L$$

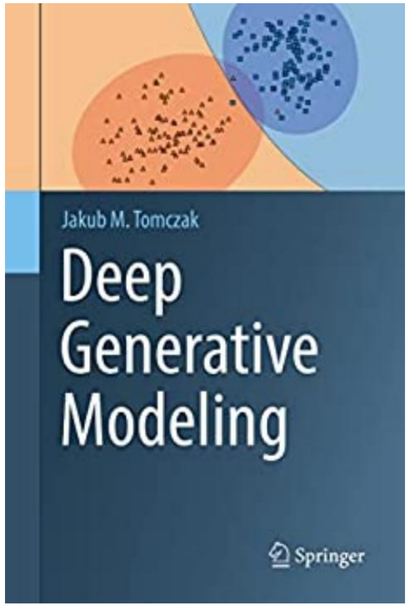
$$\text{enc}(\mathbf{y}_{<t}) = F(\mathbf{z}_t^L)$$

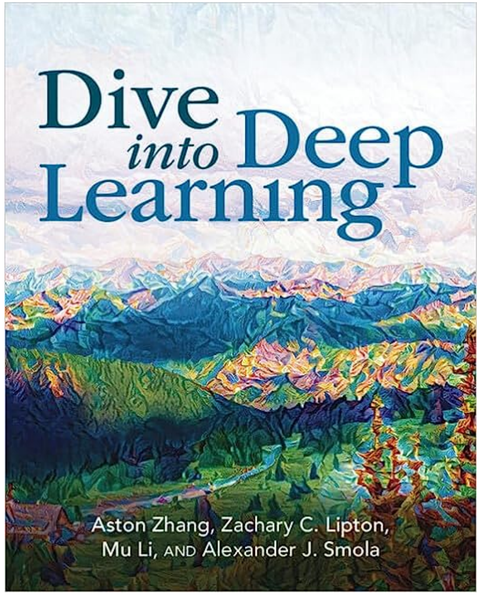
$\mathbf{F}$ is a Neural Network

10. References

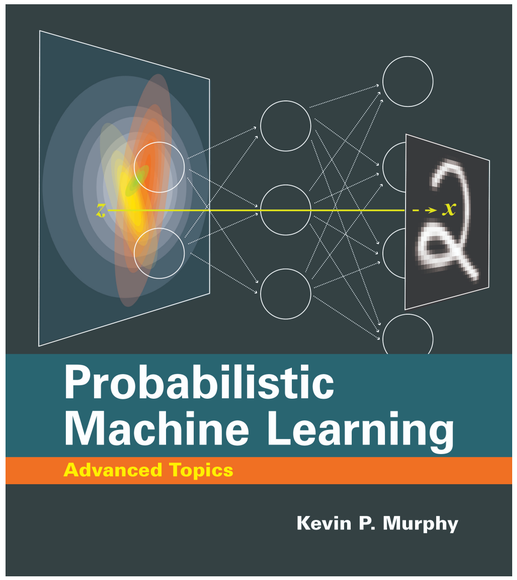
References







https://d2l.ai/



https://probml.github.io/pml-book/book2.html