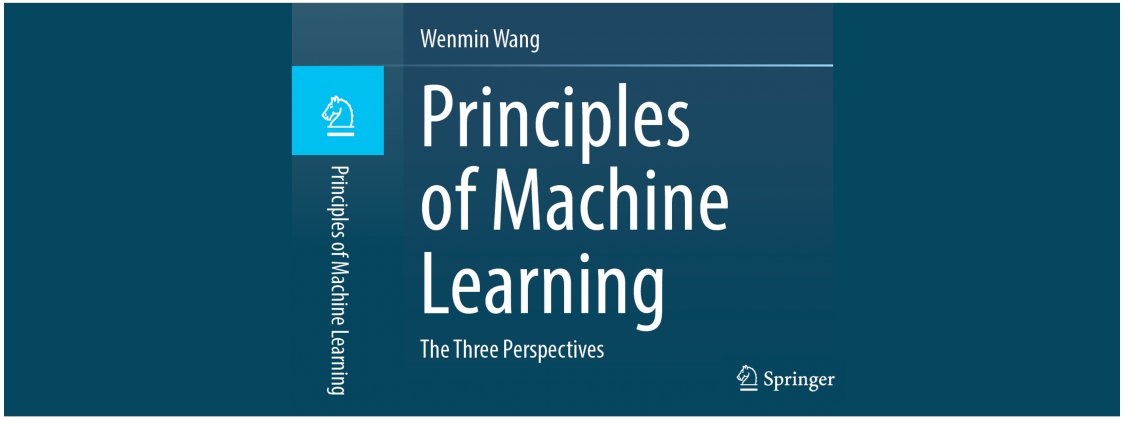

Wenmin Wang
Principles of Machine Learning
The Three Perspectives
Principles of Machine Learning
Springer

Part III Paradigms

9 Unsupervised Learning Paradigm

Contents

What is Unsupervised Learning

Unsupervised learning is second one of three major paradigms of machine learning. It is fundamentally different from the supervised learning paradigm.

Definition: Unsupervised learning

Unsupervised learning (UL) is a learning paradigm to machine learning with unlabeled data, which analyzes or processes the data based on a certain established criterion to obtain desired results.

Several unsupervised tasks are existed, but since their relatively independent natures, it is hard to abstract common principles, as in supervised learning.

However, human learning is mainly unsupervised, therefore unsupervised machine learning becomes even more important.

What is Unsupervised Learning

It is a way of “self-learning” algorithm based on a certain given criterion, without a specific “teacher”.

Supervised learning vs. unsupervised learning

Supervised Learning	Unsupervised Learning
The training data for supervised learning algorithm is labeled.	All data for unsupervised learning algorithm is unlabeled.
The algorithm is trained based on labeled data.	Analysis or processing is carried out based on unlabeled data.

Formal Description

Unsupervised learning (UL) can be represented as UL $= \langle \mathcal{X},\ \mathcal{Y}, H, \mathcal{L} \rangle$, a 4-tuple. Where: $\mathcal{X}$: input space, $\mathcal{Y}$: output space, H: hypothesis set, $\mathcal{L}$: loss function.

Using $P(\boldsymbol{x})$ from $\mathcal{X}$ to obtain n independent and identically distributed (i.i.d.) data:

$$D = \{\boldsymbol{x}_i \mid \boldsymbol{x}_i \in \mathcal{X} \text{ and } i = 1, \ldots, n\}.$$

Let target function of unsupervised learning be:

$$f : \mathcal{X} \to \mathcal{Y}$$

Hypothesis set H is the algorithm designed for a task of unsupervised learning:

$$H : \mathcal{X} \to \mathcal{Y}\ .$$

Formal Description

By directly analyzing D, find the best hypothesis $h \in H$, $h(\boldsymbol{x}) = \hat{y}$, so that the loss between the hypothesis $h(\boldsymbol{x})$ and the target function $f(\boldsymbol{x})$ is minimized:

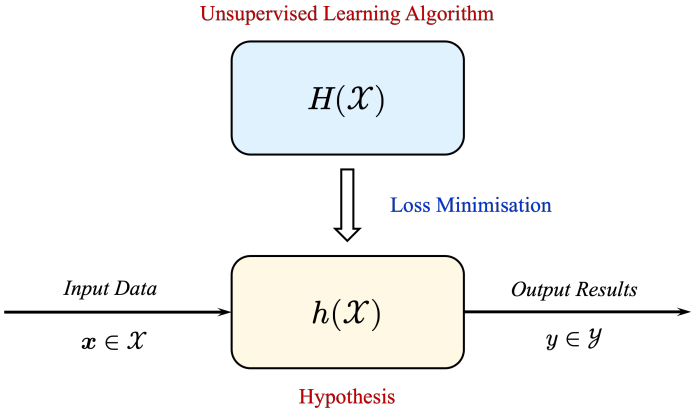
$$\mathcal{L}(h(\boldsymbol{x}), f(\boldsymbol{x})) = \mathcal{L}(\hat{y}, y) = \arg\min_{h \in H} \mathbb{E}[h(\boldsymbol{x}) - f(\boldsymbol{x})].$$

The loss function depends on specific task and its model, because it varies with the different tasks and models of unsupervised learning.

Here, $\mathcal{L}(\cdot, \cdot)$ is an abstract representation of the loss function, the purpose is to minimize the error between hypothesis and target function.

In other words, using the loss function to make the hypothesis as close as possible to its target function.

Illustrated Description



A schematic diagram of unsupervised learning,
it can be seen that it does not have a training process with labeled data.

Classic Tasks

The classic tasks of unsupervised learning mainly include following types.

Clustering:

Dividing data into several clusters based on the analysis of input data. It can be divided into classical and neo-classical clustering methods.

Dimensionality reduction:

Mapping high-dimensional data into low-dimensional space. It can be divided into linear, non-linear, and deep dimensionality reduction.

Association:

Discovering associative relationships between data objects in large databases and establishing corresponding association rules. The *shopping basket analysis* in supermarket is known as one of the examples.

Classic Tasks

Differences of the three tasks are reflected in their input and output spaces.

Clustering:

Output is k clusters $\{G_j \mid j = 1, \ldots, k\} \subseteq \mathcal{Y}$, obtained by grouping input data. Where G_j is also a subset of input data, $G_j \subseteq \mathcal{X}$; and k is unknown in advance.

Dimensionality Reduction:

Output is new data after reducing the dimensionality of input data, $h : \mathcal{X} \to \mathcal{Y}$. Where, $\mathcal{X} \subseteq \mathbb{R}^h$ is high-dimensional input space, $\mathcal{Y} \subseteq \mathbb{R}^l$ is low-dimensional output space, $l \ll h$ and $1 \leq l \leq 3$.

Association:

Output is a set of association rules. Let $\mathcal{X} = \{x_1, x_2, \ldots, x_n\}$ be a set of items, if there are a subset of items $X \subseteq \mathcal{X}$ and a single item $Y \in \mathcal{X}$, then $X \Rightarrow Y \in \mathcal{Y}$.

Why Beyond

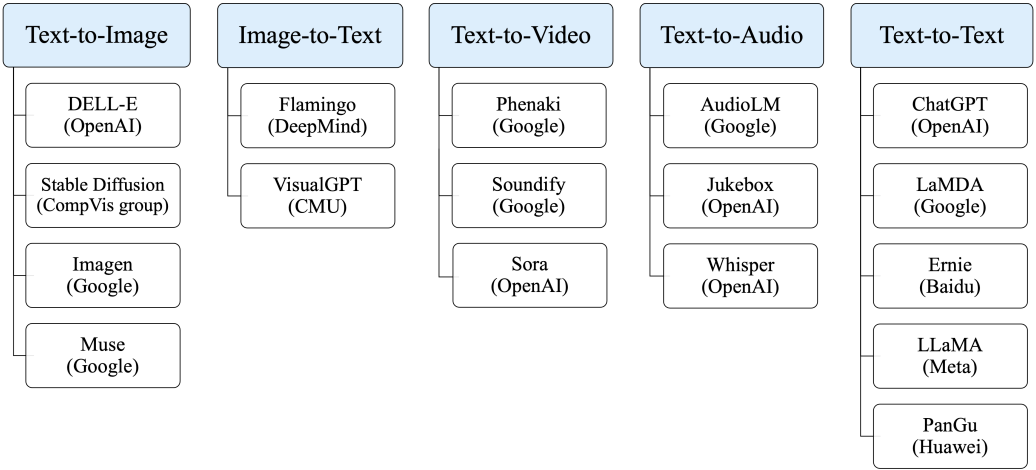
The paradigm of unsupervised learning should not be limited to the three classic tasks.

Because humans have a wide range of unsupervised learning abilities, and machines should also have relatively broad unsupervised learning functions.

With the continuous development of deep learning, unsupervised learning is increasingly attracting people's attention.

Some researchers combine deep neural networks with some other approaches in AI, ML, probability theory, statistics, physics, etc., and propose some emerging unsupervised models, namely *generative models*.

Representative Generative Models



About the Energy Models

Energy models, are also referred to as energy-based models (EBMs).

Be a type of generative model that introduces statistical mechanics into machine learning.

Two important properties in axioms of probability theory:

- Non-negativity: $p_\theta(\boldsymbol{x}) \geq 0$;
- Normalization: $\int p_\theta(\boldsymbol{x}) d\boldsymbol{x} = 1$.

Energy models can avoid the normalization problem, making the design of machine learning algorithms more flexible.

Energy models can provide a unified framework for many probabilistic and non-probabilistic learning.

Basic Energy Model

It refers to non-normalized probability scalar as energy, and characterizes its dependence by association of energy with input and latent variables.

Inference in the model includes finding the latent variables (values) that minimize energy given a set of input variables (values).

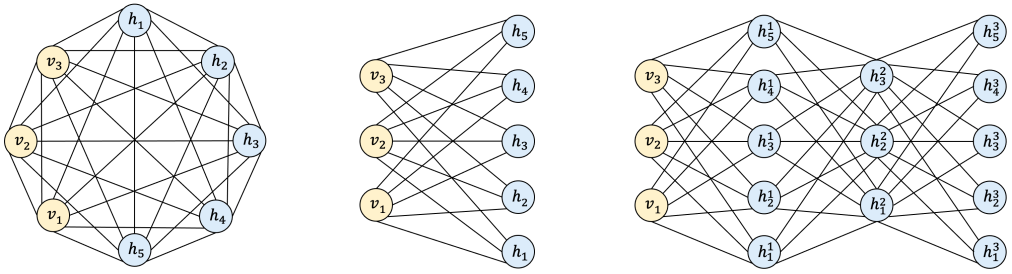
It learns a function that associates low energy with the correct values of latent variables and high energy with incorrect values.

Without loss of generality, here we only introduce the single-variable energy model:

$$p_\theta(\boldsymbol{x}) = \frac{e^{-E_\theta(\boldsymbol{x})}}{Z_\theta(\boldsymbol{x})}, \qquad Z_\theta(\boldsymbol{x}) = \int e^{-E_\theta(\boldsymbol{x})} d\boldsymbol{x}.$$

Where $E_\theta(\boldsymbol{x}) : \mathbb{R}^D \to \mathbb{R}$ is an energy function, and $Z_\theta(\boldsymbol{x})$ is normalizing constant.

Boltzmann Models



The three types of Energy-based Boltzmann models.
(a) Boltzmann machine (BM), (b) restricted Boltzmann machine (RBM), and (c) deep Boltzmann machine (DBM).

Boltzmann Models

Boltzmann Machine:

A Boltzmann machine (BM) is a undirected network of symmetrically connected.

It includes a set of visible units $\boldsymbol{v} \in \{0,1\}^D$, and a set of hidden units $\boldsymbol{h} \in \{0,1\}^P$, where D and P denote the numbers of visible and hidden units, respectively.

The energy of the state $\{\boldsymbol{v}, \boldsymbol{h}\}$ is defined as follows:

$$E_\theta(\boldsymbol{v}, \boldsymbol{h}) = -\frac{1}{2}\boldsymbol{v}^T \mathbf{L}\boldsymbol{v} - \frac{1}{2}\boldsymbol{h}^T \mathbf{J}\boldsymbol{h} - \boldsymbol{v}^T \mathbf{W}\boldsymbol{h}.$$

Where $\theta = \{\mathbf{W}, \mathbf{L}, \mathbf{J}\}$ is the parameter, and $\{\mathbf{W}, \mathbf{L}, \mathbf{J}\}$ are three symmetric interaction terms: $\mathbf{W}$ is visible-to-hidden, $\mathbf{L}$ is visible-to-visible, and $\mathbf{J}$ is hidden-to-hidden.

It is trained by contrastive divergence and finds its equilibrium state through Gibbs sampling. But, this requires an exponential time in terms of the number of units.

Boltzmann Models

Restricted Boltzmann Machine:

A restricted Boltzmann machine (RBM) is a special type of Boltzmann machine.

The restriction refers to limiting the connections between visible and hidden units to solve the exponential time problem in BM.

RBM becomes a two-layer undirected network: one is visible, and the other is hidden.

Its state $\{\boldsymbol{v}, \boldsymbol{h}\}$ is defined by the following energy:

$$E_\theta(\boldsymbol{v}, \boldsymbol{h}) = -\boldsymbol{v}^\mathsf{T}\mathbf{W}\boldsymbol{h}.$$

Where $\theta = \{\mathbf{W}\}$, i.e., there is only visible to hidden, and no other two items in BM.

RBM can be trained using the maximum likelihood method. The sampling can be done by Gibbs sampling or Markov chain Monte Carlo (MCMC) methods.

Boltzmann Models

Deep Boltzmann Machine:

A deep Boltzmann machine (DBM) is an undirected deep network with one visible layer and multiple hidden layers.

The DBM is equivalent to adding several hidden layers to the RBM.

For the DBM that has one visible layer and three hidden layers, its state $\left\{\boldsymbol{v}, \boldsymbol{h}^1, \boldsymbol{h}^2, \boldsymbol{h}^3\right\}$ is defined by the following energy:

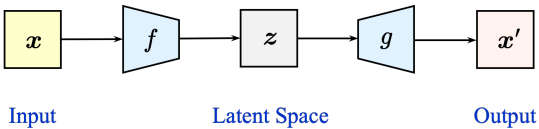
$$E_\theta\left(\boldsymbol{v}, \boldsymbol{h}^1, \boldsymbol{h}^2, \boldsymbol{h}^3\right) = -\boldsymbol{v}^\intercal \mathbf{W}^1 \boldsymbol{h}^1 - \boldsymbol{h}^{1\intercal} \mathbf{W}^2 \boldsymbol{h}^2 - \boldsymbol{h}^{2\intercal} \mathbf{W}^3 \boldsymbol{h}^3.$$

Where the parameter $\theta = \left\{\mathbf{W}^1, \mathbf{W}^2, \mathbf{W}^3\right\}$ denotes the three symmetric interaction terms from visible to hidden, hidden to hidden, and hidden to hidden, respectively.

Autoencoders

Autoencoders:

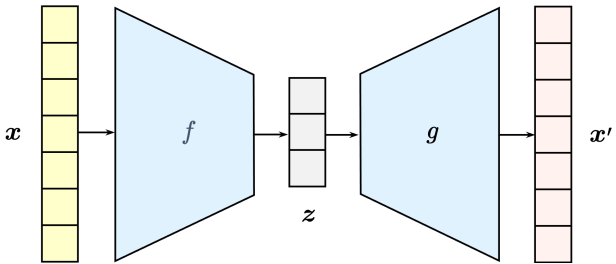
Let $\mathcal{X} \subseteq \mathbb{R}^u$ be a u-dimensional space, $\mathcal{Z} \subseteq \mathbb{R}^v$ be a v-dimensional latent space, $\boldsymbol{x}, \boldsymbol{x}' \in \mathcal{X}$ are input and output data respectively, and $\boldsymbol{z} \in \mathcal{Z}$ is intermediate code in the latent space. Let f and g denote encoder and decoder. The f encodes $\boldsymbol{x}$ to $\boldsymbol{z}$, i.e., $\boldsymbol{z} = f(\boldsymbol{x})$; and the g decodes $\boldsymbol{z}$ into $\boldsymbol{x}'$, i.e, $\boldsymbol{x}' = g(\boldsymbol{z})$. Therefore, a neural network is known as an autoencoder (AE), if it satisfies $\boldsymbol{x}' = g(\boldsymbol{z}) = g(f(\boldsymbol{x}))$ and $\boldsymbol{x}' \not\equiv \boldsymbol{x}$.



Undercomplete Autoencoders

Undercomplete Autoencoders:

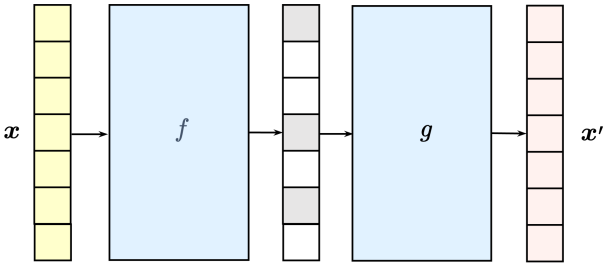
It is called an undercomplete autoencoder (UAE), iff the dimension of its latent space $\mathcal{Z} \subseteq \mathbb{R}^v$ is less than the dimension of the data space $\mathcal{X} \subseteq \mathbb{R}^u$, i.e., $v < u$. Its loss function can be expressed as $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{L}(\boldsymbol{x}, g(f(\boldsymbol{x})))$.



Sparse Autoencoders

Sparse Autoencoders:

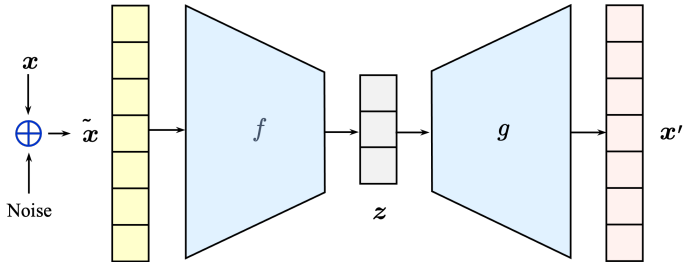
A sparse autoencoder (SAE) is an AE with a sparsity constraint, but the dimension of latent space can be equal to or even greater than input space. Its loss function is $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') + \Omega(\boldsymbol{z}) = \mathcal{L}(\boldsymbol{x}, g(f(\boldsymbol{x}))) + \Omega(\boldsymbol{z})$, where $\Omega(\boldsymbol{z})$ is a sparsity penalty term, can be adopted by KL divergence, L^1 or L^2 regularization.



Denoising Autoencoders

Denoising Autoencoders:

A denoising autoencoder (DAE) learns useful features in the input data by changing the reconstruction error term in the loss function: $\mathcal{L}(\boldsymbol{x}, \boldsymbol{x}') = \mathcal{L}(\boldsymbol{x}, g(f(\widetilde{\boldsymbol{x}})))$, where $\widetilde{\boldsymbol{x}} = \boldsymbol{x} \oplus$ noise.



Variational Autoencoders

Variational Autoencoders:

A variational autoencoder (VAE) is an ANN that uses probabilistic graphical models and variational Bayes method.

Generally, we need a probabilistic encoder and a probabilistic decoder, i.e.,

$$z \sim p_\theta(z|\boldsymbol{x}) \quad \text{and} \quad \boldsymbol{x}' \sim p_\theta(\boldsymbol{x}|z).$$

According to Bayes' theorem, the encoder should be solved by:

$$p_\theta(z|\boldsymbol{x}) = \frac{p_\theta(\boldsymbol{x}|z)\, p_\theta(z)}{p_\theta(\boldsymbol{x})}, \qquad p_\theta(\boldsymbol{x}) = \int_z p_\theta(\boldsymbol{x}, z) dz = \int_z p_\theta(\boldsymbol{x}|z)\, p_\theta(z) dz.$$

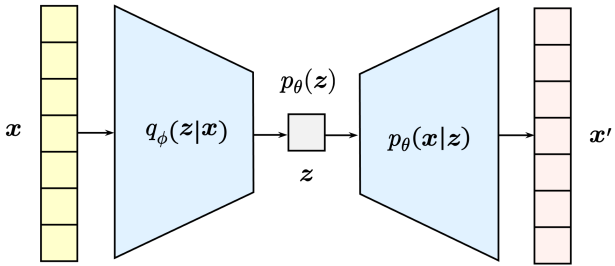
Since $p_\theta(\boldsymbol{x})$ is calculated by integrating z in latent space, which requires exponential time. So, $p_\theta(z|\boldsymbol{x})$ is a *computational intractability problem*.

Variational Autoencoders

VAE uses *variational inference* $q_\phi(\boldsymbol{z}|\boldsymbol{x}) \approx p_\theta(\boldsymbol{z}|\boldsymbol{x})$ to obtain the approximate result.

Therefore, the $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ is treated as probabilistic encoder, i.e.,

$$\boldsymbol{z} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}) \quad \text{and} \quad \boldsymbol{x}' \sim p_\theta(\boldsymbol{x}|\boldsymbol{z}).$$

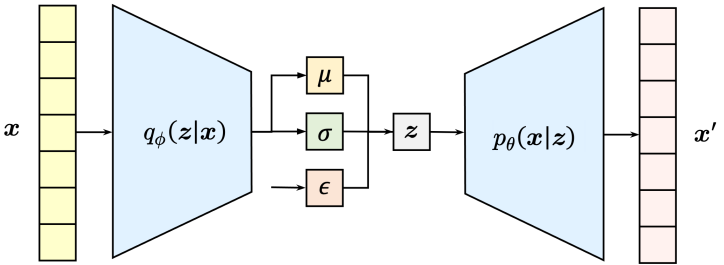


Variational Autoencoders

Re-Parameterization Trick:

It is a differentiable transformation of z into ϕ, $\boldsymbol{x}$, and ϵ, that is $z = f(\phi, \boldsymbol{x}, \epsilon)$. Suppose $z \sim q_\phi(z|\boldsymbol{x}) = \mathcal{N}(\mu, \sigma^2)$. Given $\epsilon \sim \mathcal{N}(0, 1)$, then by re-parameterization:

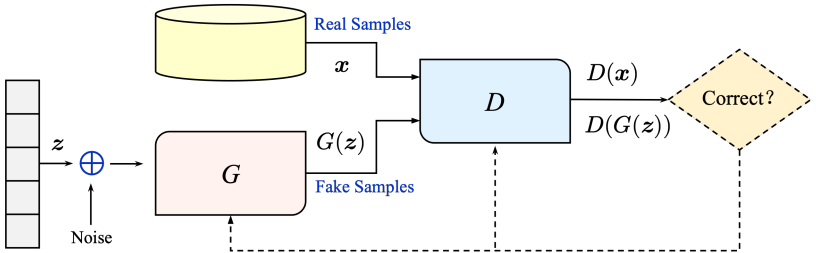
$$z = \mu + \sigma \odot \epsilon.$$



Generative Adversarial Models

Generative Adversarial Networks:

Generative adversarial network (GAN) is based on the minimax theorem in zero-sum game, a generator and a discriminator operate in a manner of two player game.



The structure of GAN, consists of a generator (G) and a discriminator (D), both of which use multi-layer neural networks.

Generative Adversarial Networks

The training process:

G first generates a fake sample $G(z)$, then D takes a real sample $\boldsymbol{x}$ and $G(z)$ as inputs, and obtains $D(\boldsymbol{x})$ and $D(G(z))$ respectively.

The G tries to make $G(z)$ true, while the D tries to make $D(\boldsymbol{x})$ true and $D(G(z))$ false.

Since GAN is based on the minimax theorem, that is:

$$\min_G \max_D V(G,D) = E_{x\sim P_{\text{data}}(x)}\left[\log D(x)\right] + E_{z\sim P_z(z)}\left[\log\left(1 - D(G(z))\right)\right].$$

The training process of these two modules is like a game between two players. After multiple rounds of adversarial training, their performance is continuously optimized.

When the D finally cannot distinguish the difference between $D(\boldsymbol{x})$ and $D(G(z))$, them reach a zero-sum state, it is considered that the GAN has reached its optimum.

Normalizing Flow Models

Normalizing Flow Models:

A normalized flow model uses the change of variables in probability to transform complex distributions into simple (normal) distributions, and then performs inverse transformations to restore simple (normal) to complex distributions.

Change of Variables:

It refers to a basic technique in mathematics to simplify problems, in which the original variables are replaced with functions of other variables.

Let $\boldsymbol{z} \in \mathbb{R}^D$ be a latent variable, $p_\theta : \mathbb{R}^D \to \mathbb{R}^D$ be its probability density function: $\boldsymbol{z} \sim p_\theta(\boldsymbol{z})$. Where $p_\theta(\boldsymbol{z})$ is a simple probability distribution, e.g. Gaussian distribution.

Using invertible function $g_\theta(\boldsymbol{z})$ to generate $\boldsymbol{x} \in \mathbb{R}^D$, and through its inverse function ${g_\theta}^{-1}(\boldsymbol{x})$ to transform $\boldsymbol{x}$ back to $\boldsymbol{z}$, i..e., $\boldsymbol{x} = g_\theta(\boldsymbol{z})$ and $\boldsymbol{z} = {g_\theta}^{-1}(\boldsymbol{x})$.

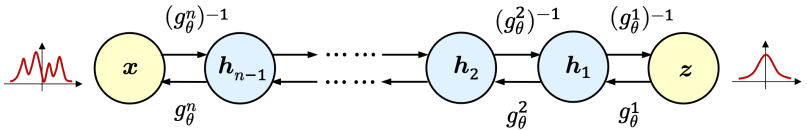
Normalizing Flow Models

Let probability density function $g_\theta = g_\theta^n \circ \cdots \circ g_\theta^2 \circ g_\theta^1$, then $\boldsymbol{x} = g_\theta(\boldsymbol{z})$ can be written as:

$$\boldsymbol{x} = g_\theta(\boldsymbol{z}) = g_\theta^n \circ \cdots \circ g_\theta^2 \circ g_\theta^1(\boldsymbol{z}).$$

Similarly, let the inverse probability density function ${g_\theta}^{-1} = (g_\theta^1)^{-1} \circ (g_\theta^2)^{-1} \circ \cdots \circ (g_\theta^n)^{-1}$, then $\boldsymbol{z} = {g_\theta}^{-1}(\boldsymbol{x})$ will be:

$$\boldsymbol{z} = {g_\theta}^{-1}(\boldsymbol{x}) = (g_\theta^1)^{-1} \circ (g_\theta^2)^{-1} \circ \cdots \circ (g_\theta^n)^{-1}(\boldsymbol{x}).$$



Autoregressive Models

Autoregressive Property:

Given a set of variables $\{x_1, x_2, \ldots, x_t\}$ over time t, it is called autoregressive property if x_t depends linearly on some or all its previous variables $\{x_1, x_2, \ldots, x_{t-1}\}$.

***k*-Order Autoregressive Models**:

A k-order autoregressive model AR(k) is a statistical model with the autoregressive property, which is defined as follows:

$$x_t = \sum_{i=1}^{k} \varphi_i x_{t-i} + \varepsilon_t.$$

Where, $\varphi_1, \ldots, \varphi_k$ are the parameters of the model, and ε_t is white noise.

And, if $k = t$, then $x_t = (x_1 + x_2 + \ldots + x_{t-1}) + \varepsilon_t$.

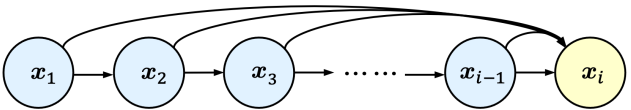
Autoregressive Models

Deep Autoregressive Models:

A deep autoregressive model (DAM) is a deep network model with autoregressive property, can be modeled based on the chain rule of joint probability.

Given a set of random variables $\{\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t\}$ over time t, and $p_\theta(\cdot)$ denotes a probability density function, then its joint probability can be expressed as:

$$p_\theta(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_t) = \prod_{i=1}^{t} p_\theta\left(\boldsymbol{x}_i | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{i-1}\right).$$



The directed graph of $p_\theta\left(\boldsymbol{x}_i | \boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_{i-1}\right)$.

Diffusion Models

Diffusion models are inspired by diffusion principle in nonequilibrium thermodynamics.

Diffusion:

It is the net process of movement of molecules or particles under a concentration gradient, generally from a region of higher concentration to a region of lower concentration.

E.g., a tea bag immersed in a cup of hot water will diffuse into the water and change its color.

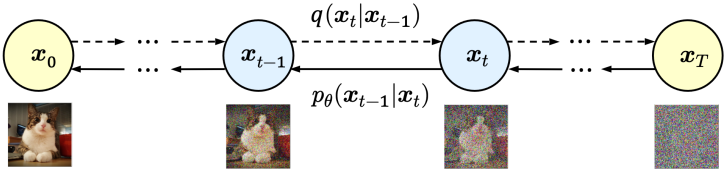
Diffusion Probabilistic Models:

A diffusion probabilistic model is a type of latent variable generative model, that consists of the forward diffusion process to slowly add random noise to data and the reverse diffusion process to reconstruct desired data from the noise.

Denoising Diffusion Models

Denoising Diffusion Probability Models:

A denoising diffusion probability model uses a Markov chain that gradually adds Gaussian noise to original image to obtain an approximate posterior probability $q\left(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0\right)$. The joint probability distribution $p_\theta\left(\boldsymbol{x}_{0:T}\right)$ is its reverse process, learning a Gaussian transition, starting from $p(\boldsymbol{x}_T) = \mathcal{N}(\boldsymbol{x}_T; 0, \mathbf{I})$.



Schematic diagram of the denoising diffusion probability model.

Denoising Diffusion Processes

Forward Diffusion Process:

$$q\left(\boldsymbol{x}_{1:T}|\boldsymbol{x}_0\right) = \prod_{t=1}^{T} q\left(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}\right) = \prod_{t=1}^{T} \mathcal{N}\left(\boldsymbol{x}_t; \sqrt{1-\beta_t}\boldsymbol{x}_{t-1}, \beta_t \mathbf{I}\right).$$

Reverse Diffusion Process:

$$p_\theta\left(\boldsymbol{x}_{0:T}\right) = p\left(\boldsymbol{x}_T\right) \prod_{t=T}^{1} p_\theta\left(\boldsymbol{x}_{t-1}|\boldsymbol{x}_t\right) = p\left(\boldsymbol{x}_T\right) \prod_{t=T}^{1} \mathcal{N}\left(\boldsymbol{x}_{t-1}; \mu_\theta(\boldsymbol{x}_t, t), \sum_\theta\left(\boldsymbol{x}_t, t\right)\right).$$

Thank You