Machine Learning

2024 (ML-2024)

Lecture 13. Unsupervised learning

by Alexei Valerievich Kornaev, Dr. habil. in Eng. Sc.,
Researcher at the RC for AI, Assoc. Prof. of the Robotics and CV Master’s Program, Innopolis University
Researcher at the RC for AI, National RC for Oncology n.a. NN Blohin
Professor at the Dept. of Mechatronics, Mechanics, and Robotics, Orel State University

ML-2024
by Alexei Kornaev

Agenda

Recap: probability distributions

The *sum* rule $\underline{p(x) = \int p(x,y)\,dy}$, and its generalization:

$$p(x_1,\dots,x_m) = \int p(x_1,\dots,x_m,x_{m+1},\dots,x_k)\,dx_{m+1}\dots dx_k.$$

The *product* rule $p(x,y) \underline{= p(x|y)p(y)}$, and its generalization:

$$p(x_1,\dots,x_m) = p(x_m|x_1,\dots,x_{m-1})p(x_{m-1}|x_1,\dots,x_{m-2})\dots p(x_2|x_1)p(x_1).$$

The *Bayes'* theorem:

$$p(y|x) = \frac{p(x|y)p(y)}{\int p(x,y)\,dy},$$

$$Posterior = \frac{Likelihood \cdot Prior}{Evidence}.$$

The *normal* or *Gaussian* distribution:

$$p(x) = N(x|\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\underline{\sigma^2}}} exp\left(-\frac{1}{2\sigma^2}(x-\underline{\mu})^2\right),$$

$$\int_{-\infty}^{+\infty} p(x)dx = \int_{-\infty}^{+\infty} N(x|\mu,\sigma^2)dx = 1.$$

The *expectation* of $f(x) = x$ under the probability distribution:

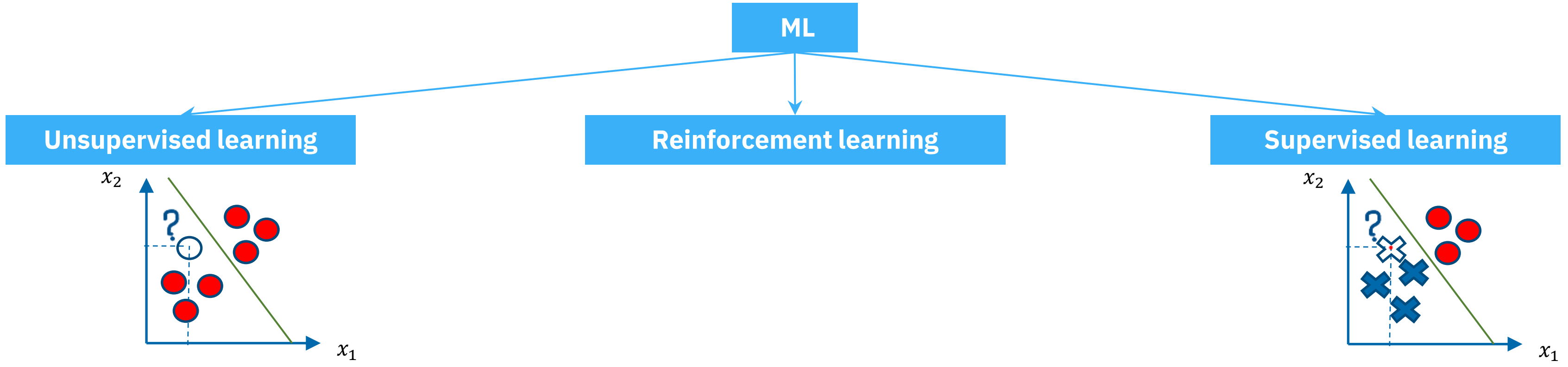
$$E[x] = \int N(x|\mu,\sigma^2)xdx = \mu.$$

The *variance* of the function $f(x) = x$ is:

$$var[x] = E[x^2] - E[x]^2 = \sigma^2.$$

Pattern Recognition and Machine Learning by Ch. Bishop, 2006

Recap: decision making in ML



Summary of Methods for Unsupervised Classification

- **Clustering Algorithms**: Techniques like K-Means, Hierarchical Clustering, DBSCAN, and Gaussian Mixture Models group similar data points together.
- **Self-Organizing Maps (SOM)**: Neural networks that map high-dimensional data into a lower-dimensional grid for clustering and visualization.
- **Autoencoders**: Neural networks that compress and reconstruct data, enabling clustering in the latent space.
- **Generative Adversarial Networks (GANs)**: Neural networks that generate synthetic data, allowing clustering on the generated samples.
- **Dimensionality Reduction**: Methods like PCA, t-SNE, and UMAP reduce data dimensions while preserving structure, aiding in clustering.
- **Density-Based Methods**: Techniques like Mean Shift and OPTICS identify clusters based on data density.
- **Graph-Based Methods**: Spectral Clustering and Community Detection algorithms analyze graph structures for clustering.
- **Semi-Supervised Learning**: Approaches like Self-Training, Co-Training, and Graph-Based methods use a small amount of labeled data to improve clustering on unlabeled data.

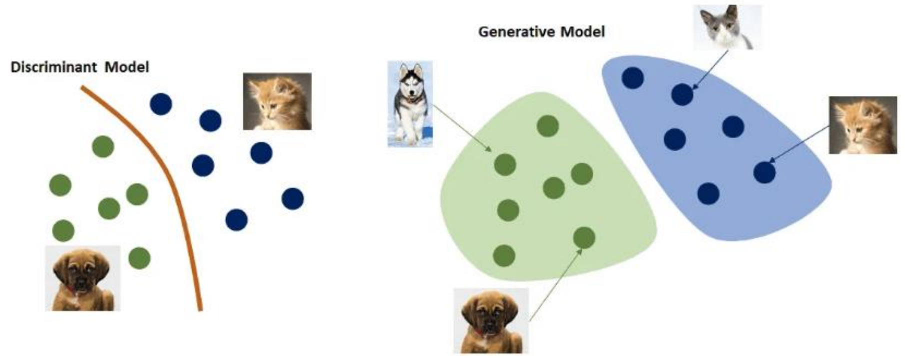
By leveraging these methods, one can uncover hidden structures in the data and make informed decisions without prior knowledge of the categories.

/Deep Seek speaks/

Discriminative vs Generative models

A **discriminative** model intuition: given $(\boldsymbol{x}, y)$, find a distribution $p(y|\boldsymbol{x})$ to make a prediction h

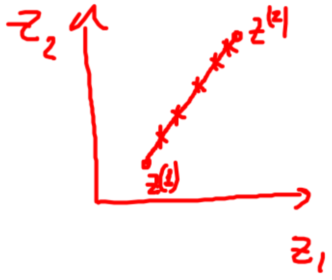
A **generative** model intuition: given y, find a distribution $p(\boldsymbol{x}|y)$ or $p(\boldsymbol{x})$ to make a sampling $\boldsymbol{x}$ from the distribution

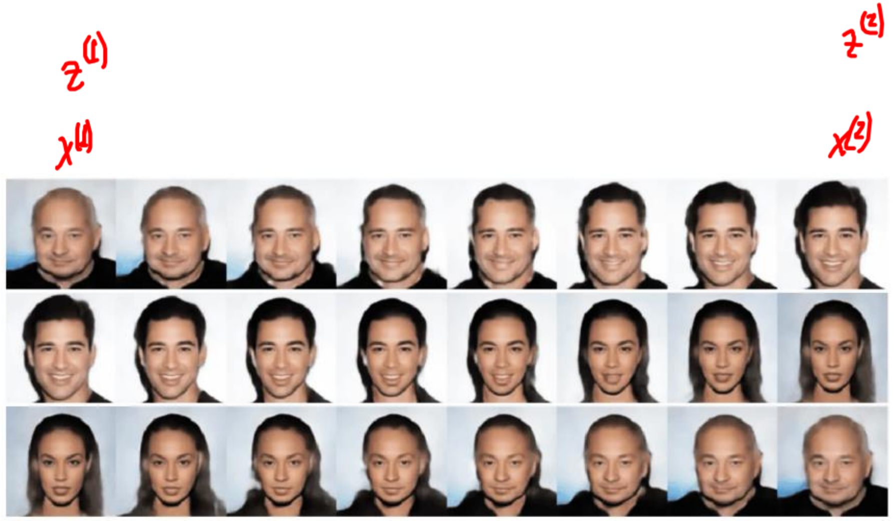


Yandex Handbook on ML

Generative models: interpolation in a latent space

Given a latent space characterized with a vector $\boldsymbol{z}$. Two samples $\boldsymbol{x}^{(1)} = \boldsymbol{G}(\boldsymbol{z}^{(1)})$ and $\boldsymbol{x}^{(2)} = \boldsymbol{G}(\boldsymbol{z}^{(2)})$ generated from the latent space can be connected with the line which corresponds with a set of interpolation points and samples $\boldsymbol{x}^{(i)} = \boldsymbol{G}(\boldsymbol{z}^{(i)})$.

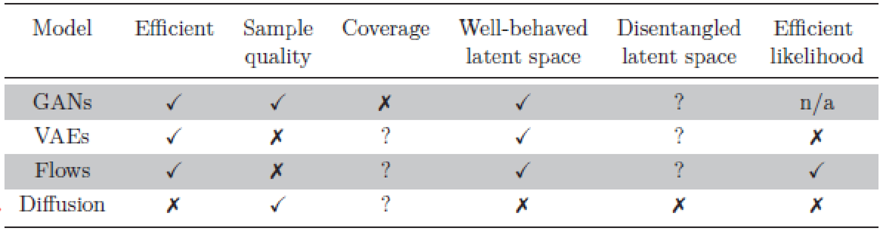




[Yandex Handbook on ML]

Generative models: what makes a good one?

- **Efficient sampling:** Generating samples from the model should be computationally inexpensive and take advantage of the parallelism.
- **High-quality sampling:** The samples should be indistinguishable from the real data with which the model was trained.
- **Coverage:** Samples should represent the entire training distribution. It is insufficient to generate samples that all look like a subset of the training examples.
- **Well-behaved latent space:** A latent variable **z** corresponds to a plausible data example **x**. Smooth changes in **z** correspond to smooth changes in **x**.
- **Disentangled latent space:** Manipulating each dimension of **z** should correspond to changing an interpretable property of the data. For example, in a model of language, it might change the topic, tense, or verbosity.
- **Efficient likelihood computation:** If the model is probabilistic, we would like to be able to calculate the probability of new examples efficiently and accurately.



Model	Efficient	Sample quality	Coverage	Well-behaved latent space	Disentangled latent space	Efficient likelihood
GANs	✓	✓	✗	✓	?	n/a
• VAEs	✓	✗	?	✓	?	✗
Flows	✓	✗	?	✓	?	✓
+ Diffusion	✗	✓	?	✗	✗	✗

Figure 14.3 Properties of four generative models. Neither generative adversarial networks (GANs), variational autoencoders (VAEs), normalizing flows (Flows), nor diffusion models (diffusion) have the full complement of desirable properties.

Simon J.D. Prince. Understanding Deep learning, MIT Press, 2024

Generative models: applications

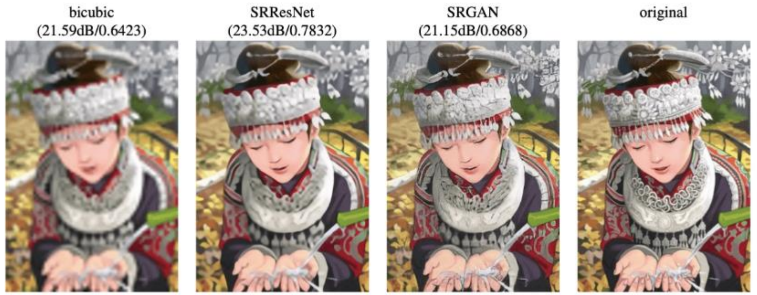


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

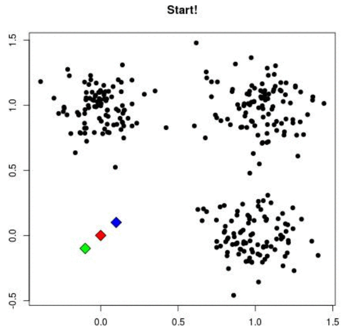
[Yandex Handbook on ML]

Agenda

Clustering algorithm

Кластерный анализ - статистическая процедура, выполняющая сбор данных, содержащих информацию о выборке объектов, и затем упорядочивающая объекты в сравнительно однородные группы (кластеры).

Метод k-средних. Основная **идея** метода в разделении данных на «k» кластеров по признаку наим. расстояния до одного из центров кластеров. При этом положение центров кластеров итерационно пересчитывается.



Алгоритм обучения.

1. Случайным образом назначаются центры кластеров $\mu_i^{(k)} (i = 1,2, \dots N)$, N – мерность пространства.
2. Определяется принадлежность каждой точки $x_i^{(j)}$ $(i = 1,2, \dots m)$ к одному из кластеров по признаку наименьшей из «k» величин: $\underline{\min_k \sum_{i=1}^m \left(x_i^{(j)} - \mu_i^{(k)}\right)^2}$, m – количество точек.
3. Вычисляются новые координаты каждого кластера $\mu_i^{(k)}$ как средние арифметические координат элементов данного кластера.
4. Пункты 2,3 повторяются, минимизируется функция: $L = \frac{1}{m} \sum_{i=1}^m \left(x_i^{(j)} - \mu_i^{(c_i)}\right)^2$.

Agenda

ML-2024
by Alexei Kornaev

Dimensionality reduction

Principal Component Analysis (PCA) $\longrightarrow \vec{y}\cdot T_a = \lambda \vec{y} \quad \vec{y}\cdot(T_a - \lambda T_\delta) = 0$

t-Distributed Stochastic Neighbor Embedding (t-SNE)

Uniform Manifold Approximation and Projection (UMAP)

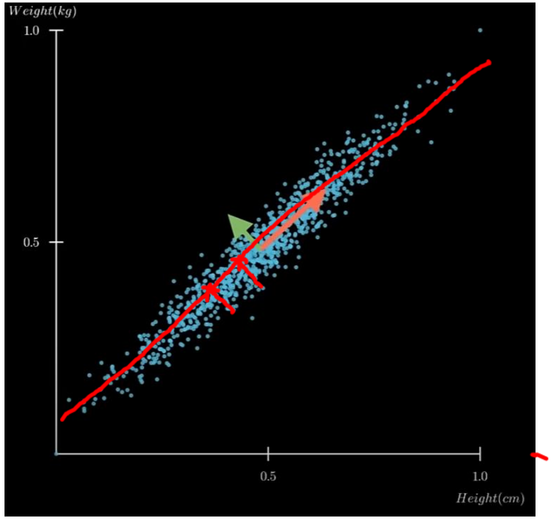
$|T_a - \lambda T_\delta| = 0$

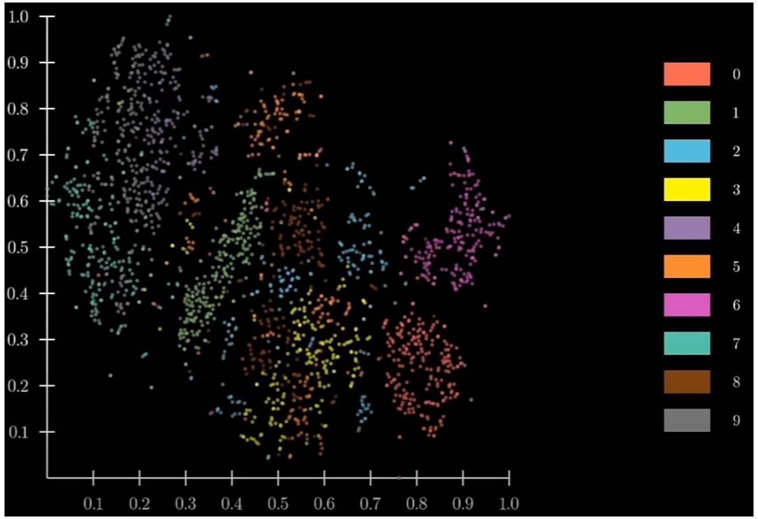
ranging $\lambda_1\ \lambda_2\ \lambda_3$

$a_1 \geq a_2 \geq a_3$

$$T_a = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

$$T_a = \begin{bmatrix} a_1 & 0 & 0 \\ 0 & a_2 & 0 \\ 0 & 0 & a_3 \end{bmatrix}$$

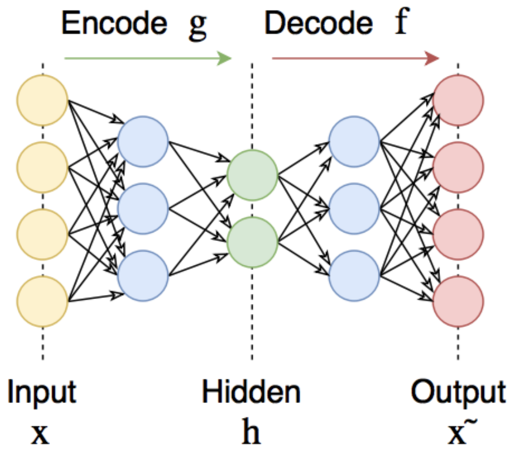




Latent space visualization by Deepia

Agenda

AE architecture and intuition



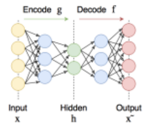
Автоэнкодеры в Keras. Статья в Habr.

AE algorithm

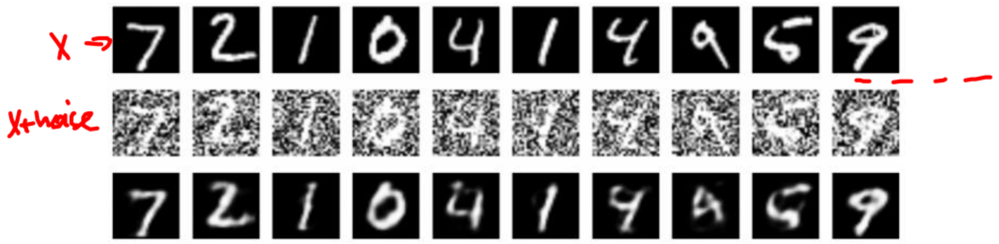
Step	Description
1. Data Preparation	- **Input Data**: Prepare the input data (e.g., images, text). - **Normalization**: Normalize the data if necessary. - **Batching**: Divide the data into batches for training.
2. Model Definition	- **Encoder**: Define the encoder part of the autoencoder, which compresses the input data into a lower-dimensional representation. - **Decoder**: Define the decoder part, which reconstructs the input data from the compressed representation.
3. Loss Function	- **Reconstruction Loss**: Choose a loss function to measure the difference between the input data and the reconstructed data (e.g., Mean Squared Error for continuous data, Cross-Entropy for binary data).
4. Optimization	- **Optimizer**: Choose an optimization algorithm (e.g., Adam, SGD) to minimize the loss function. - **Learning Rate**: Set the learning rate for the optimizer.
5. Training Loop	- **Epochs**: Set the number of epochs for training. - **Batch Processing**: For each epoch, iterate through the batches of data. - **Forward Pass**: Pass the input data through the encoder and decoder to get the reconstructed data. - **Loss Calculation**: Compute the loss between the input data and the reconstructed data. - **Backward Pass**: Compute the gradients of the loss with respect to the model parameters. - **Parameter Update**: Update the model parameters using the optimizer.

Deep Seek speaks

AE applications



1. Noise reduction



2. Feature extractor, dim reduction.

3. Anomaly detection $L \geq tr \rightarrow$ anomaly

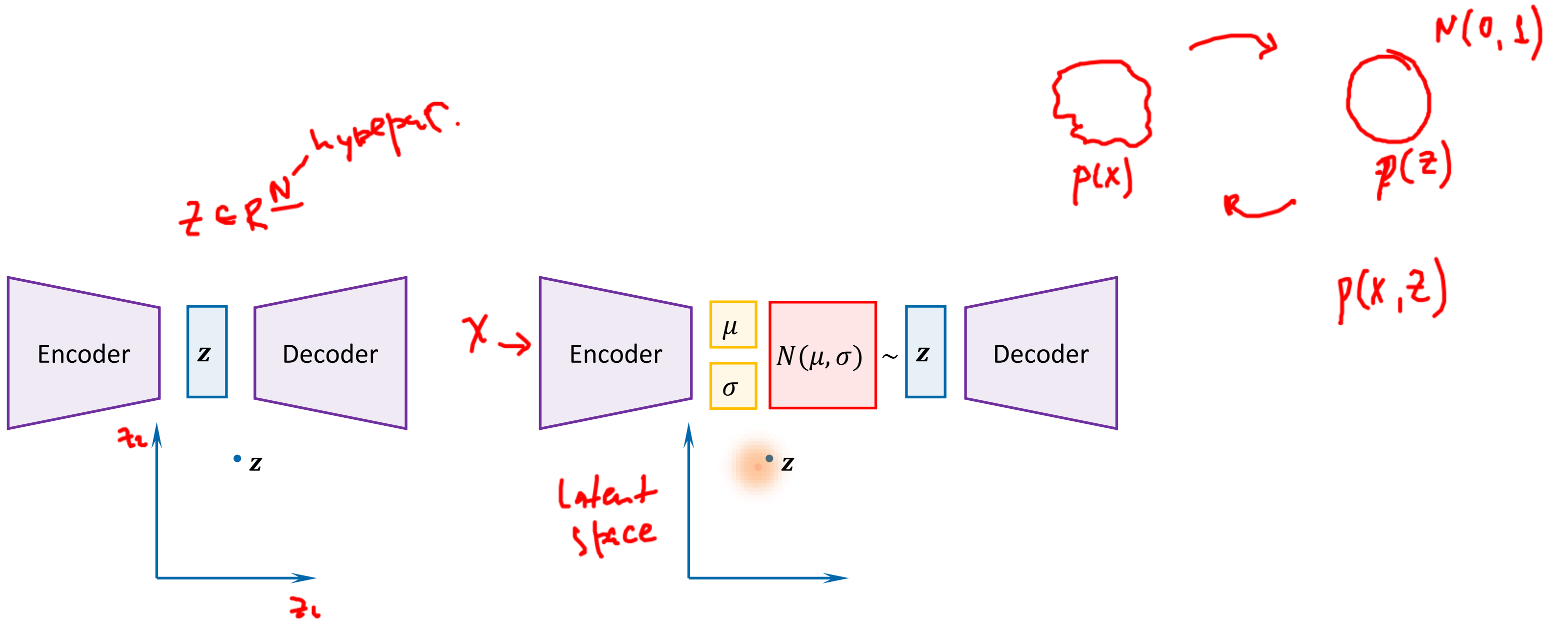
4. Pretraing of the encoder

Автоэнкодеры в Keras. Статья в Habr.

Agenda

VAE: intuition

The main task of a generative model is to learn the underlying distribution of the input data and generate new data samples that are similar to the original data. In essence, generative models aim to model the data-generating process, allowing them to produce synthetic data that mimics the characteristics and patterns of the real data.



AE (left) and VAE (right) intuition

VAE: generation from a latent variable model

Let's draw $\boldsymbol{z}^*$ from the prior $p_{\boldsymbol{\varphi}}(\boldsymbol{z})$ and pass this through the model $\boldsymbol{f}[\boldsymbol{z}^*, \boldsymbol{\varphi}]$ (decoder) to compute the mean of the likelihood $\underline{p(\boldsymbol{x}|\boldsymbol{z}^*)}$, from which we draw $\boldsymbol{x}^*$. Both the prior and the likelihood are normal distributions:

$$p_{\boldsymbol{\varphi}}(\boldsymbol{x}) = \int p_{\boldsymbol{\varphi}}\underline{(\boldsymbol{x}, \boldsymbol{z})}\, d\boldsymbol{z} = \int p_{\boldsymbol{\varphi}}(\boldsymbol{x}|\boldsymbol{z}) p_{\boldsymbol{\varphi}}(\boldsymbol{z})\, d\boldsymbol{z} = \int N_{\boldsymbol{x}}[\boldsymbol{\mu}, \boldsymbol{\sigma}^2 \boldsymbol{I}] N_{\boldsymbol{z}}[\boldsymbol{0}, \boldsymbol{I}]\, d\boldsymbol{z}$$

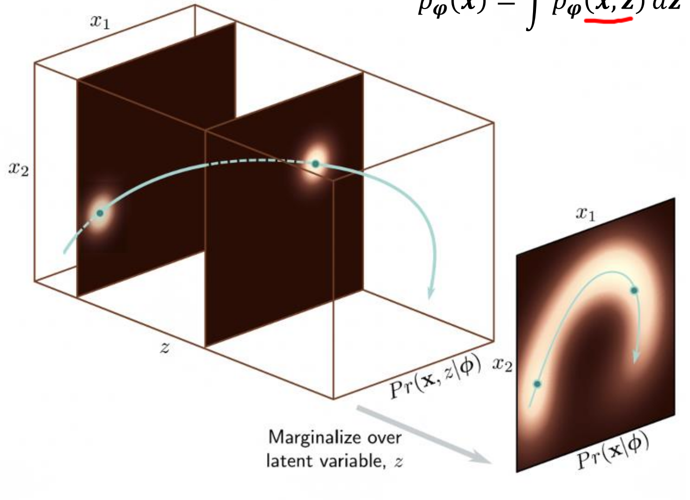


Figure 17.2 Nonlinear latent variable model. A complex 2D density $Pr(\mathbf{x})$ (right) is created as the marginalization of the joint distribution $Pr(\mathbf{x}, z)$ (left) over the latent variable z; to create $Pr(\mathbf{x})$, we integrate the 3D volume over the dimension z. For each z, the distribution over $\mathbf{x}$ is a spherical Gaussian (two slices shown) with a mean $\mathbf{f}[z, \phi]$ that is a nonlinear function of z and depends on parameters ϕ. The distribution $Pr(\mathbf{x})$ is a weighted sum of these Gaussians.

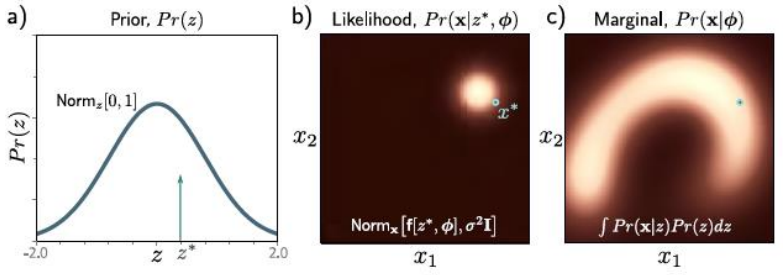


Figure 17.3 Generation from nonlinear latent variable model. a) We draw a sample z^* from the prior probability $Pr(z)$ over the latent variable. b) A sample $\mathbf{x}^*$ is then drawn from $Pr(\mathbf{x}|z^*, \phi)$. This is a spherical Gaussian with a mean that is a nonlinear function $\mathbf{f}[\bullet, \phi]$ of z^* and a fixed variance $\sigma^2\mathbf{I}$. c) If we repeat this process many times, we recover the density $Pr(\mathbf{x}|\phi)$.

Simon J.D. Prince. Understanding Deep learning, MIT Press, 2024

Evidence Lower Bound (ELBO) loss

The goal is to maximize log-likelihood: $log[p_\varphi(\boldsymbol{x})] = log[\int p_\varphi(\boldsymbol{x},\boldsymbol{z})\, d\boldsymbol{z}] = log\left[\int q_\theta(\boldsymbol{z}) \frac{p_\varphi(\boldsymbol{x},\boldsymbol{z})}{q_\theta(\boldsymbol{z})}\, d\boldsymbol{z}\right].$

Use Jensen's inequality for the logarithm to find a lower bound: $log\left[\int q_\theta(\boldsymbol{z}) \frac{p_\varphi(\boldsymbol{x},\boldsymbol{z})}{q_\theta(\boldsymbol{z})}\, d\boldsymbol{z}\right] \geq \underline{\int q_\theta(\boldsymbol{z}) log\left[\frac{p_\varphi(\boldsymbol{x},\boldsymbol{z})}{q_\theta(\boldsymbol{z})}\right] d\boldsymbol{z}}.$

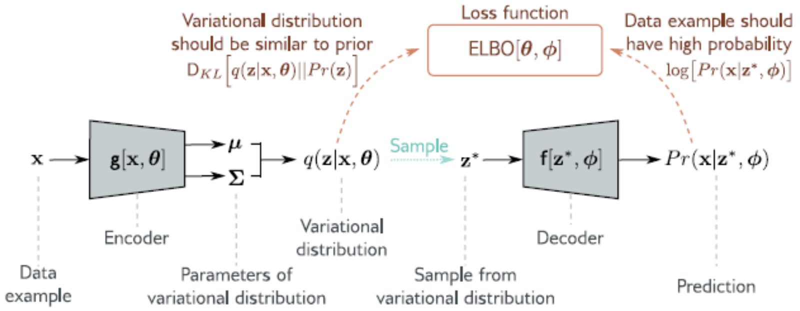

Variational distribution should be similar to prior
$D_{KL}[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})||Pr(\mathbf{z})]$
Loss function
ELBO$[\boldsymbol{\theta},\phi]$
Data example should have high probability
$\log[Pr(\mathbf{x}|\mathbf{z}^*,\phi)]$
$\mathbf{x}$
$\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$
$\boldsymbol{\mu}$
$\boldsymbol{\Sigma}$
$q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$
Sample
$\mathbf{z}^*$
$\mathbf{f}[\mathbf{z}^*,\phi]$
$Pr(\mathbf{x}|\mathbf{z}^*,\phi)$
Encoder
Variational distribution
Decoder
Data example
Parameters of variational distribution
Sample from variational distribution
Prediction

Figure 17.9 Variational autoencoder. The encoder $\mathbf{g}[\mathbf{x},\theta]$ takes a training example $\mathbf{x}$ and predicts the parameters μ, Σ of the variational distribution $q(\mathbf{z}|\mathbf{x},\theta)$. We sample from this distribution and then use the decoder $\mathbf{f}[\mathbf{z},\phi]$ to predict the data $\mathbf{x}$. The loss function is the negative ELBO, which depends on how accurate this prediction is and how similar the variational distribution $q(\mathbf{z}|\mathbf{x},\theta)$ is to the prior $Pr(\mathbf{z})$ (equation 17.21).

$$\text{ELBO}(\boldsymbol{\theta},\boldsymbol{\varphi}) = \int q_\theta(\boldsymbol{z}) log\left[\frac{p_\varphi(\boldsymbol{x},\boldsymbol{z})}{q_\theta(\boldsymbol{z})}\right] d\boldsymbol{z}$$

$$= \int q_\theta(\boldsymbol{z}) log\left[\frac{p_\varphi(\boldsymbol{z}|\boldsymbol{x}) p_\varphi(\boldsymbol{x})}{q_\theta(\boldsymbol{z})}\right] d\boldsymbol{z}$$

$$= \int q_\theta(\boldsymbol{z}) log[p_\varphi(\boldsymbol{x})]\, d\boldsymbol{z} + \int q_\theta(\boldsymbol{z}) log\left[\frac{p_\varphi(\boldsymbol{z}|\boldsymbol{x})}{q_\theta(\boldsymbol{z})}\right] d\boldsymbol{z}$$

$$= log[p_\varphi(\boldsymbol{x})] - D_{KL}[q_\theta(\boldsymbol{z})||p_\varphi(\boldsymbol{z}|\boldsymbol{x})].$$

Simon J.D. Prince. Understanding Deep learning, MIT Press, 2024

Evidence Lower Bound (ELBO) loss

The goal is to maximize log-likelihood: $log[p_{\boldsymbol{\varphi}}(\boldsymbol{x})] = log[\int p_{\boldsymbol{\varphi}}(\boldsymbol{x},\boldsymbol{z})\,d\boldsymbol{z}] = log\left[\int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\frac{p_{\varphi}(x,z)}{q_{\theta}(z|x)}\,d\boldsymbol{z}\right]$.

Use Jensen’s inequality for the logarithm to find a lower bound: $log\left[\int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\frac{p_{\varphi}(x,z)}{q_{\theta}(z|x)}\,d\boldsymbol{z}\right] \geq \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\,log\left[\frac{p_{\varphi}(x,z)}{q_{\theta}(z|x)}\right]d\boldsymbol{z}$.

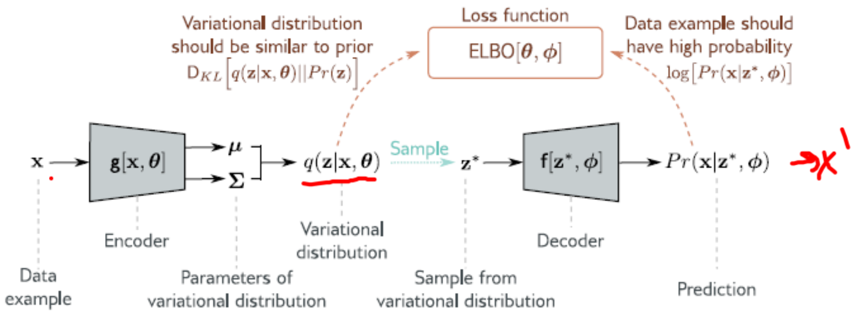

Variational distribution should be similar to prior
$D_{KL}[q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})||Pr(\mathbf{z})]$
Loss function
$ELBO[\boldsymbol{\theta},\phi]$
Data example should have high probability
$\log[Pr(\mathbf{x}|\mathbf{z}^*,\phi)]$
$\mathbf{x}$
$\mathbf{g}[\mathbf{x},\boldsymbol{\theta}]$
μ
Σ
$q(\mathbf{z}|\mathbf{x},\boldsymbol{\theta})$
Sample
$\mathbf{z}^*$
$\mathbf{f}[\mathbf{z}^*,\phi]$
$Pr(\mathbf{x}|\mathbf{z}^*,\phi)$
$\to x'$
Encoder
Variational distribution
Decoder
Data example
Parameters of variational distribution
Sample from variational distribution
Prediction

Figure 17.9 Variational autoencoder. The encoder $\mathbf{g}[\mathbf{x},\theta]$ takes a training example $\mathbf{x}$ and predicts the parameters μ,Σ of the variational distribution $q(\mathbf{z}|\mathbf{x},\theta)$. We sample from this distribution and then use the decoder $\mathbf{f}[\mathbf{z},\phi]$ to predict the data $\mathbf{x}$. The loss function is the negative ELBO, which depends on how accurate this prediction is and how similar the variational distribution $q(\mathbf{z}|\mathbf{x},\theta)$ is to the prior $Pr(\mathbf{z})$ (equation 17.21).

$$\mathrm{L}(\boldsymbol{\theta},\boldsymbol{\varphi}) = \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\,log\left[\frac{p_{\boldsymbol{\varphi}}(\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]d\boldsymbol{z}$$

$$= \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\,log\left[\frac{p_{\boldsymbol{\varphi}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})p_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}\right]d\boldsymbol{z}$$

$$= \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\,log\left[\frac{p_{\boldsymbol{\varphi}}(\boldsymbol{x},\boldsymbol{z})}{q_{\boldsymbol{\theta}}(\boldsymbol{z})}\right]d\boldsymbol{z} + \int q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})\,log\left[\frac{p_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})}{q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})}\right]d\boldsymbol{z}$$

$$= ELBO + D_{KL}[q_{\boldsymbol{\theta}}(\boldsymbol{z}|\boldsymbol{x})||p_{\boldsymbol{\varphi}}(\boldsymbol{z}|\boldsymbol{x})]$$

Simon J.D. Prince. Understanding Deep learning, MIT Press, 2024

VAE: algorithm

Step	Description
1. Data Preparation	- Prepare input data (e.g., images, text). - Normalize data if necessary. - Divide data into batches.
2. Model Definition	- **Encoder**: Map input data to latent distribution parameters (mean $\underline{\mu}$ and log variance $\underline{\log\sigma^2}$). - **Reparameterization Trick**: Sample latent vector $\underline{z \text{ using } z=\mu+\sigma\cdot\epsilon}$. - **Decoder**: Map latent vector z back to data space.
3. Loss Function	- **Reconstruction Loss**: Measure difference between input and reconstructed data (e.g., MSE for continuous, BCE for binary). - **KL Divergence**: Measure difference between learned latent distribution and prior distribution (standard normal). - **Total Loss**: Combine reconstruction loss and KL divergence.
4. Optimization	- Choose optimizer (e.g., Adam, SGD). - Set learning rate.
5. Training Loop	- Set number of epochs. - Iterate through batches: 1. **Encoding**: Obtain μ and $\log\sigma^2$. 2. **Sampling**: Use reparameterization trick to sample *z*. 3. **Decoding**: Reconstruct data. 4. **Loss Calculation**: Compute reconstruction loss and KL divergence. 5. **Backward Pass**: Compute gradients and update parameters. - Optionally, log loss values.

Deep Seek speaks

AE vs VAE: architecture

Feature	**Autoencoder (AE)**	**Variational Autoencoder (VAE)**
Objective	Learn a compressed representation (encoding) of the input data.	Learn a probabilistic model of the data distribution in the latent space.
Latent Space	Deterministic: Each input is mapped to a single point in the latent space.	Stochastic: Each input is mapped to a distribution (usually Gaussian) in the latent space.
Encoder	Outputs a single vector (encoding) for each input.	Outputs parameters of a distribution (mean and variance) for each input.
Decoder	Takes the encoding as input and reconstructs the original data.	Takes a sample from the latent distribution (reparameterization trick) and reconstructs the original data.
Loss Function	Reconstruction loss (e.g., Mean Squared Error) to measure the difference between the input and the reconstructed data.	Combination of reconstruction loss and KL divergence to ensure the latent distribution matches a prior distribution (usually standard normal).

Deep Seek speaks

AE vs VAE: training process

Feature	Autoencoder (AE)	Variational Autoencoder (VAE)
Training Objective	Minimize the reconstruction error.	Minimize the reconstruction error and the KL divergence between the learned latent distribution and the prior distribution.
Reparameterization Trick	Not used.	Used to make the sampling process differentiable, allowing backpropagation through the latent space.
Regularization	Typically not regularized (unless using techniques like dropout).	Regularized to ensure the latent space is smooth and continuous, allowing for meaningful interpolation and generation of new data points.

Deep Seek speaks

AE vs VAE: applications

Feature	Autoencoder (AE)	Variational Autoencoder (VAE)
Dimensionality Reduction	Yes, AE can be used for dimensionality reduction by extracting the encoding.	Yes, VAE can also be used for dimensionality reduction, but the latent space is more structured and interpretable.
Data Generation	No, AE cannot generate new data points because the latent space is deterministic.	Yes, VAE can generate new data points by sampling from the latent space and decoding the samples.
Anomaly Detection	Yes, AE can be used for anomaly detection by identifying inputs with high reconstruction error.	Yes, VAE can also be used for anomaly detection, but the probabilistic nature of the latent space can provide additional insights.
Interpolation	Limited, as the latent space is not guaranteed to be smooth.	Better, as the latent space is regularized to be smooth and continuous, allowing for meaningful interpolation between data points.

Deep Seek speaks

Agenda

https://github.com/Mechanics-Mechatronics-and-Robotics/ML-2024/tree/main/Lecture_13

Deep Seek codes

Agenda

CONCLUSION

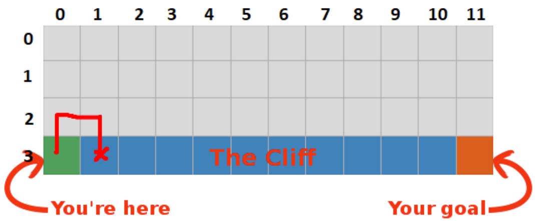
Generative models are powerful tools in AI with a wide range of applications, including **data augmentation, anomaly detection, content creation, and simulation**. They enable the creation of new data samples that mimic real-world data, enhancing the performance and capabilities of AI systems. However, they also face challenges related to computational complexity, quality and diversity trade-offs, training instability, evaluation difficulties, and ethical concerns. Addressing these challenges is crucial for the responsible and effective deployment of generative models in AI.

Deep Seek speaks

Thank you for your attention!

a.kornaev@innopolis.ru, @avkornaev

Пример. Задача о прогулке по скале: необходимо найти кратчайший путь до цели, не упав с обрыва. Награда за обычный шаг -1, за достижение цели +0, за срыв с обрыва -100. Дисконт 0.9. $S = [\![s^{(i,j)}]\!]$, $A = [\![a^{(k)}]\!] \rightarrow Q = [\![q^{(i,j,k)}]\!]$. Q-?



Алгоритм обучения. Инициализировать $Q(S,A)$ случайными значениями или нулями. Задать гиперпараметры: вероятность ϵ, скорость обучения α.
Для каждого эпизода обучения:

1. Получить данные о начальном состоянии S_t ($t = 1$)
2. Повторять для каждого шага t до достижения terminal state:
   - 2.1 Для текущего состояния S_t выбрать случайное действие A_t с вероятностью ϵ (может уменьшаться в процессе обучения), иначе действие, для которого значение наибольшее: $A_t = \max_A [Q_t(S_t, A_t)]$
   - 2.2 Выполнить действие A_t, получить награду r_t и данные о новом состоянии S_{t+1}.
   - 2.3 Если новое состояние S_{t+1} терминальное, то установить значение целевой функции $y_t = r_t$. Иначе $y_t = r_t + \gamma \max_A [q_{t+1}(S_{t+1}, A_{t+1})]$
   - 2.4 Обновить компоненту матрицы $Q(S,A)$:
     $q(S_t, A_t) = q(S_t, A_t) + \propto [y_t - q(S_t, A_t)]$.

	$a^{(1)}$ $\uparrow$	$a^{(2)}$ $\downarrow$	$a^{(3)}$ $\rightarrow$	$a^{(4)}$ $\leftarrow$
$s^{(0,0)}$	0	0	0	0
$s^{(0,1)}$	0	0	0	0
…	0	0	0	0
$s^{(2,0)}$	0	0	0	0
$s^{(2,1)}$	0	0	0	0
$s^{(2,2)}$	0	0	0	0
…	0	0	0	0
$s^{(3,0)}$	0	0	0	0
$s^{(3,1)}$	0	0	0	0
$s^{(3,2)}$	0	0	0	0
…	0	0	0	0
$s^{(3,11)}$	0	0	0	0

Реализация алгоритма.
Инициализация $Q(S,A)$ нулями. Задание: $\epsilon = 1$, $\alpha = 0.1$, $\gamma = 0.9$.
Эпизод 1. $\epsilon = 1$, тогда все действия 1 эпизода случайные.

1. $t = 1$. $\epsilon = 1$ (все действия случайные), $s_t = s^{(3,0)}$.
   - 2.1 Выбор сл. действия $a_t =$
   - 2.2 Вып. действия $a_t =$, награда $r_t =$, $s_{t+1} = s^{(2,0)}$.
   - 2.3 Расчет цел. функц. $y_t =$
   - 2.4 Обновление $q^{(3,0,1)} =$

   $t = 2$. $\epsilon = 1$ (все действия случайные), $s_t = s^{(2,0)}$.
   - 2.1 Выбор сл. действия $a_t =$
   - 2.2 Вып. действия $a_t =$, награда $r_t =$, $s_{t+1} = s^{(2,1)}$.
   - 2.3 Расчет цел. функц. $y_t =$
   - 2.4 Обновление $q^{(2,0,3)} =$

   $t = 3$. $\epsilon = 1$ (все действия случайные), $s_t = s^{(2,1)}$.
   - 2.1 Выбор сл. действия $a_t =$.
   - 2.2 Вып. действия $a_t =$, награда $r_t =$.
   - 2.3 Расчет цел. функц. $y_t =$
   - 2.4 Обновление $q^{(2,1,2)} =$

   ….

Agenda: Unsupervised Learning with VAEs and Diffusion Models

1. **Introduction to Unsupervised Learning** (10 minutes)
  - **Definition and Importance**: What is unsupervised learning? Why is it important?
  - **Types of Unsupervised Learning**: Clustering, Dimensionality Reduction, Generative Models.
  - **Applications**: Real-world applications of unsupervised learning.

2. **Clustering Techniques** (15 minutes)
  - **K-Means Clustering**: Algorithm, Advantages, and Limitations.
  - **Hierarchical Clustering**: Agglomerative and Divisive methods.
  - **Density-Based Clustering**: DBSCAN, OPTICS.

3. **Dimensionality Reduction** (15 minutes)
  - **Principal Component Analysis (PCA)**: Theory and Implementation.
  - **t-Distributed Stochastic Neighbor Embedding (t-SNE)**: Visualization technique.
  - **Uniform Manifold Approximation and Projection (UMAP)**: Comparison with t-SNE.

4. **Introduction to Generative Models** (10 minutes)
  - **Generative vs. Discriminative Models**: Key differences.
  - **Common Generative Models**: GANs, VAEs, Diffusion Models.

5. **Variational Autoencoders (VAEs)** (20 minutes)
  - **Basic Concepts**: Autoencoders, Latent Space, Reconstruction Error.
  - **Variational Inference**: ELBO (Evidence Lower Bound), KL Divergence.
  - **Training VAEs**: Loss Function, Backpropagation.
  - **Applications**: Image Generation, Anomaly Detection.

6. **Diffusion Models** (20 minutes)
  - **Basic Concepts**: Forward and Reverse Diffusion Processes.
  - **Mathematical Framework**: Stochastic Differential Equations (SDEs).
  - **Training Diffusion Models**: Loss Function, Optimization.
  - **Applications**: Image Synthesis, Denoising.

7. **Comparison of VAEs and Diffusion Models** (10 minutes)
  - **Similarities and Differences**: Architecture, Training, Applications.
  - **Advantages and Disadvantages**: Trade-offs in performance and complexity.

8. **Hands-On Session** (30 minutes)
  - **Implementing a Simple VAE**: Step-by-step implementation in PyTorch.
  - **Exploring Diffusion Models**: Overview of existing libraries (e.g., Denoising Diffusion Probabilistic Models).

9. **Q&A and Discussion** (10 minutes)
  - **Open Floor**: Addressing questions from the audience.
  - **Discussion**: Future directions in unsupervised learning.

10. **Conclusion** (5 minutes)
  - **Summary**: Recap of key points.
  - **Further Reading**: Recommended resources and papers.

Notes:
- **Visual Aids**: Use slides, diagrams, and code snippets to illustrate concepts.
- **Interactive Elements**: Include live coding or demos to engage the audience.
- **Time Management**: Ensure each section is covered within the allotted time.

This agenda provides a balanced overview of unsupervised learning techniques, with a deep dive into VAEs and Diffusion Models, making it suitable for an audience with a basic understanding of machine learning.