Deep Learning & Applied AI

PCA and VAEs

Emanuele Rodolà
rodola@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

2nd semester a.y. 2024/2025 · April 28, 2025

Generative models

Overall idea:

Learn a distribution from some given training samples, and generate new samples from the same distribution.

Generative models

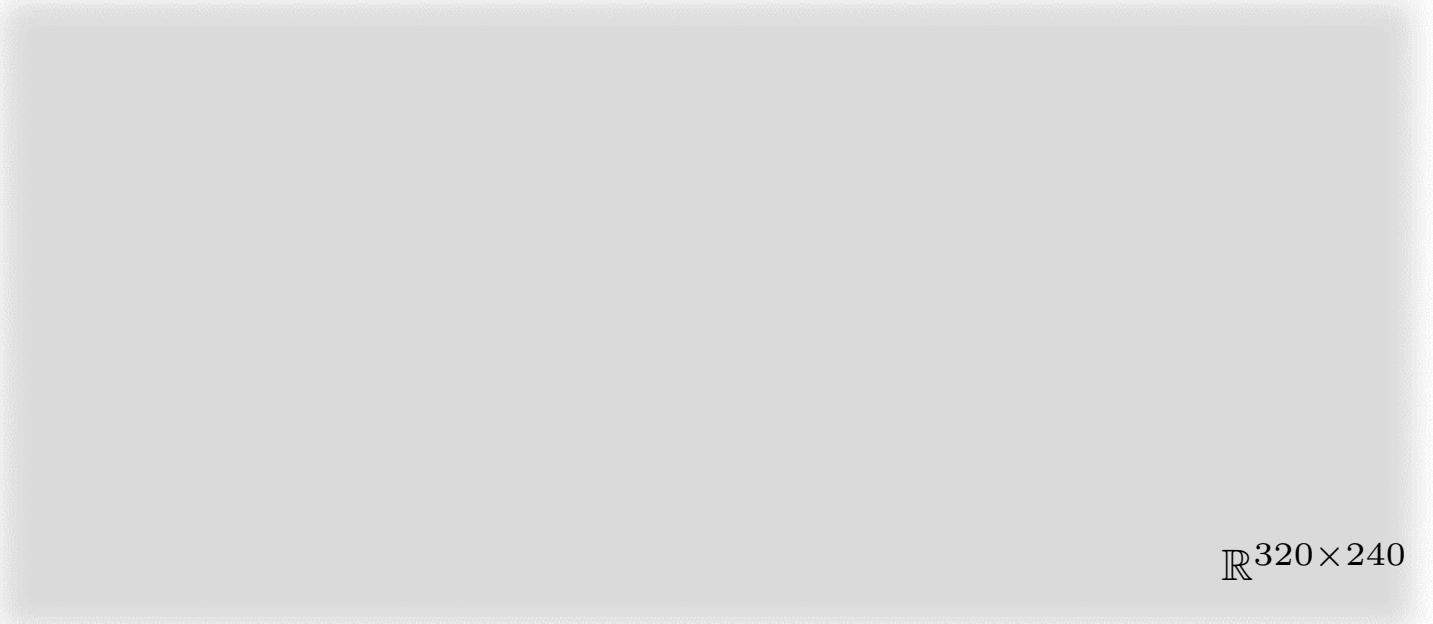
Overall idea:

Learn a distribution from some given training samples, and generate new samples from the same distribution.

https://this-person-does-not-exist.com/

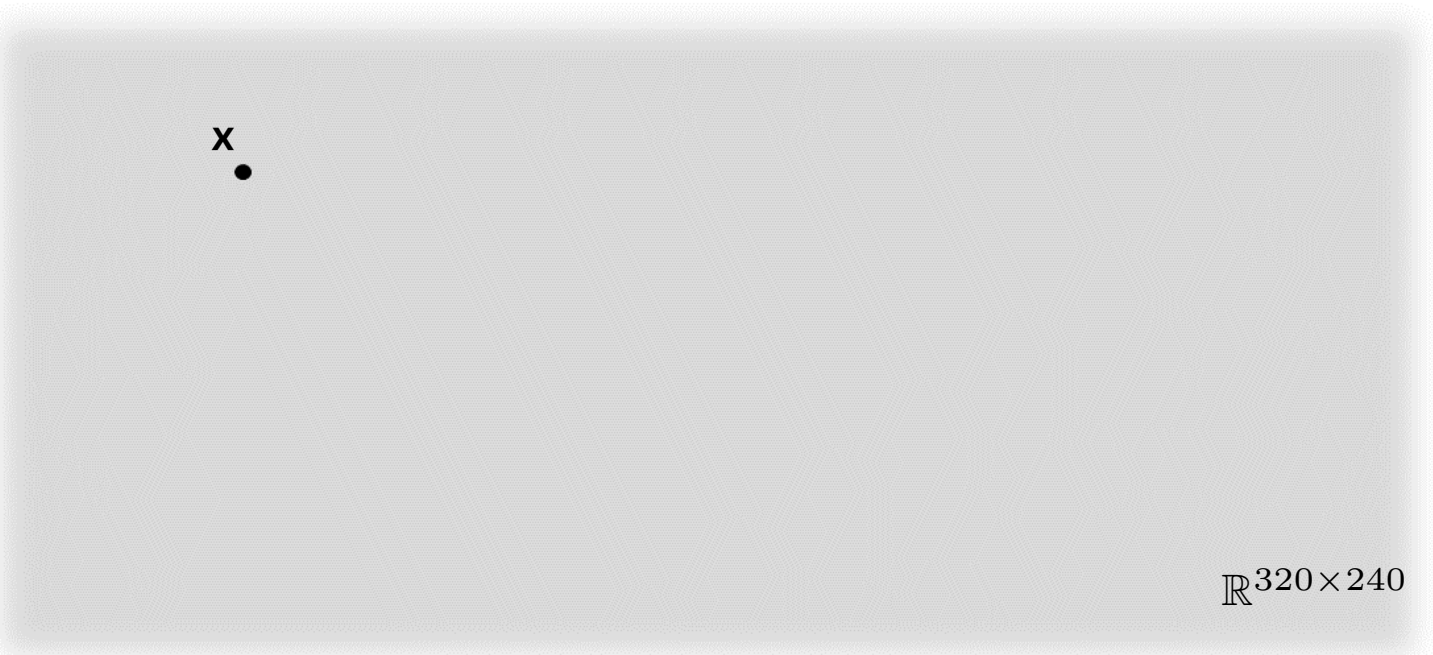
Generative models

What does it mean to learn a distribution?



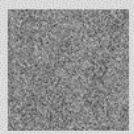
Generative models

What does it mean to learn a distribution?



Generative models

What does it mean to learn a distribution?



$\mathbb{R}^{320 \times 240}$

Generative models

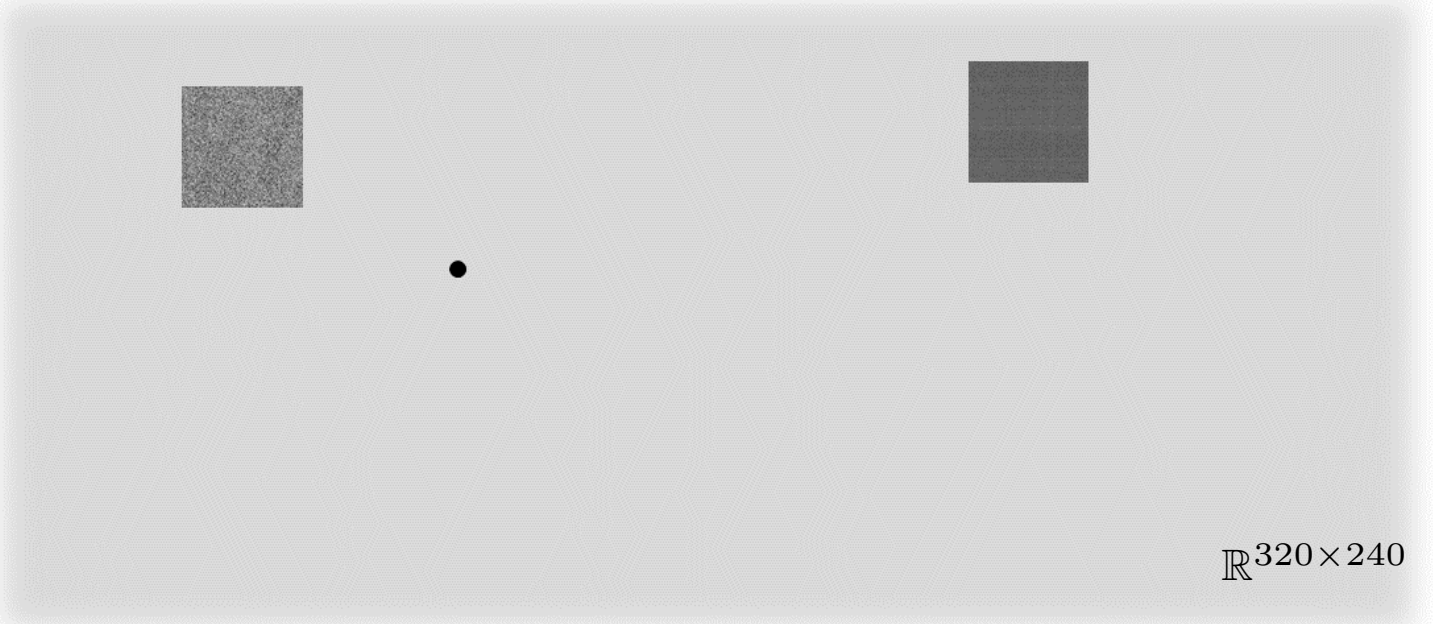
What does it mean to learn a distribution?

Generative models

What does it mean to learn a distribution?

Generative models

What does it mean to learn a distribution?

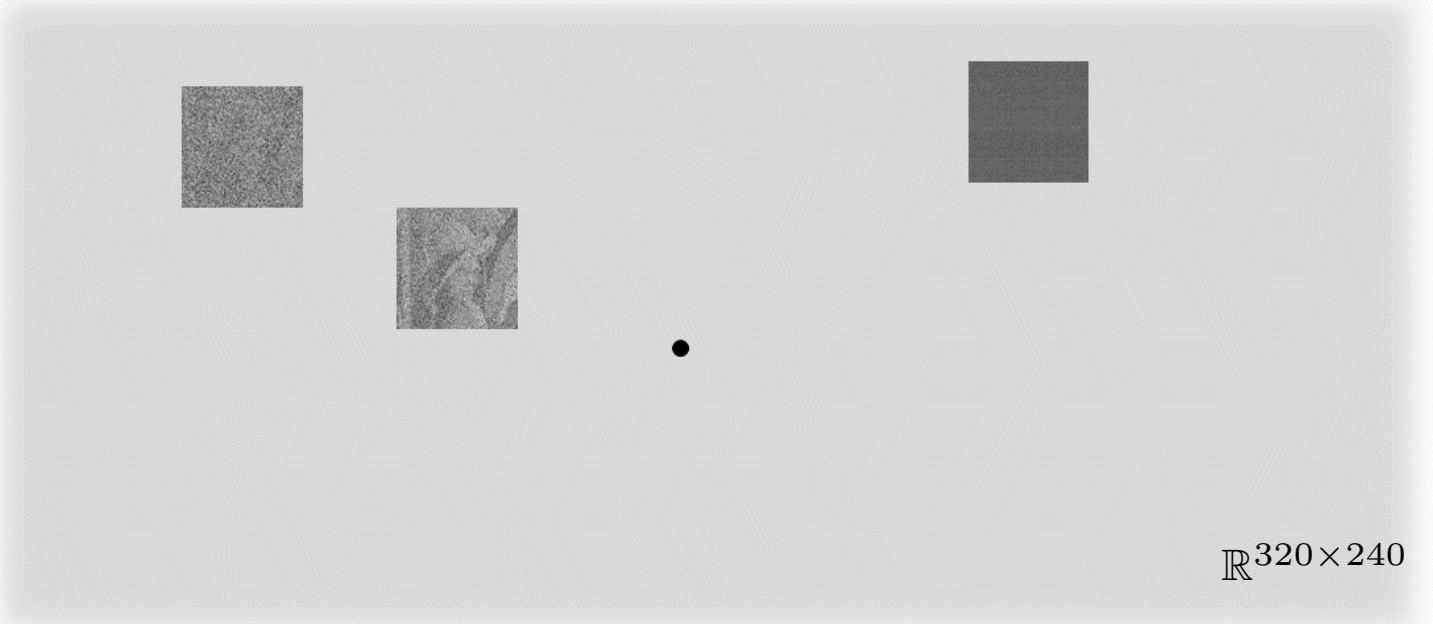


Generative models

What does it mean to learn a distribution?

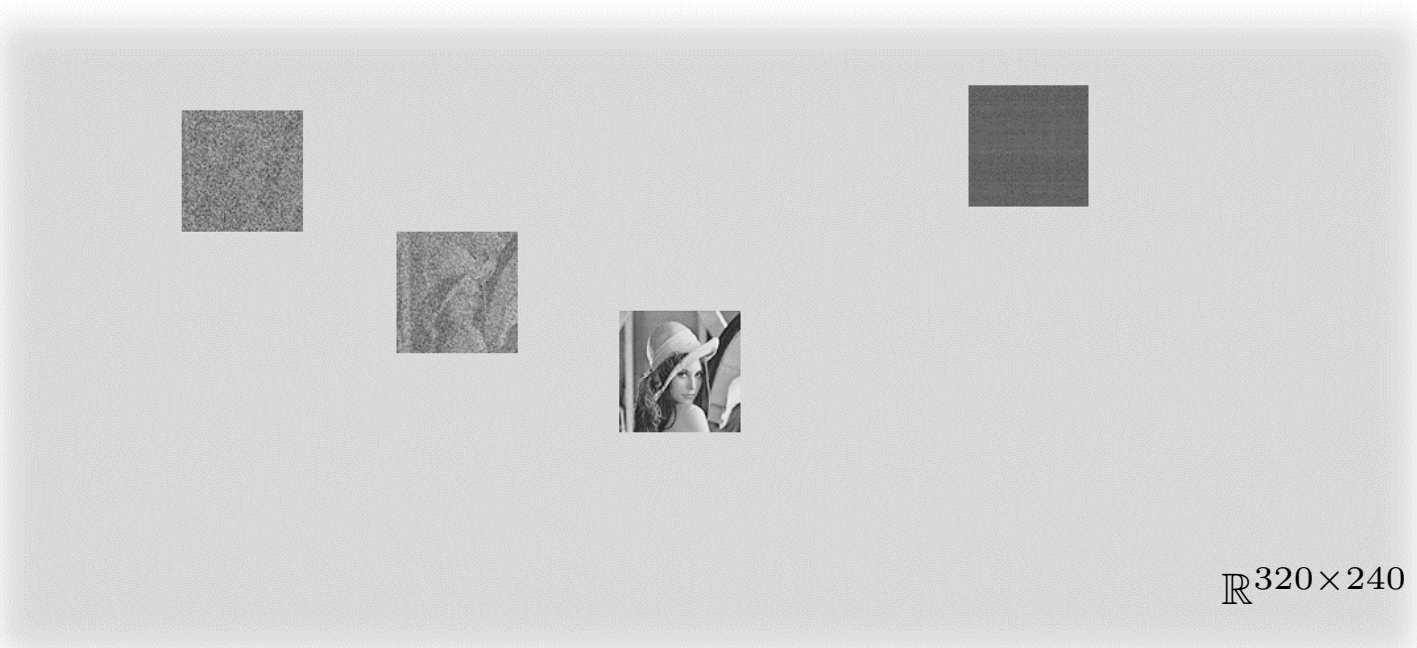
Generative models

What does it mean to learn a distribution?



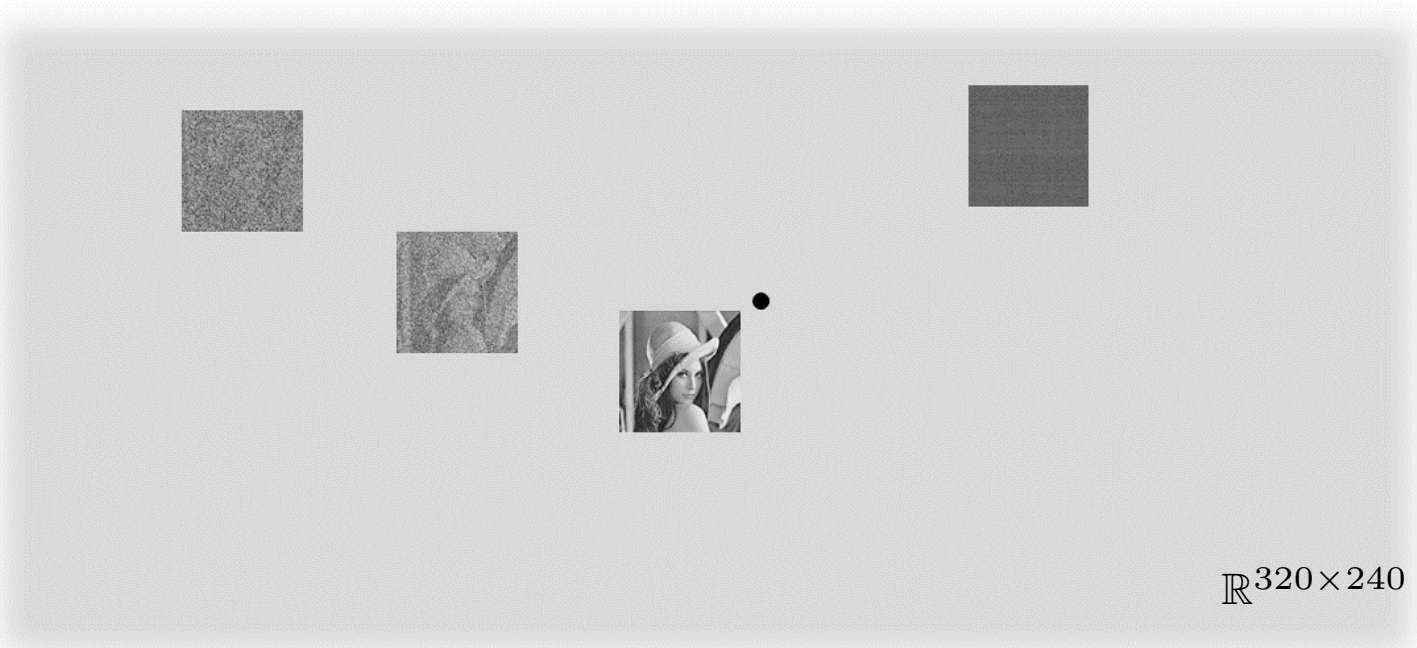
Generative models

What does it mean to learn a distribution?



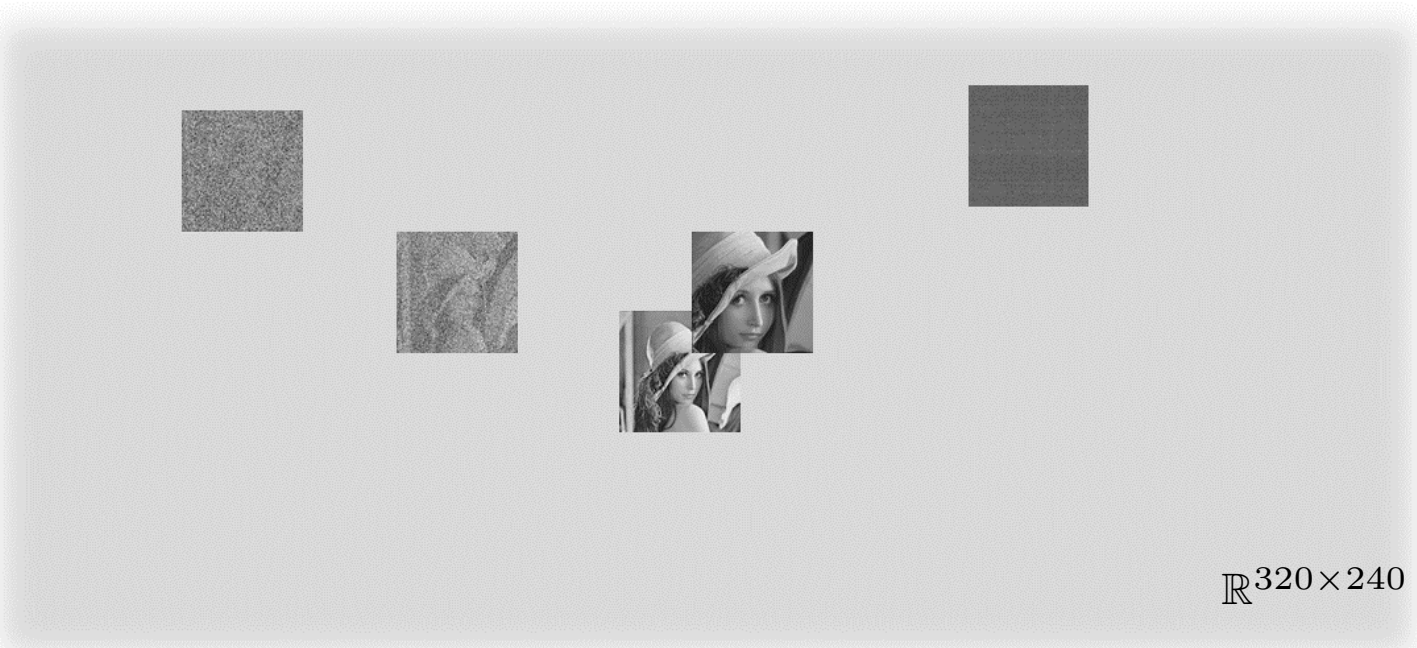
Generative models

What does it mean to learn a distribution?



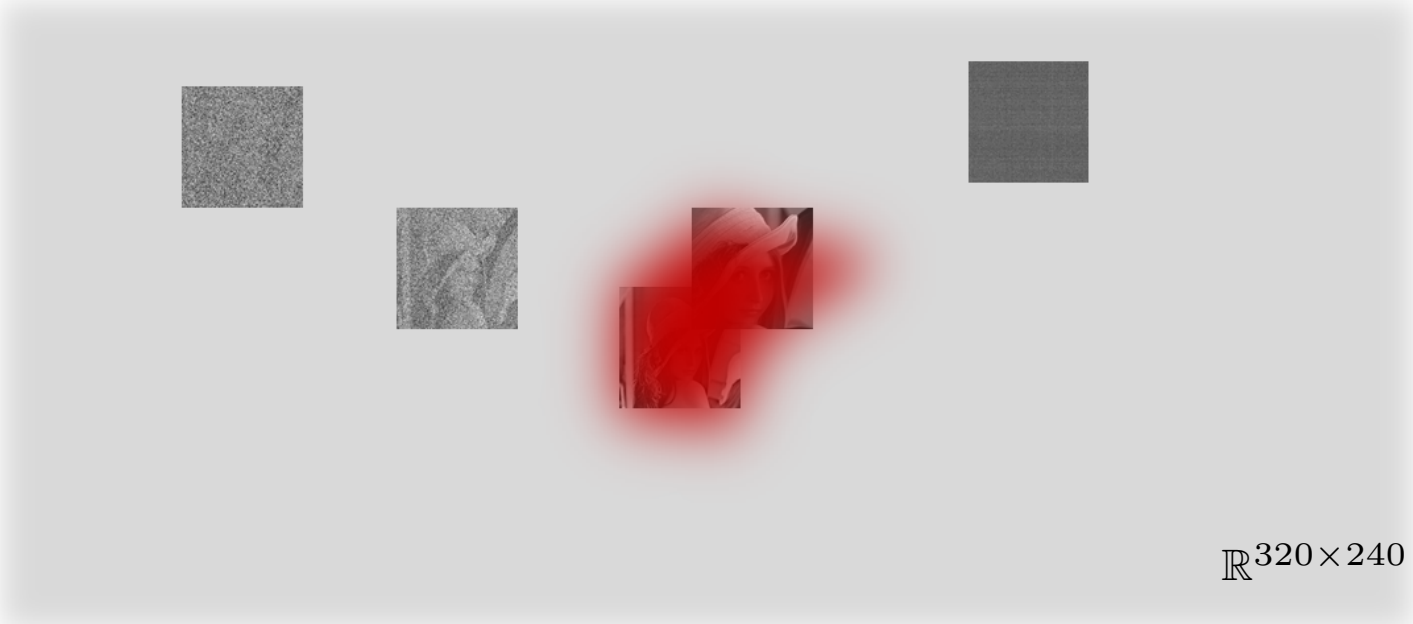
Generative models

What does it mean to learn a distribution?



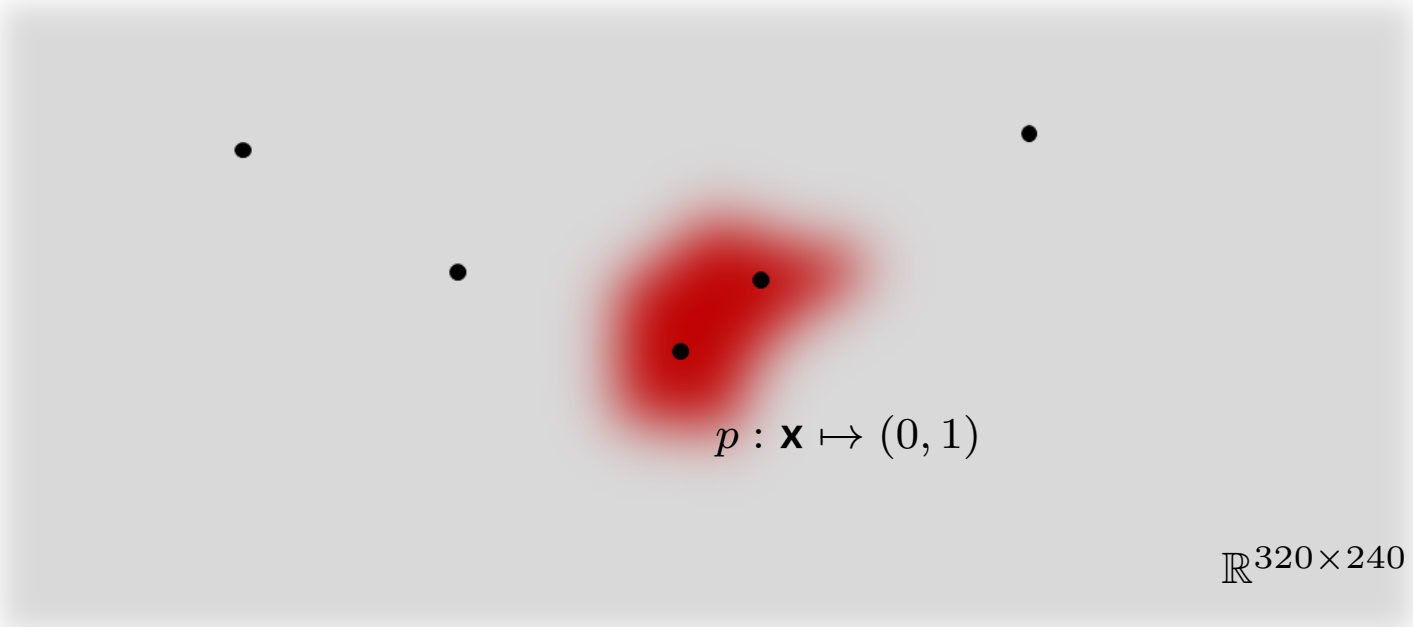
Generative models

What does it mean to learn a distribution?



Generative models

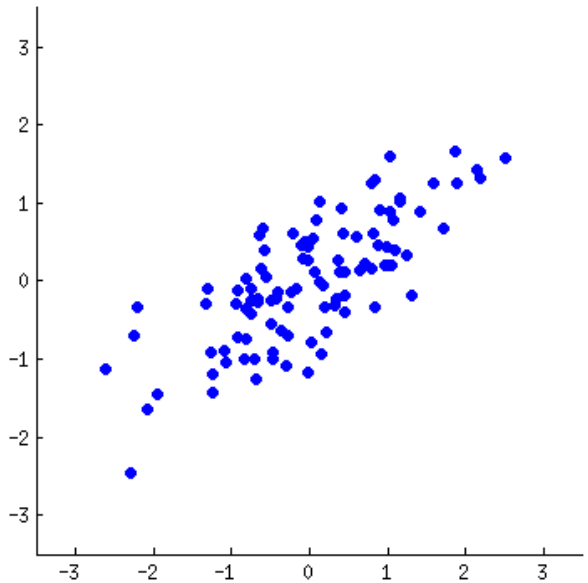
What does it mean to learn a distribution?



Probability distribution p defined on the entire data space.

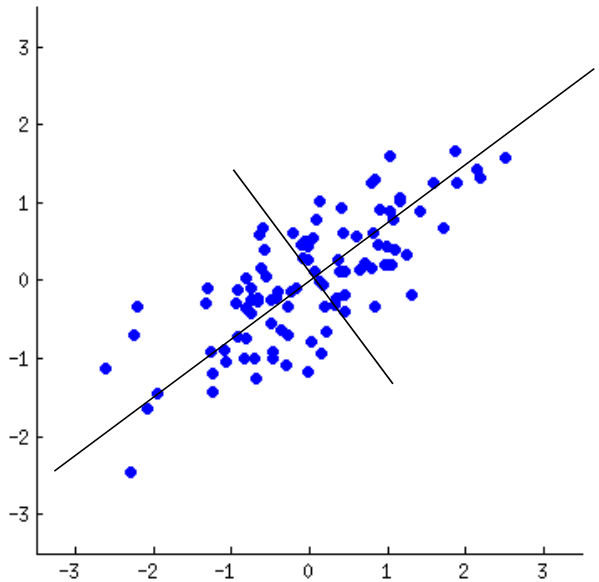
Principal component analysis (PCA)

Regard our data as n points in $\mathbb{R}^d$:



Principal component analysis (PCA)

Regard our data as n points in $\mathbb{R}^d$:

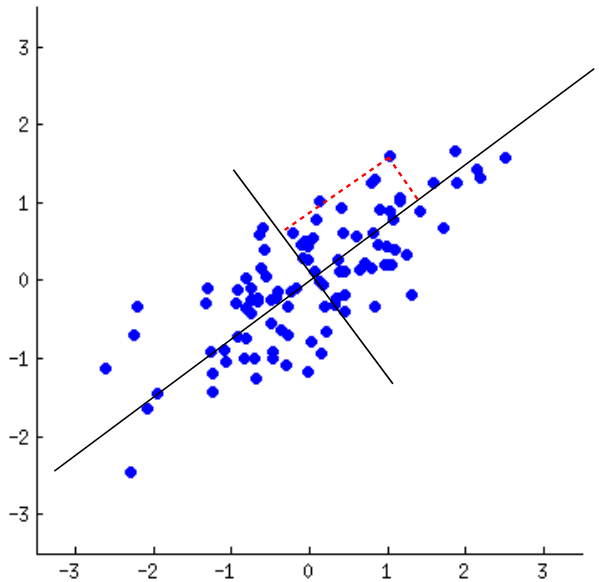


Overall idea:

- Find $k \leq d$ orthogonal directions with the most variance.

Principal component analysis (PCA)

Regard our data as n points in $\mathbb{R}^d$:



Overall idea:

- Find $k \leq d$ orthogonal directions with the most variance.
- Project all the data points onto these directions.

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d}$$

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k}$$

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} — & \mathbf{z}_1^\top & — \\ & \vdots & \\ — & \mathbf{z}_n^\top & — \end{pmatrix}}_{n \times k}$$

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} — & \mathbf{z}_1^\top & — \\ & \vdots & \\ — & \mathbf{z}_n^\top & — \end{pmatrix}}_{n \times k}$$

Assuming $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$, for $k = d$ we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top$$
$$\mathbf{X} = \mathbf{W}\mathbf{Z}$$

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} — & \mathbf{z}_1^\top & — \\ & \vdots & \\ — & \mathbf{z}_n^\top & — \end{pmatrix}}_{n \times k}$$

Assuming $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$, for $k < d$ we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top$$
$$\mathbf{X} \approx \mathbf{W}\mathbf{Z}$$

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} — & \mathbf{z}_1^\top & — \\ & \vdots & \\ — & \mathbf{z}_n^\top & — \end{pmatrix}}_{n \times k}$$

Assuming $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$, for $k < d$ we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$
$$\mathbf{X} \approx \mathbf{W}\mathbf{Z} \quad \text{reconstruction}$$

Principal component analysis (PCA)

In matrix notation:

$$\underbrace{\begin{pmatrix} — & \mathbf{x}_1^\top & — \\ & \vdots & \\ — & \mathbf{x}_n^\top & — \end{pmatrix}}_{n \times d} \underbrace{\begin{pmatrix} | & & | \\ \mathbf{w}_1 & \cdots & \mathbf{w}_k \\ | & & | \end{pmatrix}}_{d \times k} = \underbrace{\begin{pmatrix} — & \mathbf{z}_1^\top & — \\ & \vdots & \\ — & \mathbf{z}_n^\top & — \end{pmatrix}}_{n \times k}$$

Assuming $\mathbf{W}^\top \mathbf{W} = \mathbf{I}$, for $k < d$ we get:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$
$$\mathbf{X} \approx \mathbf{W}\mathbf{Z} \quad \text{reconstruction}$$

We call the columns of $\mathbf{W}$ principal components.

They are unknown and must be computed.

Principal component analysis (PCA)

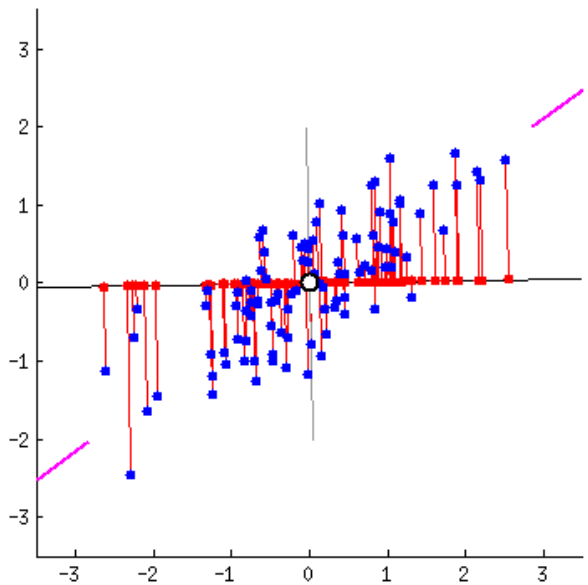
We seek the direction $\mathbf{w}$ (a column of $\mathbf{W}$) that:

- Minimizes the projection/reconstruction error.
- Maximizes the variance of the projected data.

Principal component analysis (PCA)

We seek the direction $\mathbf{w}$ (a column of $\mathbf{W}$) that:

- Minimizes the projection/reconstruction error.
- Maximizes the variance of the projected data.



Principal component analysis (PCA)

Assume the data points $\mathbf{X}$ are centered at zero.

For a given $\mathbf{w}$, the projection of all n points onto $\mathbf{w}$ is $\mathbf{X}^\top \mathbf{w}$.

Principal component analysis (PCA)

Assume the data points $\mathbf{X}$ are centered at zero.

For a given $\mathbf{w}$, the projection of all n points onto $\mathbf{w}$ is $\mathbf{X}^\top \mathbf{w}$.

The variance to maximize is $\|\mathbf{X}^\top \mathbf{w}\|_2^2$:

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w})$$

Principal component analysis (PCA)

Assume the data points $\mathbf{X}$ are centered at zero.

For a given $\mathbf{w}$, the projection of all n points onto $\mathbf{w}$ is $\mathbf{X}^\top \mathbf{w}$.

The variance to maximize is $\|\mathbf{X}^\top \mathbf{w}\|_2^2$:

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top (\mathbf{X}\mathbf{X}^\top) \mathbf{w}$$

Principal component analysis (PCA)

Assume the data points $\mathbf{X}$ are centered at zero.

For a given $\mathbf{w}$, the projection of all n points onto $\mathbf{w}$ is $\mathbf{X}^\top \mathbf{w}$.

The variance to maximize is $\|\mathbf{X}^\top \mathbf{w}\|_2^2$:

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top \underbrace{(\mathbf{X}\mathbf{X}^\top)}_{\mathbf{C}} \mathbf{w}$$

where $\mathbf{C} \in \mathbb{R}^{d \times d}$ is the symmetric covariance matrix.

Principal component analysis (PCA)

Assume the data points $\mathbf{X}$ are centered at zero.

For a given $\mathbf{w}$, the projection of all n points onto $\mathbf{w}$ is $\mathbf{X}^\top \mathbf{w}$.

The variance to maximize is $\|\mathbf{X}^\top \mathbf{w}\|_2^2$:

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top \underbrace{(\mathbf{X}\mathbf{X}^\top)}_{\mathbf{C}} \mathbf{w}$$

where $\mathbf{C} \in \mathbb{R}^{d \times d}$ is the symmetric covariance matrix.

We want to solve the problem:

$$\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{C} \mathbf{w} \quad \text{s.t. } \|\mathbf{w}\|_2 = 1$$

Principal component analysis (PCA)

Assume the data points $\mathbf{X}$ are centered at zero.

For a given $\mathbf{w}$, the projection of all n points onto $\mathbf{w}$ is $\mathbf{X}^\top \mathbf{w}$.

The variance to maximize is $\|\mathbf{X}^\top \mathbf{w}\|_2^2$:

$$(\mathbf{X}^\top \mathbf{w})^\top (\mathbf{X}^\top \mathbf{w}) = \mathbf{w}^\top \underbrace{(\mathbf{X}\mathbf{X}^\top)}_{\mathbf{C}} \mathbf{w}$$

where $\mathbf{C} \in \mathbb{R}^{d \times d}$ is the symmetric covariance matrix.

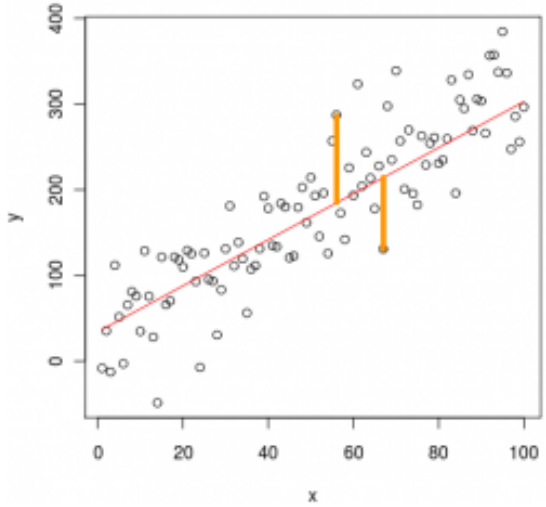
We want to solve the problem:

$$\max_{\mathbf{w}} \mathbf{w}^\top \mathbf{C} \mathbf{w} \quad \text{s.t. } \|\mathbf{w}\|_2 = 1$$

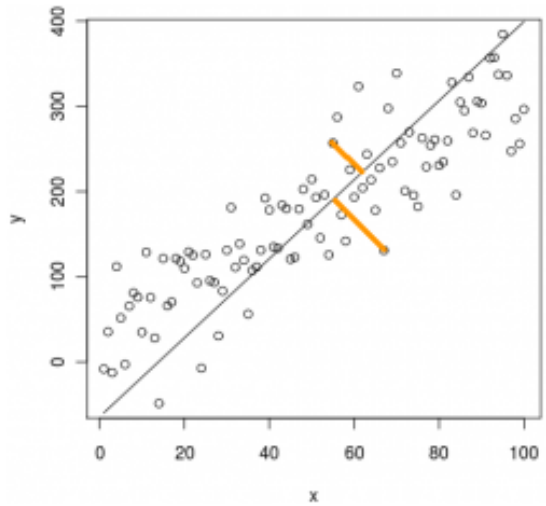
Solution: $\mathbf{w} =$ principal eigenvector of $\mathbf{C}$

PCA is not linear regression

Linear regression: measure along the y coordinate:



PCA: measure orthogonal to the principal direction:



PCA as a generative model

Given the $\mathbf{W}$ satisfying:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$
$$\mathbf{X} \approx \mathbf{W}\mathbf{Z} \quad \text{reconstruction}$$

Generate new data just by sampling $\mathbf{z}_{\text{new}} \in \mathbb{R}^k$:

$$\mathbf{x}_{\text{new}} = \mathbf{W}\mathbf{z}_{\text{new}}$$

PCA as a generative model

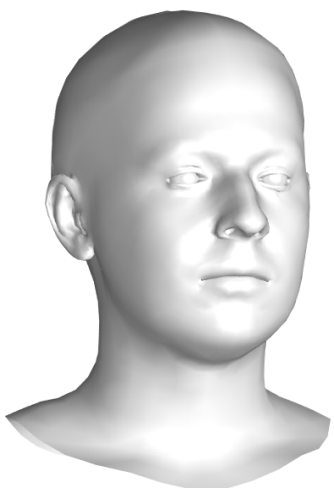
Given the $\mathbf{W}$ satisfying:

$$\mathbf{X}^\top \mathbf{W} = \mathbf{Z}^\top \quad \text{projection}$$
$$\mathbf{X} \approx \mathbf{W}\mathbf{Z} \quad \text{reconstruction}$$

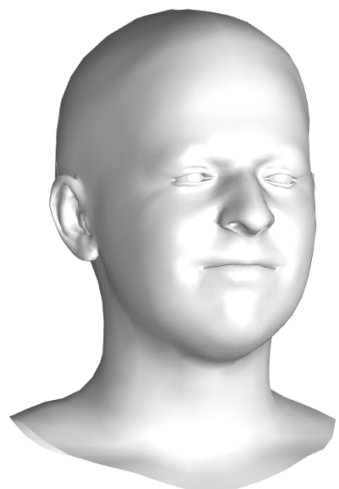
Generate new data just by sampling $\mathbf{z}_{\text{new}} \in \mathbb{R}^k$:

$$\mathbf{x}_{\text{new}} = \mathbf{W}\mathbf{z}_{\text{new}}$$

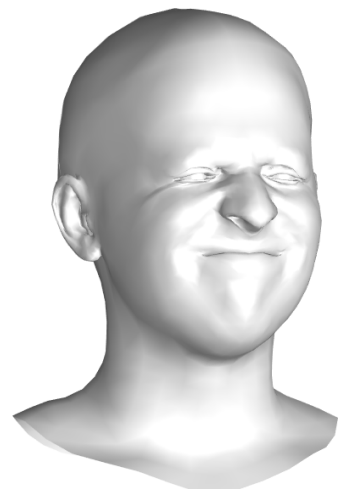
Example:



Data point $\mathbf{x}_1$



Generated
$\mathbf{x}_{\text{new}} = \frac{1}{2}\mathbf{W}(\mathbf{z}_1 + \mathbf{z}_2)$



Data point $\mathbf{x}_2$

Codes

Consider again the relations:

$$\mathbf{W}^\top \mathbf{x} = \mathbf{z} \qquad \text{projection}$$
$$\mathbf{x} \approx \mathbf{W}\mathbf{z} \qquad \text{reconstruction}$$

New perspective: PCA as a parametric model.

Codes

Consider again the relations:

$$\mathbf{W}^\top \mathbf{x} = \mathbf{z} \qquad \text{encoding}$$
$$\mathbf{x} \approx \mathbf{W}\mathbf{z} \quad \text{decoding}$$

New perspective: PCA as a parametric model.

Each data point $\mathbf{x}$ is transformed into a low-dimensional code $\mathbf{z} \in \mathbb{R}^k$, where the dimension $k < d$ is fixed.

Encoding and decoding are linear.

Codes

Consider again the relations:

$$\mathbf{W}^\top \mathbf{x} = \mathbf{z} \qquad \text{encoding}$$
$$\mathbf{x} \approx \mathbf{W}\mathbf{z} \quad \text{decoding}$$

New perspective: PCA as a parametric model.

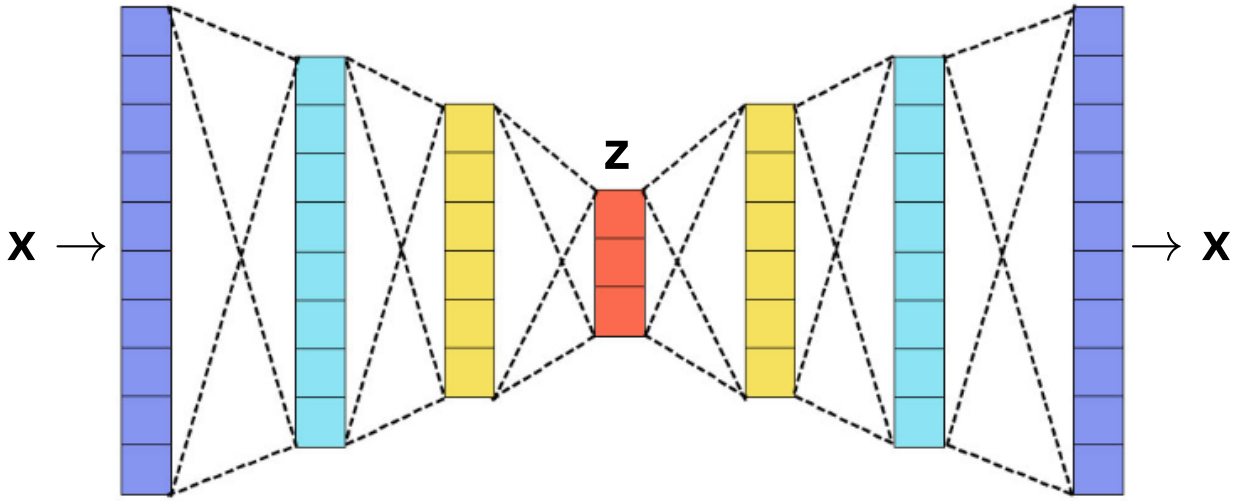
Each data point $\mathbf{x}$ is transformed into a low-dimensional code $\mathbf{z} \in \mathbb{R}^k$, where the dimension $k < d$ is fixed.

Encoding and decoding are linear.

How to generalize this idea?

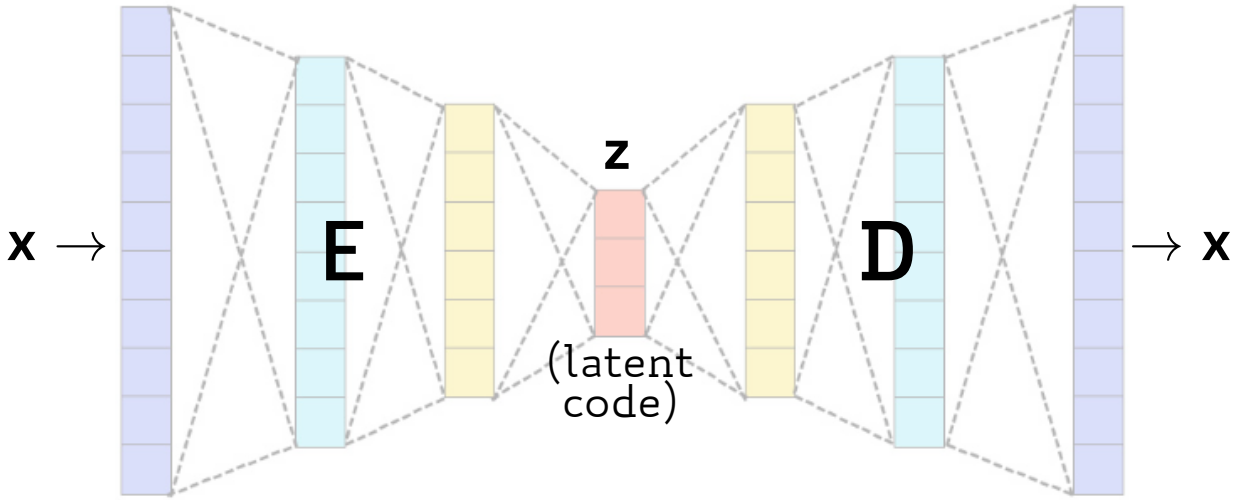
Autoencoders (AE)

Implement encoding and decoding via deep nets.



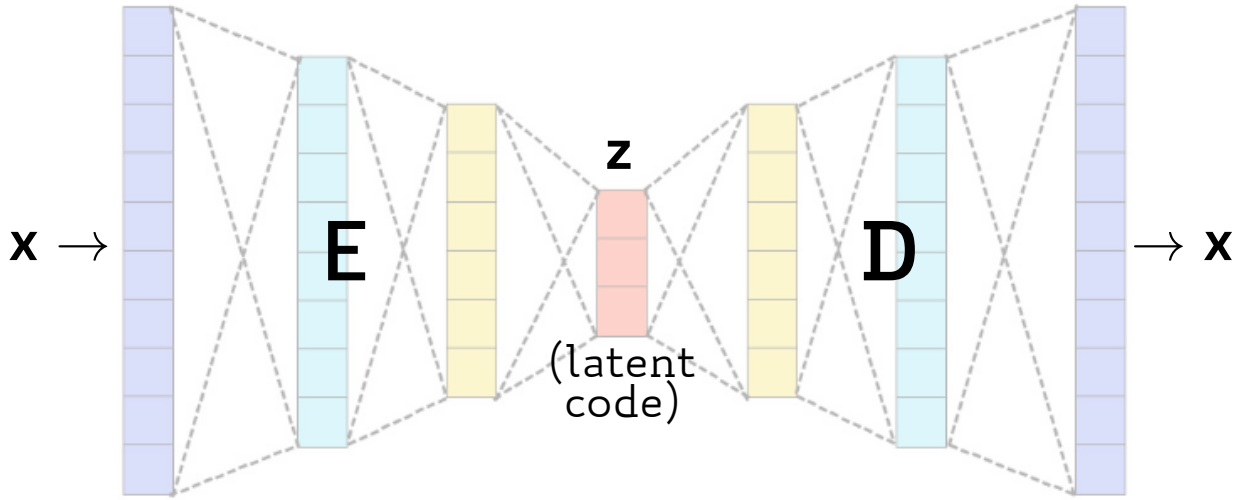
Autoencoders (AE)

Implement encoding and decoding via deep nets.



Autoencoders (AE)

Implement encoding and decoding via deep nets.

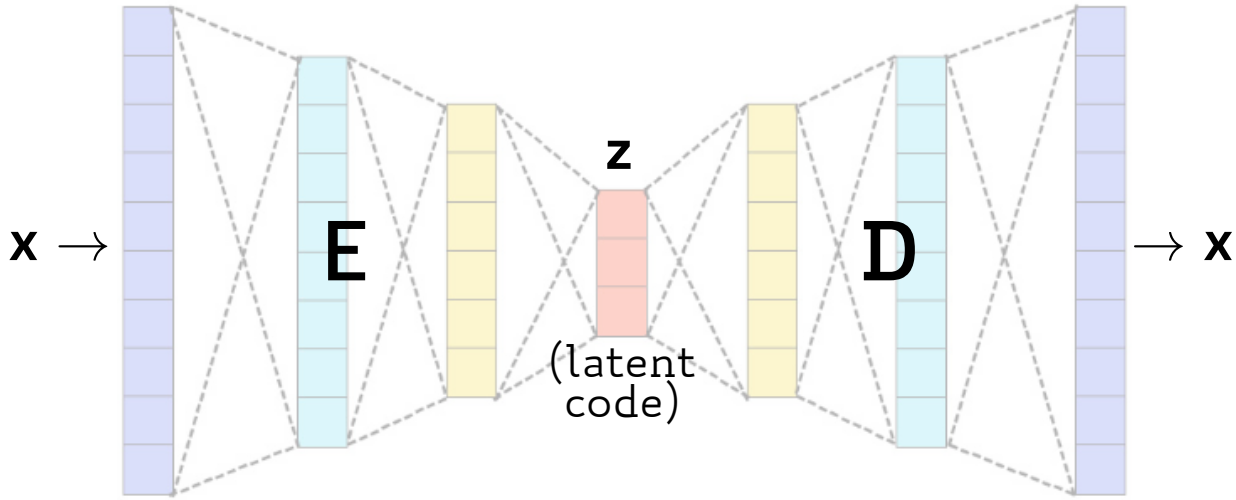


Given a dataset $\{\mathbf{x}_i\}$, require encoder E and decoder (or generator) D to minimize the reconstruction loss:

$$\ell_\Theta = \sum_i \|\mathbf{x}_i - D_\Theta(E_\Theta(\mathbf{x}_i))\|$$

Autoencoders (AE)

Implement encoding and decoding via deep nets.



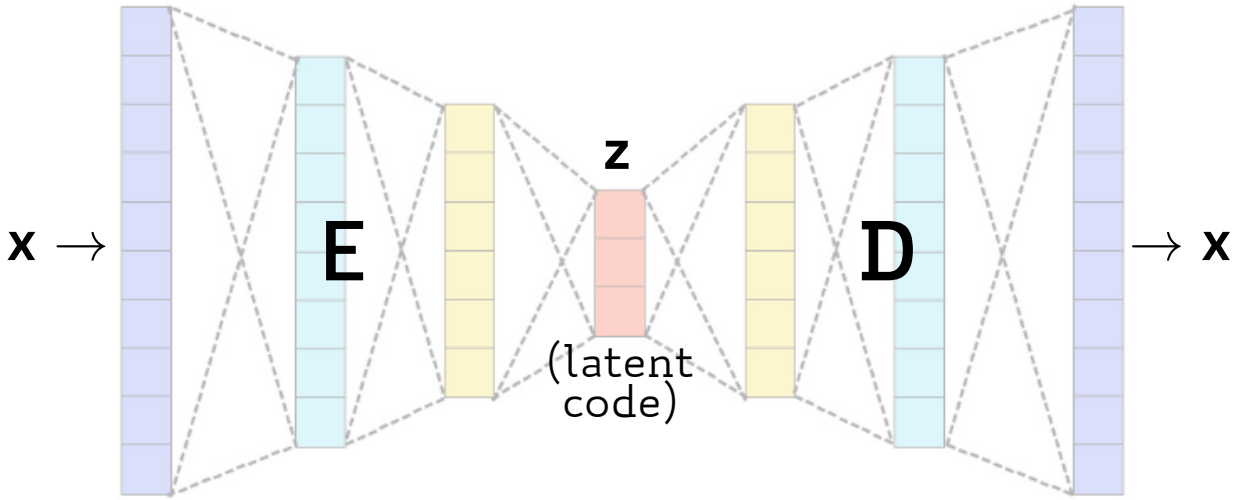
Given a dataset $\{\mathbf{x}_i\}$, require encoder E and decoder (or generator) D to minimize the reconstruction loss:

$$\ell_\Theta = \sum_i \|\mathbf{x}_i - D_\Theta(E_\Theta(\mathbf{x}_i))\|$$

The choice of the metric depends on the data.

Autoencoders (AE)

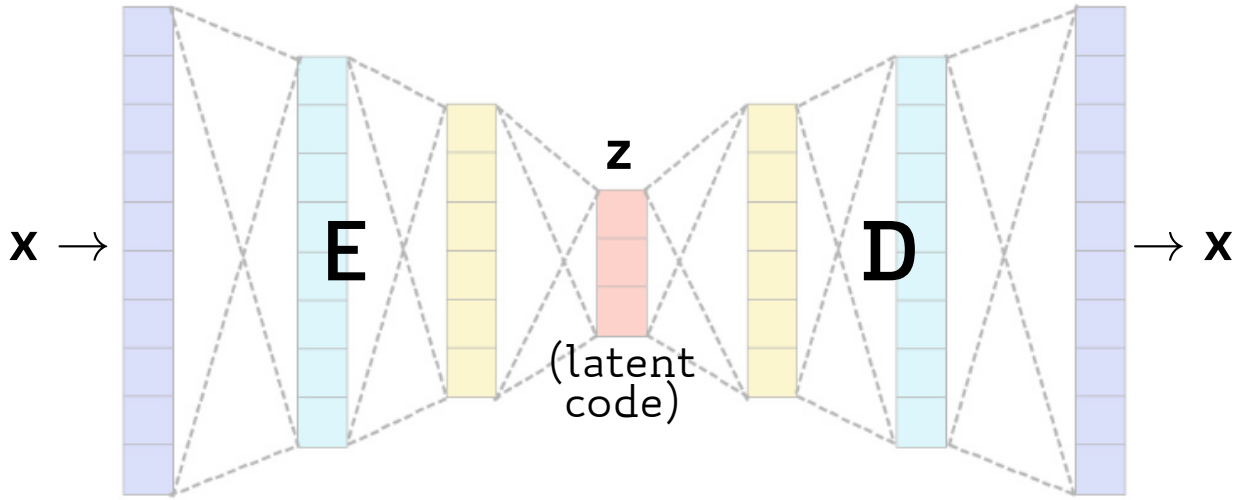
Implement encoding and decoding via deep nets.



- The bottleneck prevents trivial solutions.

Autoencoders (AE)

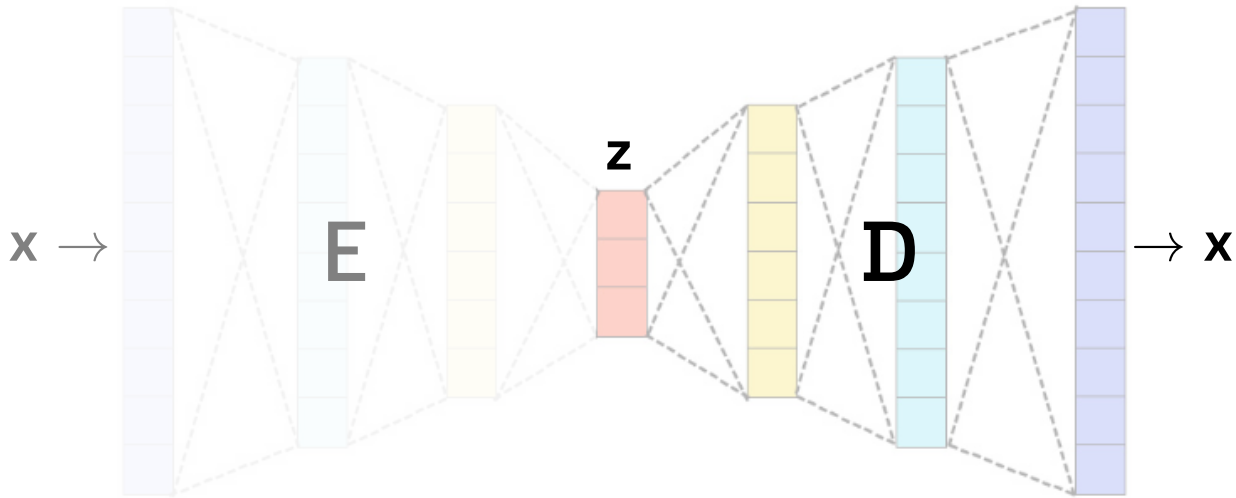
Implement encoding and decoding via deep nets.



- The bottleneck prevents trivial solutions.
- Task: reconstruction.

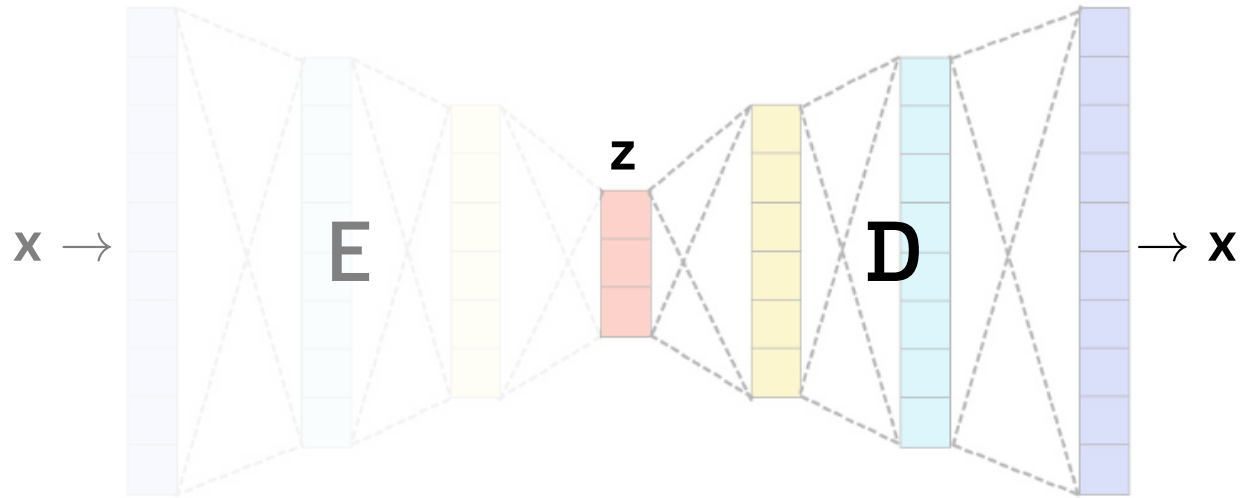
  Once the AE is trained, study the structure of the latent space and use E, D for new tasks.

Manifold hypothesis



The decoder maps from a low-dimensional latent space to a high-dimensional embedding of observed data.

Manifold hypothesis



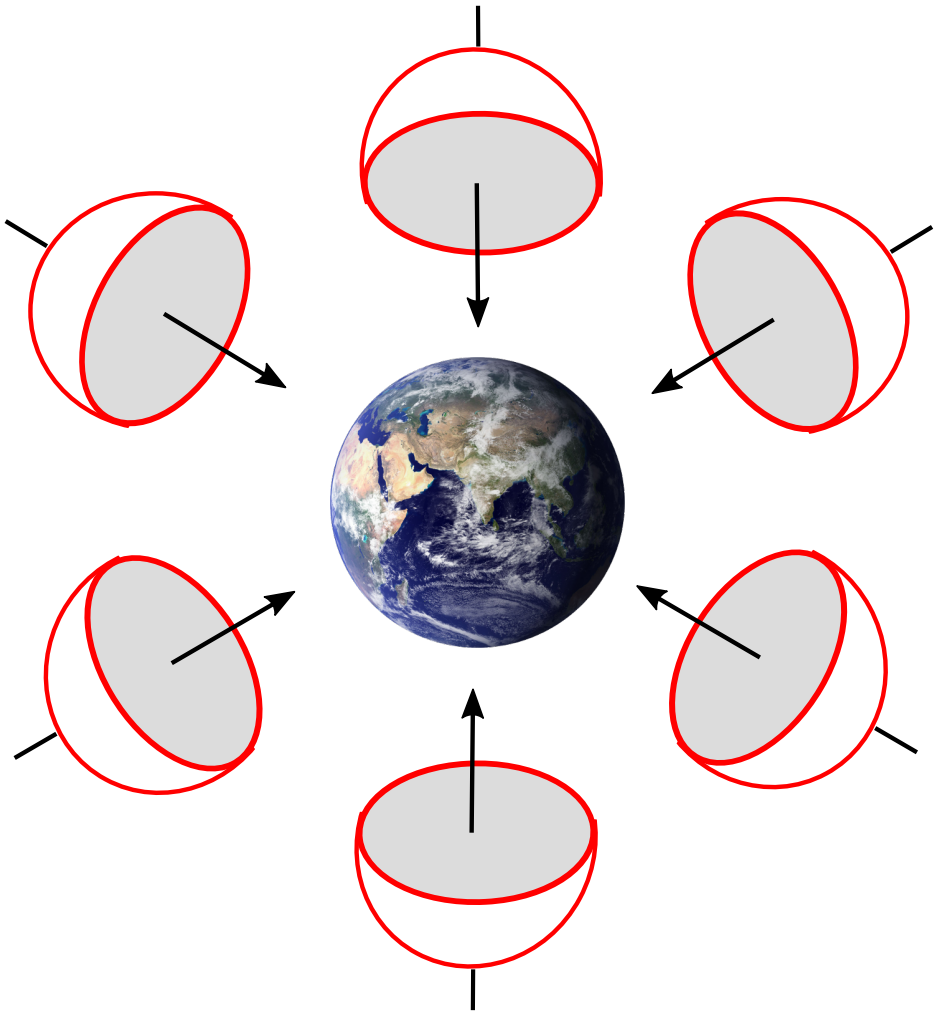
The decoder maps from a low-dimensional latent space to a high-dimensional embedding of observed data.

The latent space is Euclidean.

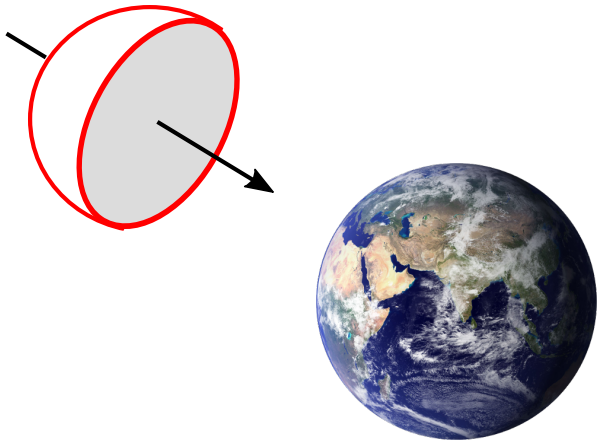
The data embedding space is curved (manifold hypothesis).

Manifolds

Manifolds are unions of charts:



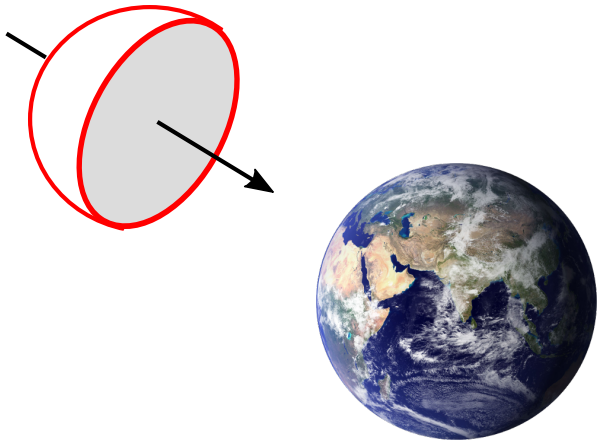
2D manifolds (surfaces)



chart

Each chart can be seen as a mapping $\phi : \mathbb{R}^2 \to \mathcal{S} \subset \mathbb{R}^3$.

2D manifolds (surfaces)

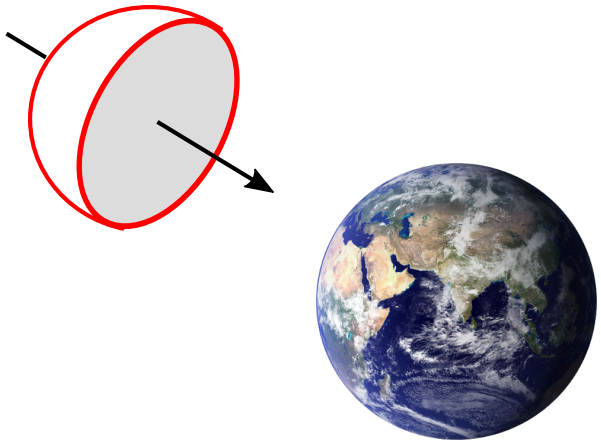


chart

Each chart can be seen as a mapping $\phi : \mathbb{R}^2 \to \mathcal{S} \subset \mathbb{R}^3$.

ϕ must be smooth and invertible (diffeomorphism).

2D manifolds (surfaces)



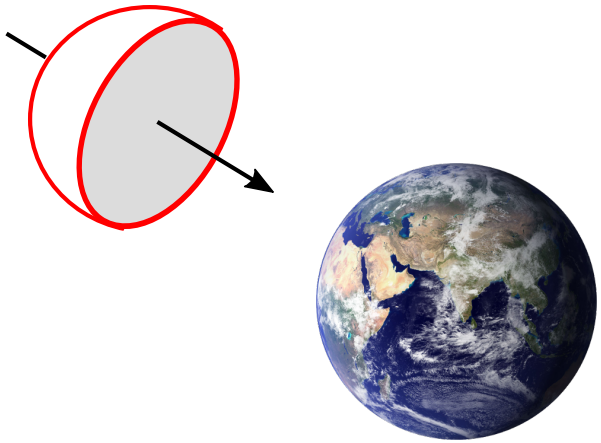
chart

Each chart can be seen as a mapping $\phi : \mathbb{R}^2 \to \mathcal{S} \subset \mathbb{R}^3$.

ϕ must be smooth and invertible (diffeomorphism).

- The domain of ϕ is the parametric space and is Euclidean.

2D manifolds (surfaces)



chart

Each chart can be seen as a mapping $\phi : \mathbb{R}^2 \to \mathcal{S} \subset \mathbb{R}^3$.

ϕ must be smooth and invertible (diffeomorphism).

- The domain of ϕ is the parametric space and is Euclidean.
- The image of ϕ is the embedding and is a surface.

Manifolds

Manifolds can be k-dimensional:

$$\phi : \mathbb{R}^k \to \mathcal{M} \subset \mathbb{R}^d \qquad \text{with } k < d$$

Manifolds

Manifolds can be k-dimensional:

$$\phi : \mathbb{R}^k \to \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

The parametrization is not unique:

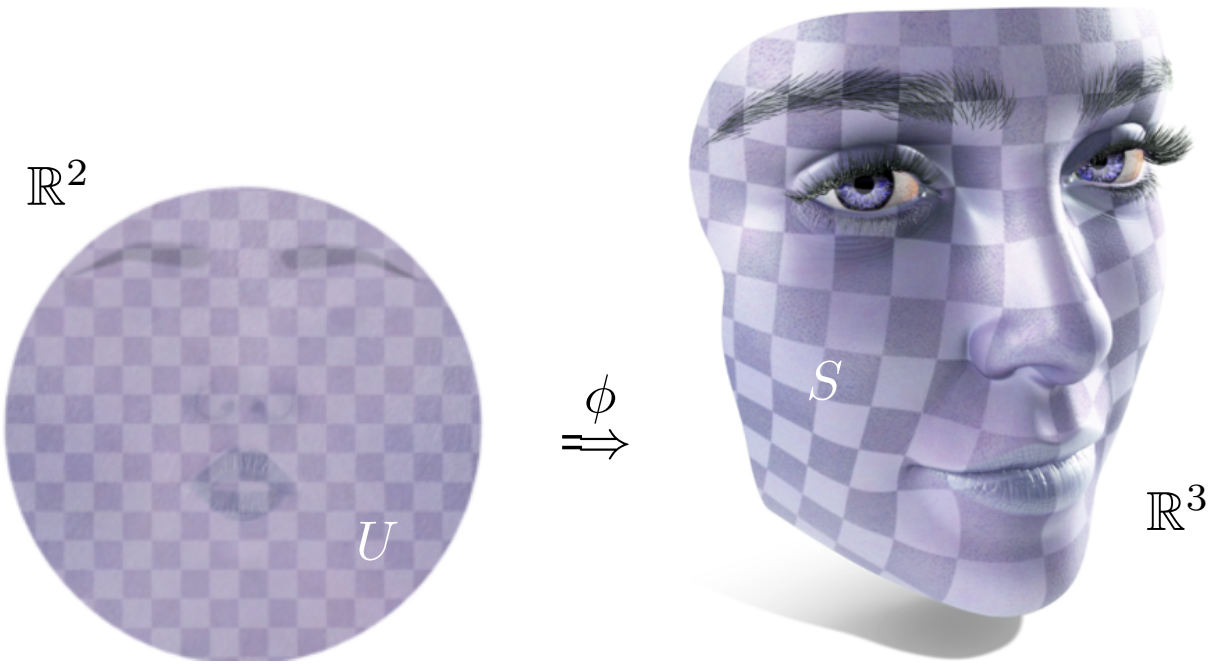


Figure by Keenan Crane

Manifolds

Manifolds can be k-dimensional:

$$\phi : \mathbb{R}^k \to \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

The parametrization is not unique:

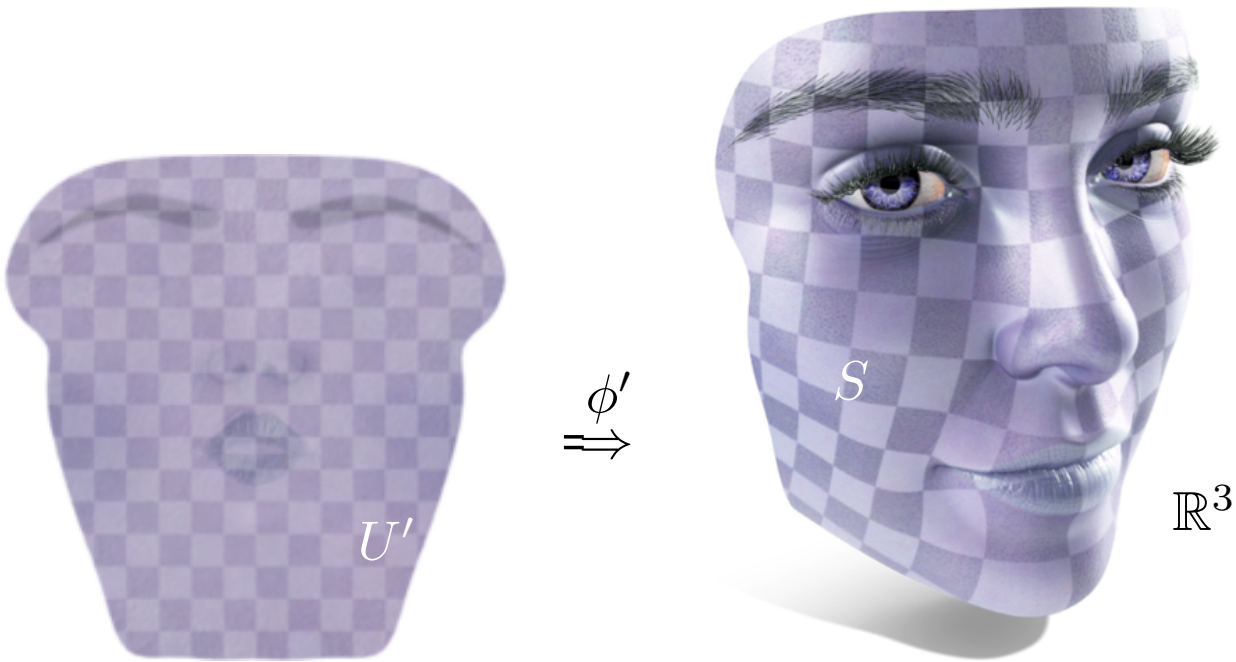


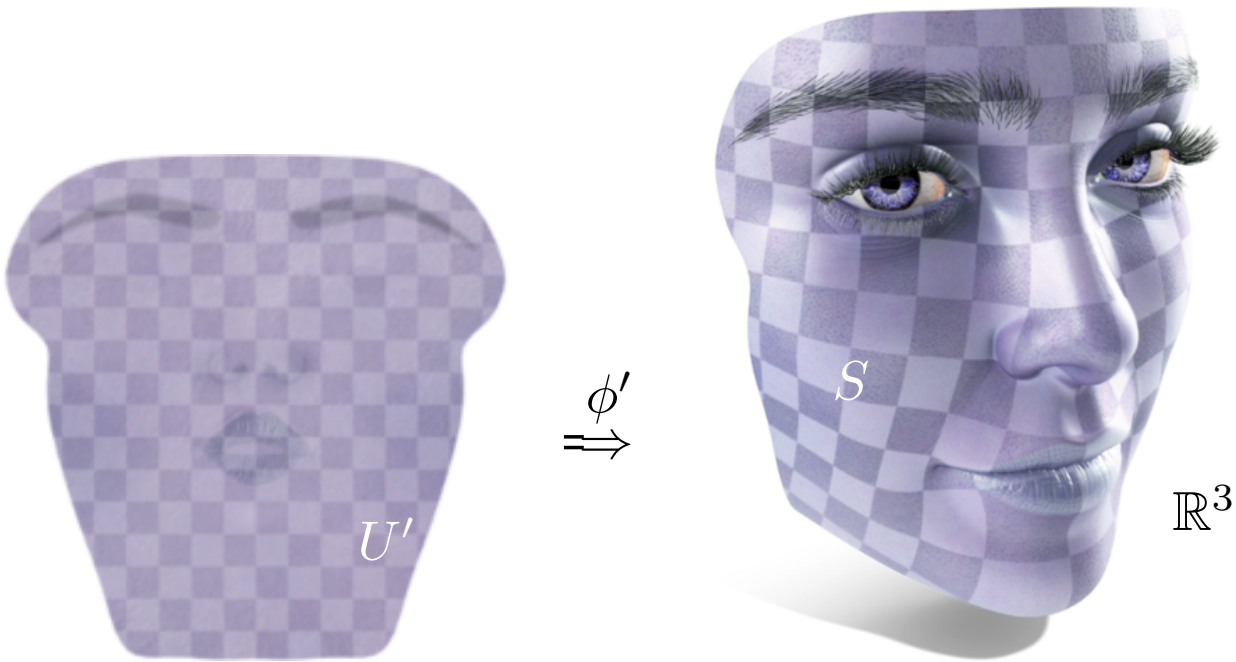
Figure by Keenan Crane

Manifolds

Manifolds can be k-dimensional:

$$\phi : \mathbb{R}^k \to \mathcal{M} \subset \mathbb{R}^d \quad \text{with } k < d$$

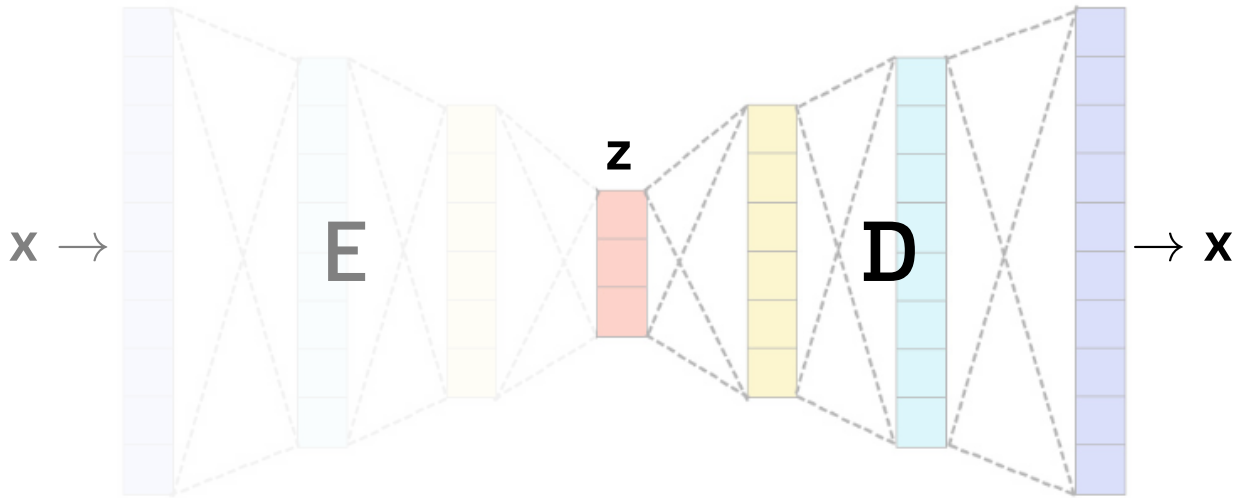
The parametrization is not unique:



They all encode the same geometric information.

Figure by Keenan Crane

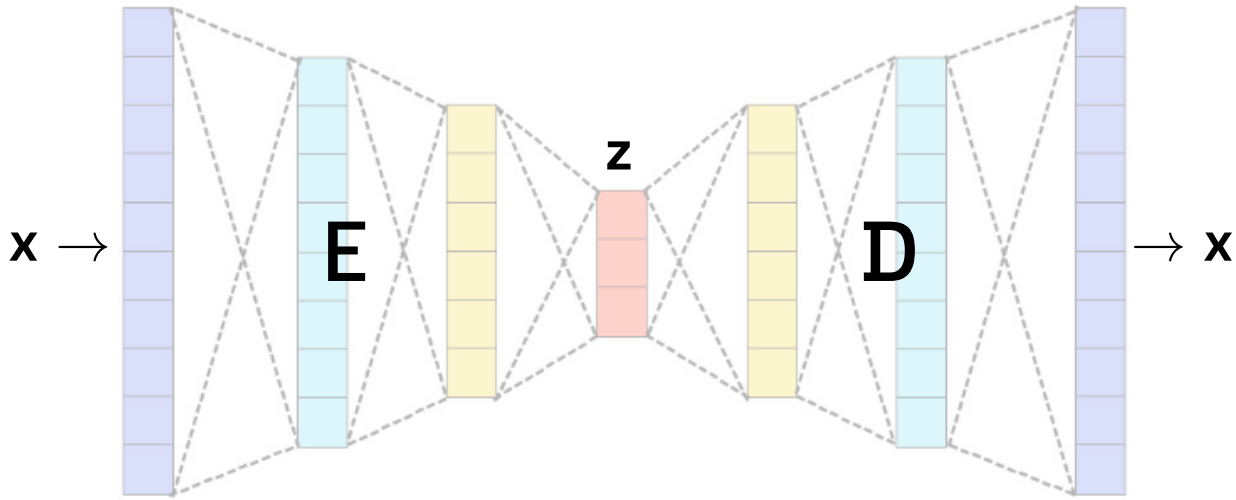
Manifolds and generative models



The decoder $D : \mathbb{R}^k \to \mathbb{R}^d$ is a chart from the latent space spanned by the codes **z** to the data space of the inputs **x**.

Shao et al, "The Riemannian geometry of deep generative models", 2018

Manifolds and generative models



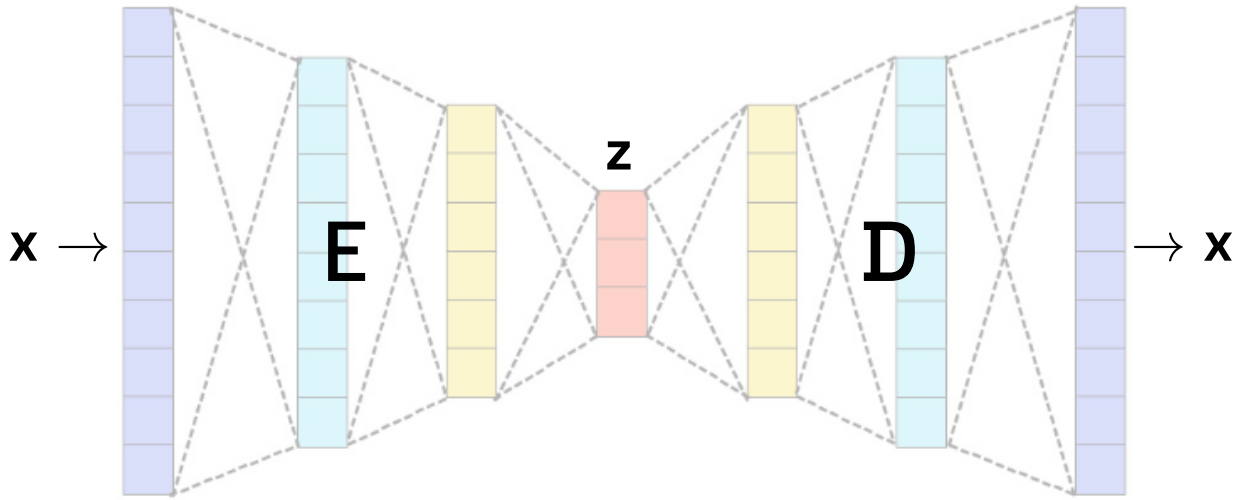
The decoder $D : \mathbb{R}^k \to \mathbb{R}^d$ is a chart from the latent space spanned by the codes **z** to the data space of the inputs **x**.

It is differentiable.

It is invertible via the encoder E.

Shao et al, "The Riemannian geometry of deep generative models", 2018

Manifolds and generative models



The decoder $D : \mathbb{R}^k \to \mathbb{R}^d$ is a chart from the latent space spanned by the codes **z** to the data space of the inputs **x**.

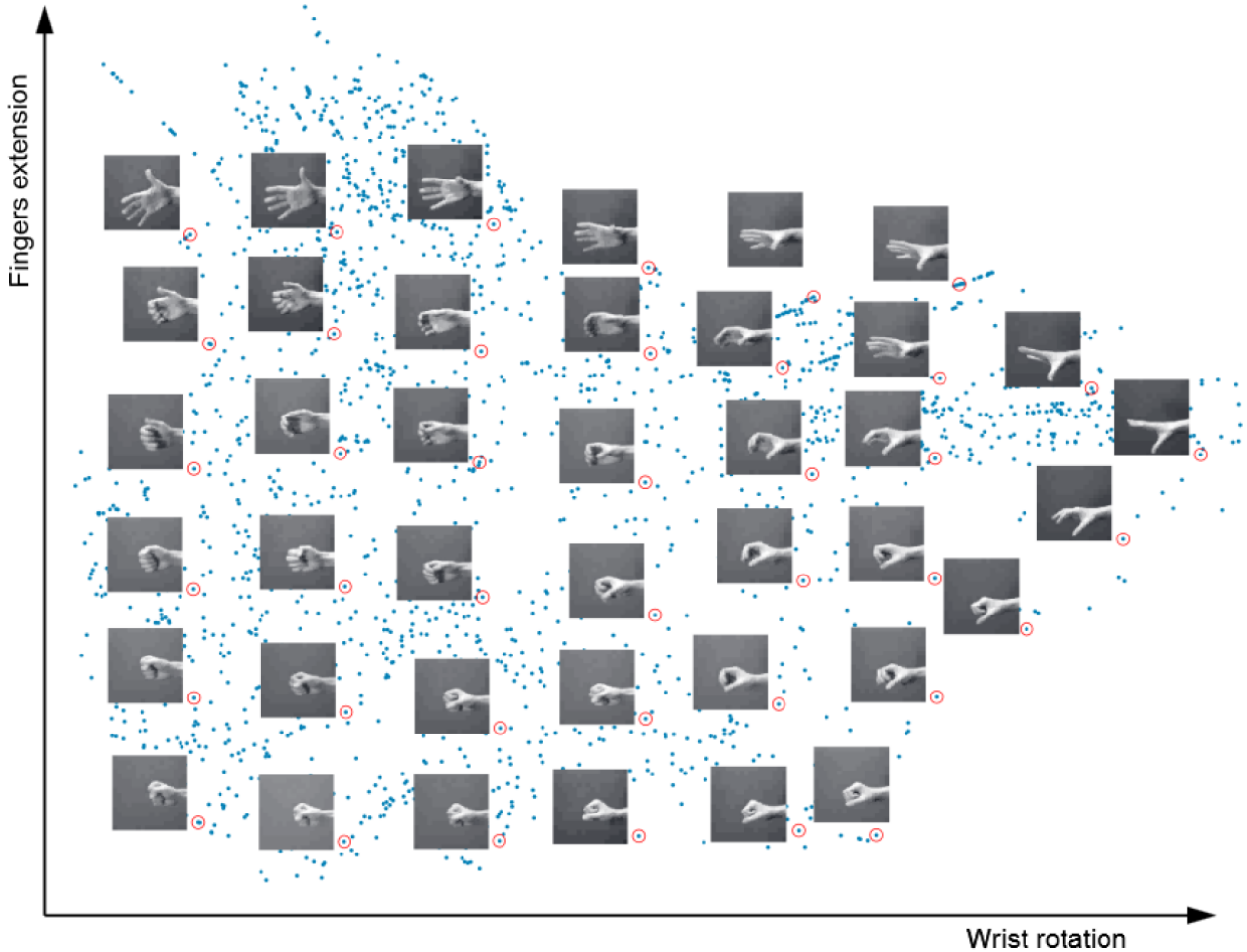
It is differentiable.

It is invertible via the encoder E.

PCA puts the data on a linear (flat) manifold, since D simply performs a linear combination of orthogonal vectors.

Shao et al, "The Riemannian geometry of deep generative models", 2018

Manifolds and generative models



Have a look at the notebook:
https://colab.research.google.com/github/erodola/ML-s2-2025/blob/main/labs/05_PCA.ipynb

Variational autoencoders (VAE)

AEs can map similar inputs to dissimilar latent codes.

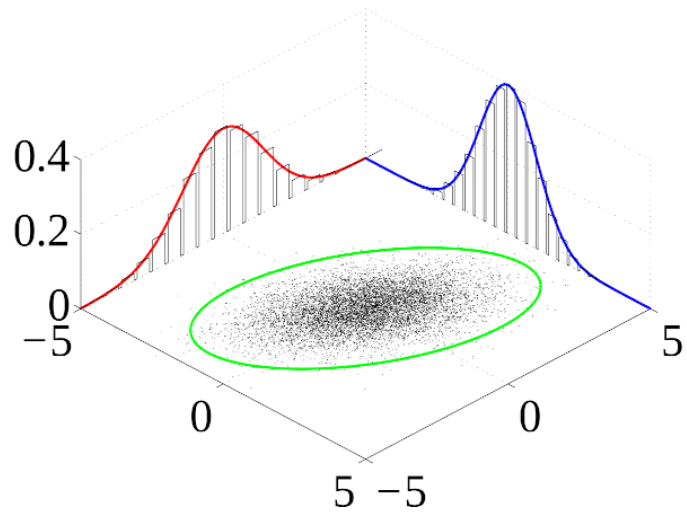
Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoders (VAE)

AEs can map similar inputs to dissimilar latent codes.

A variational AE learns a probability distribution on the latent space.

- The data is seen as a sampling of the distribution.
- The distribution is fixed and decided a priori (e.g. Gaussian).



Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in p is its entropy:

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in p is its entropy:

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions p and q, the Kullback-Leibler divergence:

$$KL(p\|q) \approx H(q) - H(p)$$

measures their dissimilarity in terms of their entropy.

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in p is its entropy:

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions p and q, the Kullback-Leibler divergence:

$$KL(p\|q) \approx -\sum q(\mathbf{x}) \log q(\mathbf{x}) + \sum p(\mathbf{x}) \log p(\mathbf{x})$$

measures their dissimilarity in terms of their entropy.

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in p is its entropy:

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions p and q, the Kullback-Leibler divergence:

$$KL(p\|q) = -\sum p(\mathbf{x}) \log q(\mathbf{x}) + \sum p(\mathbf{x}) \log p(\mathbf{x}) \geq 0$$

measures their dissimilarity in terms of their entropy.

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in p is its entropy:

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions p and q, the Kullback-Leibler divergence:

$$KL(p\|q) = \sum p(\mathbf{x}) \log \frac{p(\mathbf{x})}{q(\mathbf{x})} \geq 0$$

measures their dissimilarity in terms of their entropy.

Entropy and divergence

The information carried by an event $\mathbf{x}$ can be quantified as:

$$I(\mathbf{x}) = -\log p(\mathbf{x})$$

The average information encoded in p is its entropy:

$$H(p) = -\sum_{\mathbf{x}} p(\mathbf{x}) \log p(\mathbf{x})$$

Given two distributions p and q, the Kullback-Leibler divergence:

$$KL(p\|q) = -\sum p(\mathbf{x}) \log \frac{q(\mathbf{x})}{p(\mathbf{x})} \geq 0$$

measures their dissimilarity in terms of their entropy.

Variational inference

In our scenario: **x** is a data point, and **z** is a latent code.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

However, this integral:

$$p_{\boldsymbol{\theta}}(\mathbf{x}) = \int p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})d\mathbf{z}$$

is intractable over the entire latent space.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})/p_{\boldsymbol{\theta}}(\mathbf{x})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} + \log \frac{1}{p_{\boldsymbol{\theta}}(\mathbf{x})} \right)$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \left(\log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} - \log p_{\boldsymbol{\theta}}(\mathbf{x}) \right)$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} + \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} + \log p_{\boldsymbol{\theta}}(\mathbf{x}) \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} + \log p_{\boldsymbol{\theta}}(\mathbf{x}) \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$$

$\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = 1$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\min_{\boldsymbol{\phi},\boldsymbol{\theta}} \; -\sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} + \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} - \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} - \cancel{\log p_{\boldsymbol{\theta}}(\mathbf{x})} \quad \text{relax}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational inference

In our scenario: $\mathbf{x}$ is a data point, and $\mathbf{z}$ is a latent code.

Define a parametric probabilistic encoder:

$$p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}) = \frac{p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})p_{\boldsymbol{\theta}}(\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})} = \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{x})}$$

Instead, compute an approximation $q_{\boldsymbol{\phi}^*}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})$ with:

$$\boldsymbol{\phi}^* = \arg\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \; ELBO_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) - \cancel{\log p_{\boldsymbol{\theta}}(\mathbf{x})} \quad \text{relax}$$

where $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$ is a neural network with weights $\boldsymbol{\phi}$ and:

$$ELBO_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x}) = \sum_{\mathbf{z}} q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \log \frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \leq \log p_{\boldsymbol{\theta}}(\mathbf{x})$$

is called the Evidence variational Lower BOund.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \; ELBO_{\boldsymbol{\phi},\boldsymbol{\theta}}(\mathbf{x})$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\phi,\theta} \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\phi,\theta} \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\phi,\theta} \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \left(\log p_\theta(\mathbf{x}|\mathbf{z}) + \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})} \right)$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\phi,\theta} \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log p_\theta(\mathbf{x}|\mathbf{z}) + \sum_{\mathbf{z}} q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{p_\theta(\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\boldsymbol{\phi},\boldsymbol{\theta}} \; \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z}) - KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \| p_{\boldsymbol{\theta}}(\mathbf{z}))$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\max_{\phi,\theta} \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} - KL(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

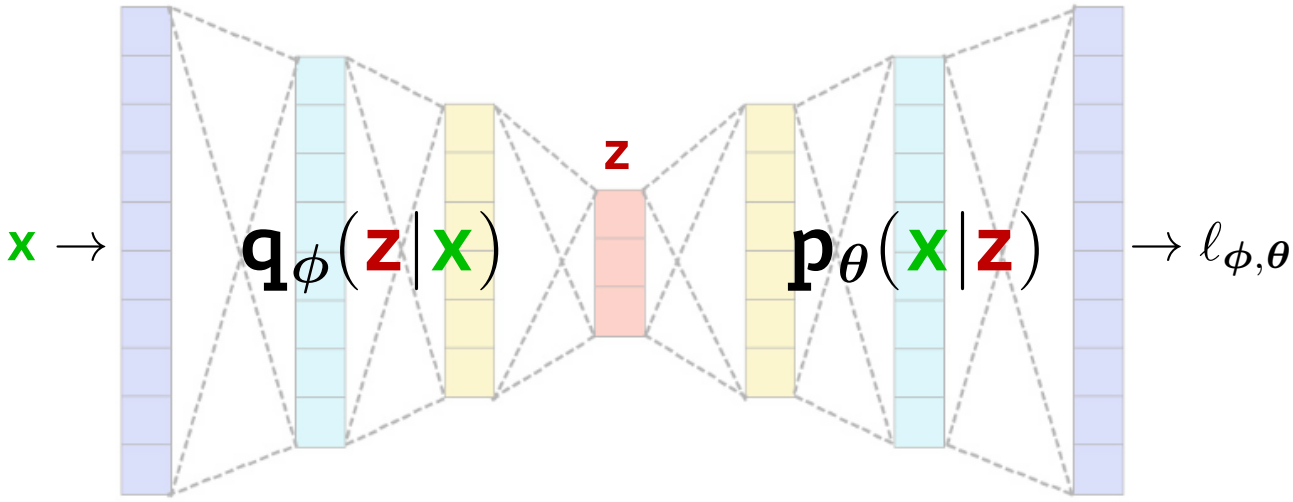
Variational autoencoder (VAE)

$$\max_{\phi,\theta} \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} \underbrace{-KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{ensures } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z})}$$



Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\substack{\text{likelihood of observing} \\ \mathbf{x} \text{ given } \mathbf{z}}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{ensures } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

$$p(\mathbf{z}) = \mathcal{N}_{\mathbf{0},\mathbf{I}}(\mathbf{z})$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{ensures } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

$$p(\mathbf{z}) = \mathcal{N}_{\mathbf{0},\mathbf{I}}(\mathbf{z})$$

The probabilistic encoder also generates a Gaussian distribution:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{z})$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{ensures } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

$$p(\mathbf{z}) = \mathcal{N}_{\mathbf{0},\mathbf{I}}(\mathbf{z})$$

The probabilistic encoder also generates a Gaussian distribution:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{z})$$

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\substack{\text{likelihood of observing} \\ \mathbf{x} \text{ given } \mathbf{z}}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{ensures } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

$$p(\mathbf{z}) = \mathcal{N}_{\mathbf{0},\mathbf{I}}(\mathbf{z})$$

The probabilistic encoder also generates a Gaussian distribution:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{z})$$

$\boldsymbol{\mu}, \boldsymbol{\sigma}$ are functions of the input $\mathbf{x}$ and the network parameters $\boldsymbol{\phi}$.

The probabilistic encoder outputs $\boldsymbol{\mu}, \boldsymbol{\sigma}$, not $\mathbf{z}$.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\substack{\text{likelihood of observing} \\ \mathbf{x} \text{ given } \mathbf{z}}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{ensures } q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) \approx p_{\boldsymbol{\theta}}(\mathbf{z})}$$

Fixed Gaussian prior over the latent variables:

$$p(\mathbf{z}) = \mathcal{N}_{\mathbf{0},\mathbf{I}}(\mathbf{z})$$

The probabilistic encoder also generates a Gaussian distribution:

$$q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x}) = \mathcal{N}_{\boldsymbol{\mu},\boldsymbol{\sigma}}(\mathbf{z})$$

$\boldsymbol{\mu}, \boldsymbol{\sigma}$ are functions of the input $\mathbf{x}$ and the network parameters $\boldsymbol{\phi}$.

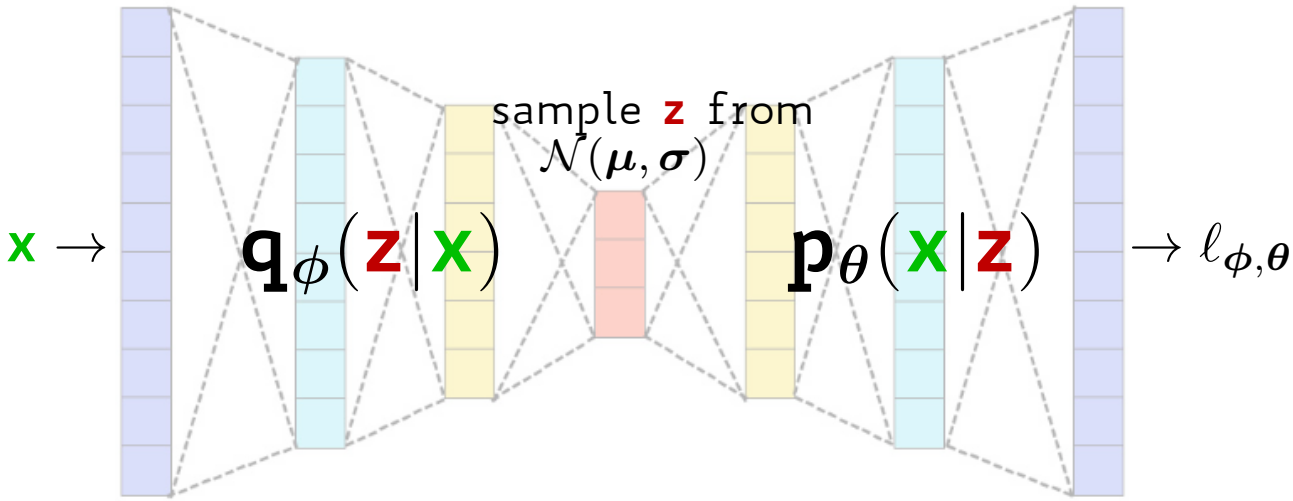
The probabilistic encoder outputs $\boldsymbol{\mu}, \boldsymbol{\sigma}$, not $\mathbf{z}$.

Using Gaussians, the KL term has a closed form.

Kingma and Welling, "Auto-Encoding Variational Bayes", 2013

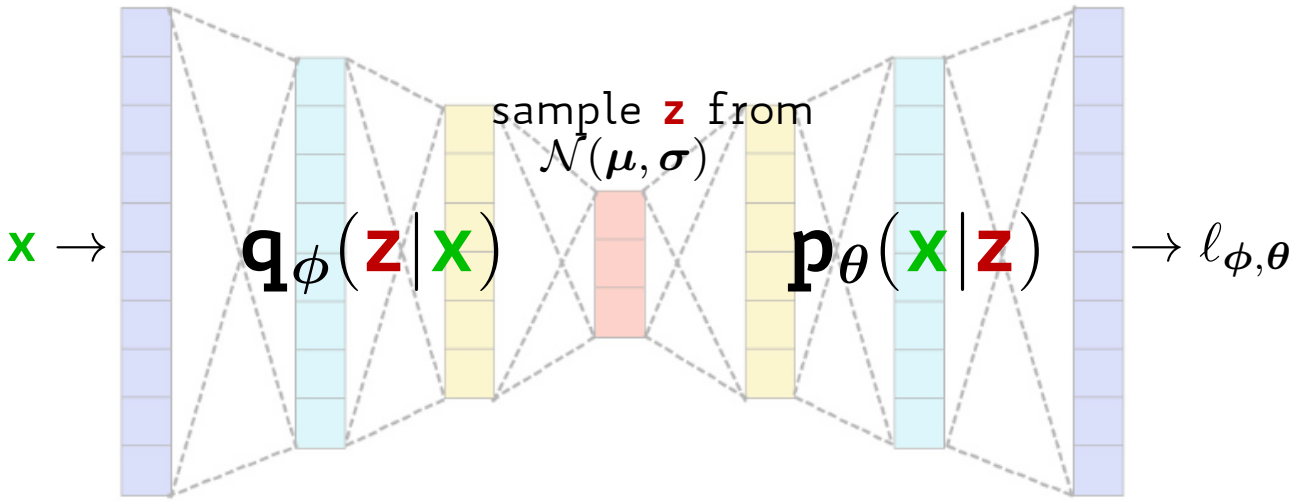
Variational autoencoder (VAE)

$$\ell_{\phi,\theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{likelihood of observing } \mathbf{x} \text{ given } \mathbf{z}} \underbrace{+KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))}_{\text{ensures } q_\phi(\mathbf{z}|\mathbf{x}) \approx p_\theta(\mathbf{z})}$$



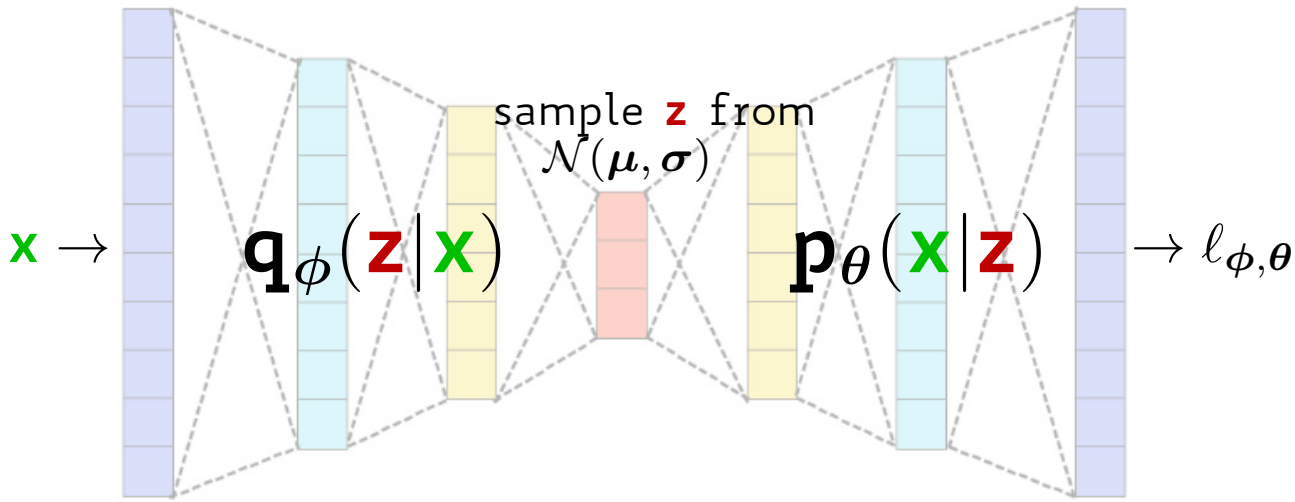
Variational autoencoder (VAE)

$$\ell_{\phi,\theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss (choose your own)}} \underbrace{+KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))}_{\text{regularizer}}$$



Variational autoencoder (VAE)

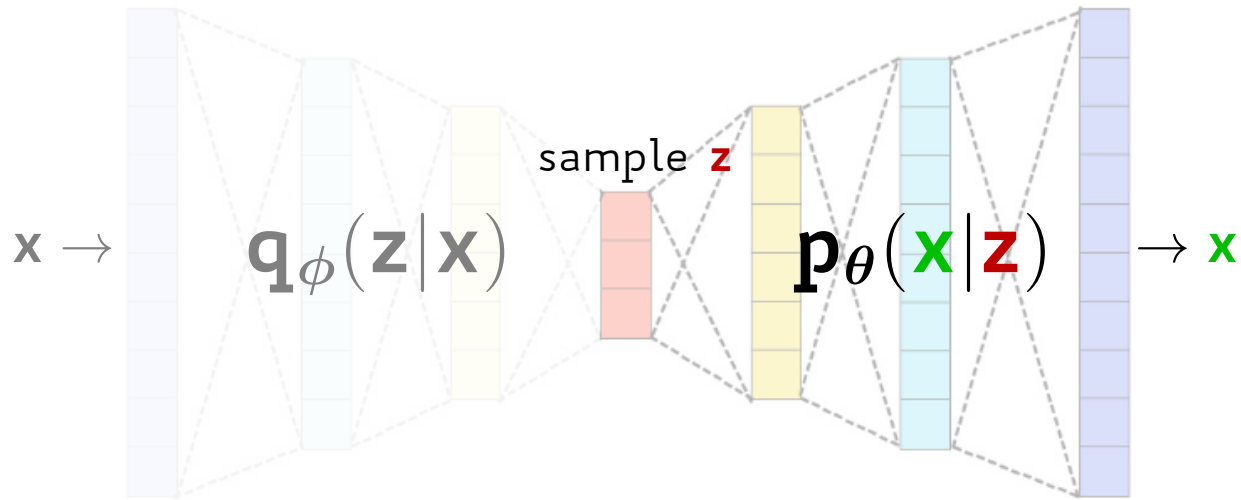
$$\ell_{\phi,\theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss (choose your own)}} \underbrace{+KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))}_{\text{regularizer}}$$



For clarity, classical AEs are called deterministic AEs.

Variational autoencoder (VAE)

$$\ell_{\boldsymbol{\phi},\boldsymbol{\theta}} = \underbrace{-\mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})} \log p_{\boldsymbol{\theta}}(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss (choose your own)}} \underbrace{+KL(q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\|p_{\boldsymbol{\theta}}(\mathbf{z}))}_{\text{regularizer}}$$

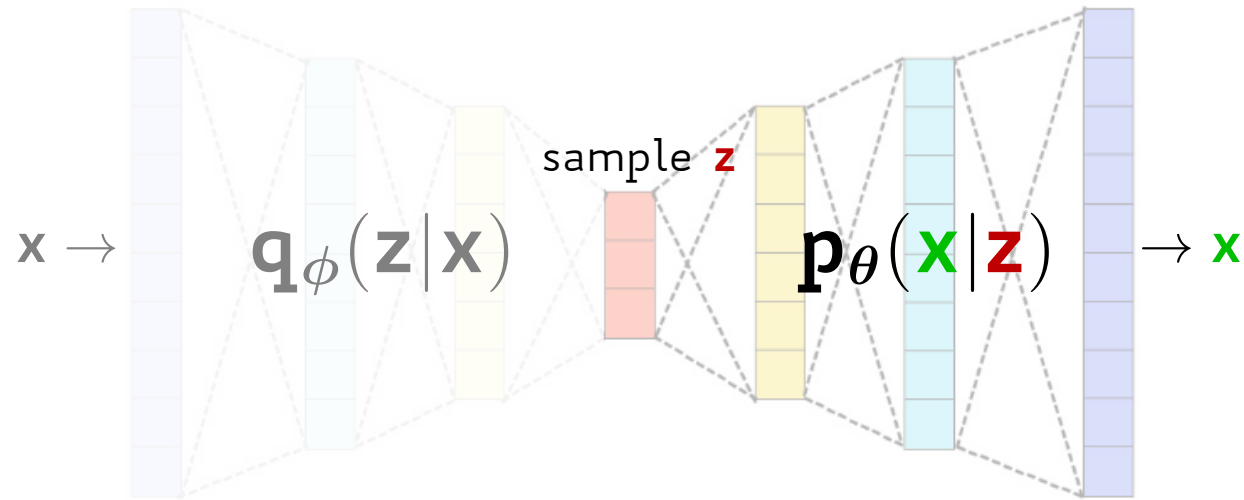


For clarity, classical AEs are called deterministic AEs.

Generation: Sample $\mathbf{z}$ according to the learned distribution.

Variational autoencoder (VAE)

$$\ell_{\phi,\theta} = \underbrace{-\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})}_{\text{reconstruction loss (choose your own)}} \underbrace{+KL(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))}_{\text{regularizer}}$$

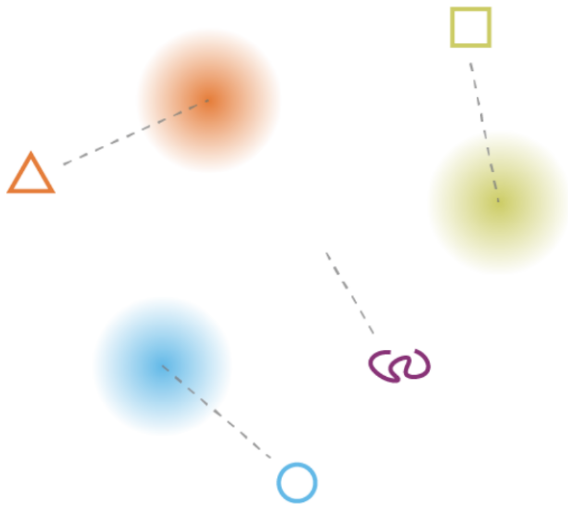


For clarity, classical AEs are called deterministic AEs.

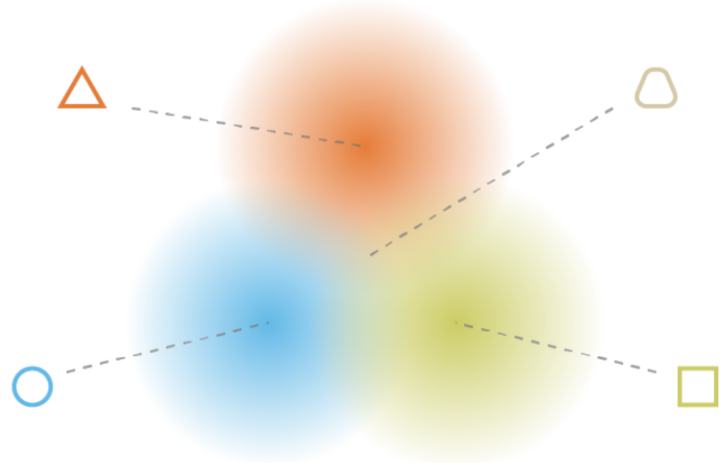
Generation: Sample $\mathbf{z}$ according to the learned distribution.

Note: also the decoder outputs a probability distribution.
If the reconstruction loss does not need a distribution, take $\mathbf{x} := \mu$.

Regularizing effect



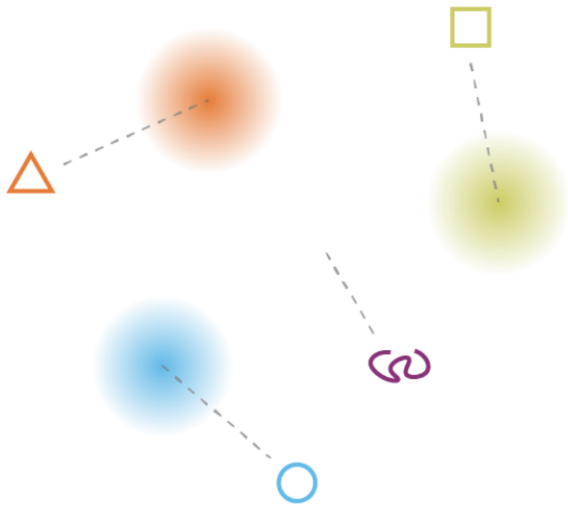
latent space of deterministic AE



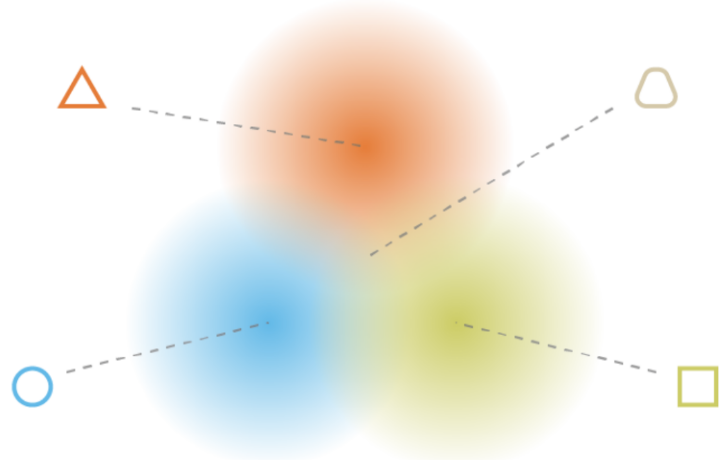
latent space of variational AE

The requirement $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ leads to a compact latent space.
Small distances $\|\mathbf{z}_1 - \mathbf{z}_2\| \Rightarrow$ small changes in the decodings.

Regularizing effect



latent space of deterministic AE

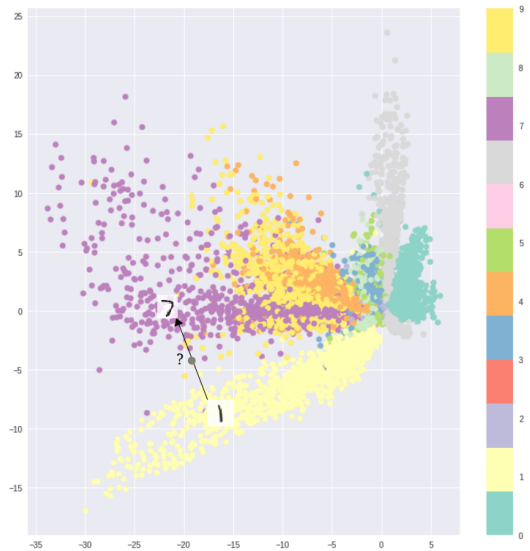


latent space of variational AE

The requirement $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ leads to a compact latent space.
Small distances $\|\mathbf{z}_1 - \mathbf{z}_2\| \Rightarrow$ small changes in the decodings.

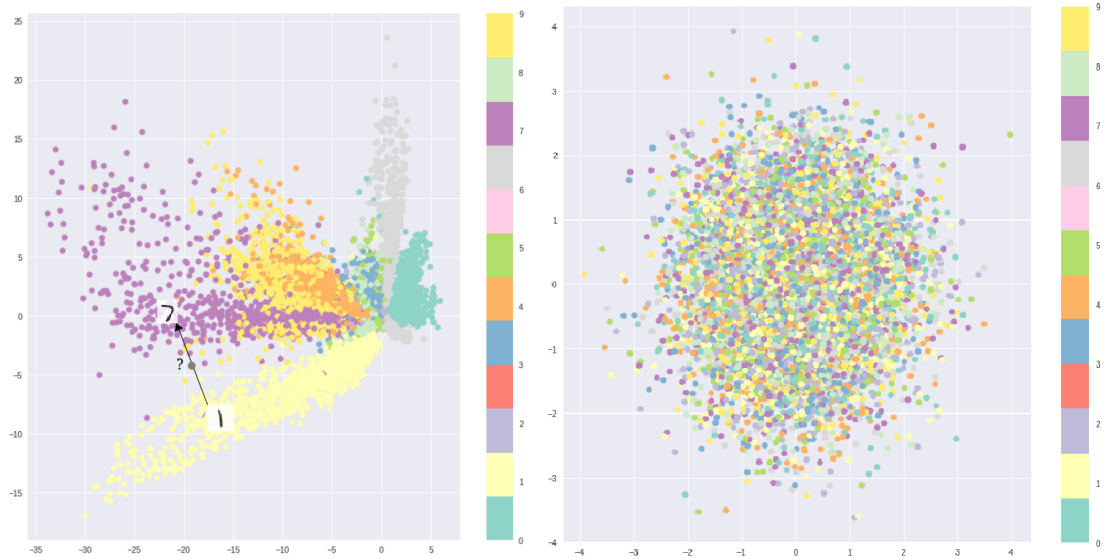
A deterministic AE might generate arbitrary noise.

Regularizing effect



Only reconstruction loss (BCE)

Regularizing effect



Only reconstruction loss (BCE)

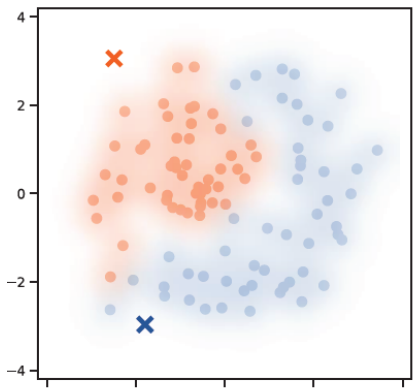
Only continuity loss (KLD)

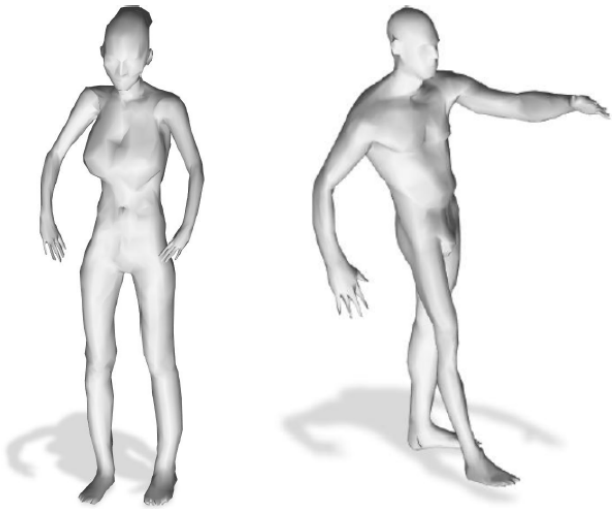
Regularizing effect

The distributional regularizer brings smoothness in the learned space.

Unlikely samples

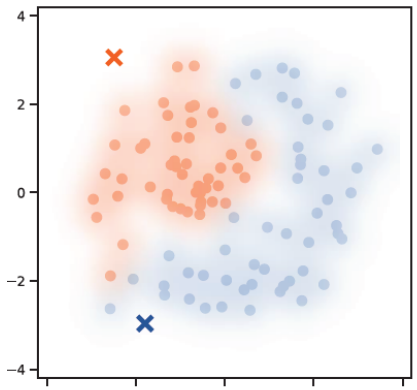
Sampling on the boundaries of the learned distribution leads to the generation of unlikely samples:

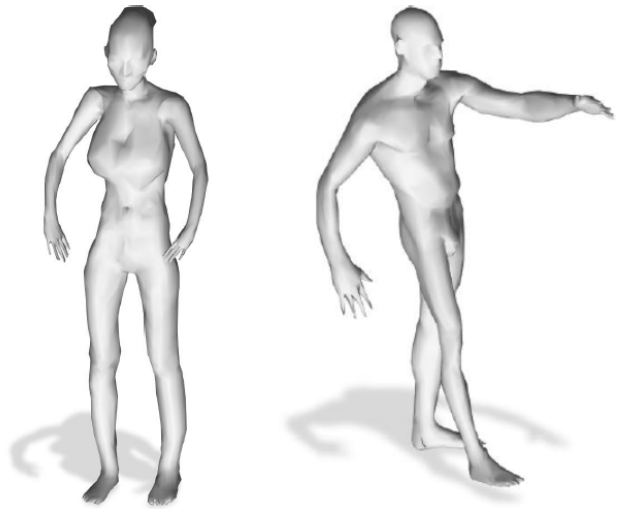




Unlikely samples

Sampling on the boundaries of the learned distribution leads to the generation of unlikely samples:



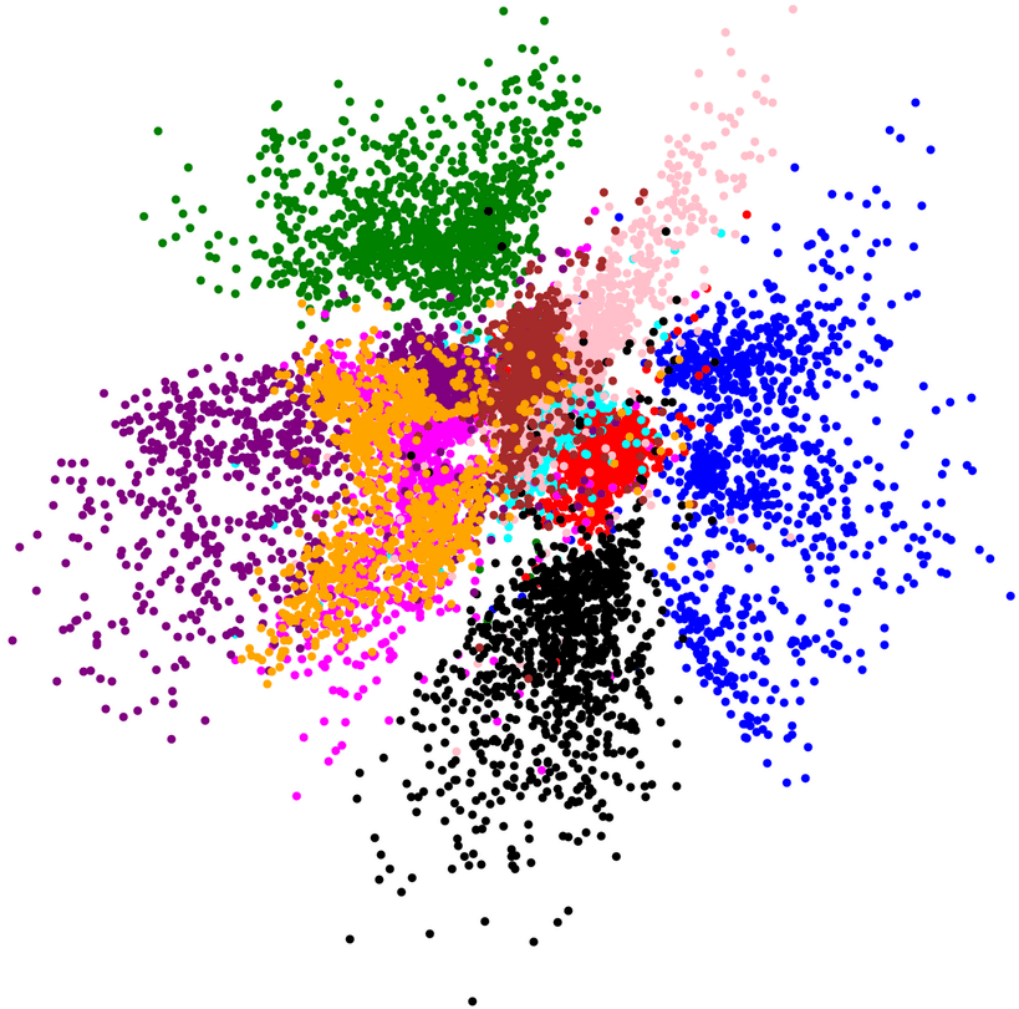


Posterior collapse: the latents are ignored if the decoder is powerful enough to model the data perfectly!

Razavi et al, "Preventing Posterior Collapse with delta-VAEs", 2019

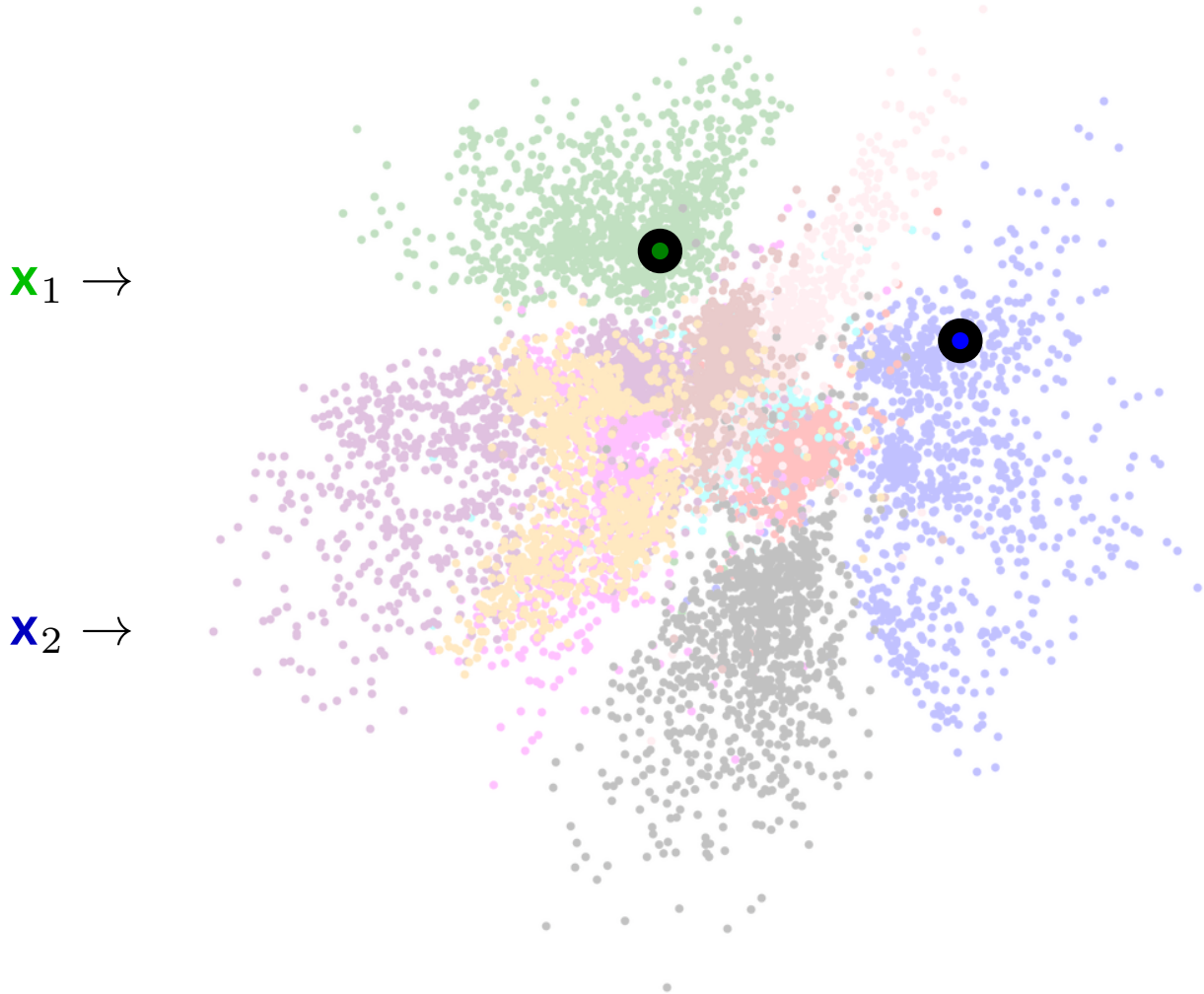
VAE interpolation

Assume we have a trained VAE, colors correspond to different classes:



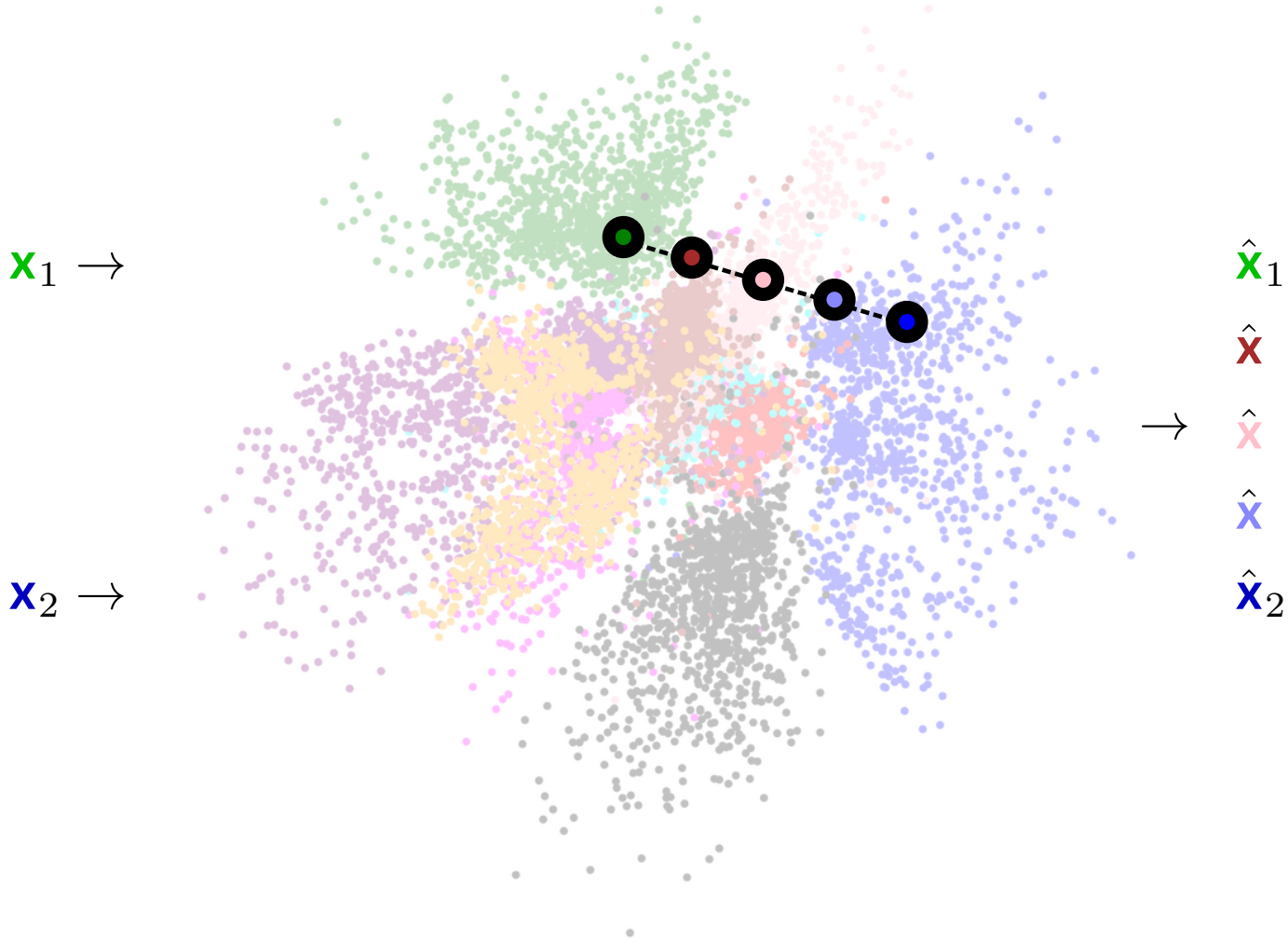
VAE interpolation

Assume we have a trained VAE, colors correspond to different classes:



VAE interpolation

Assume we have a trained VAE, colors correspond to different classes:



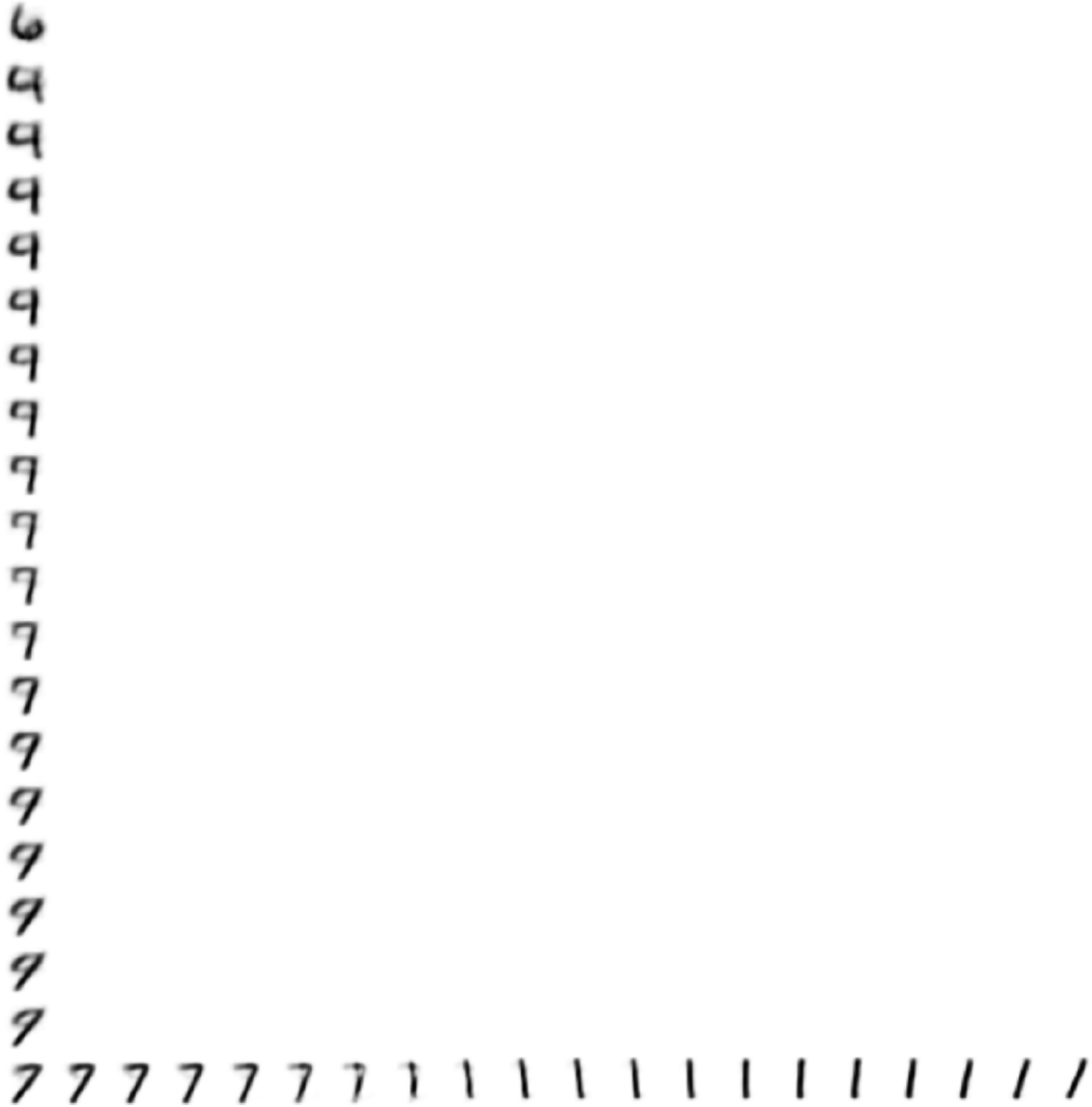
The interpolated samples are **not** part of the training set.

VAE interpolation

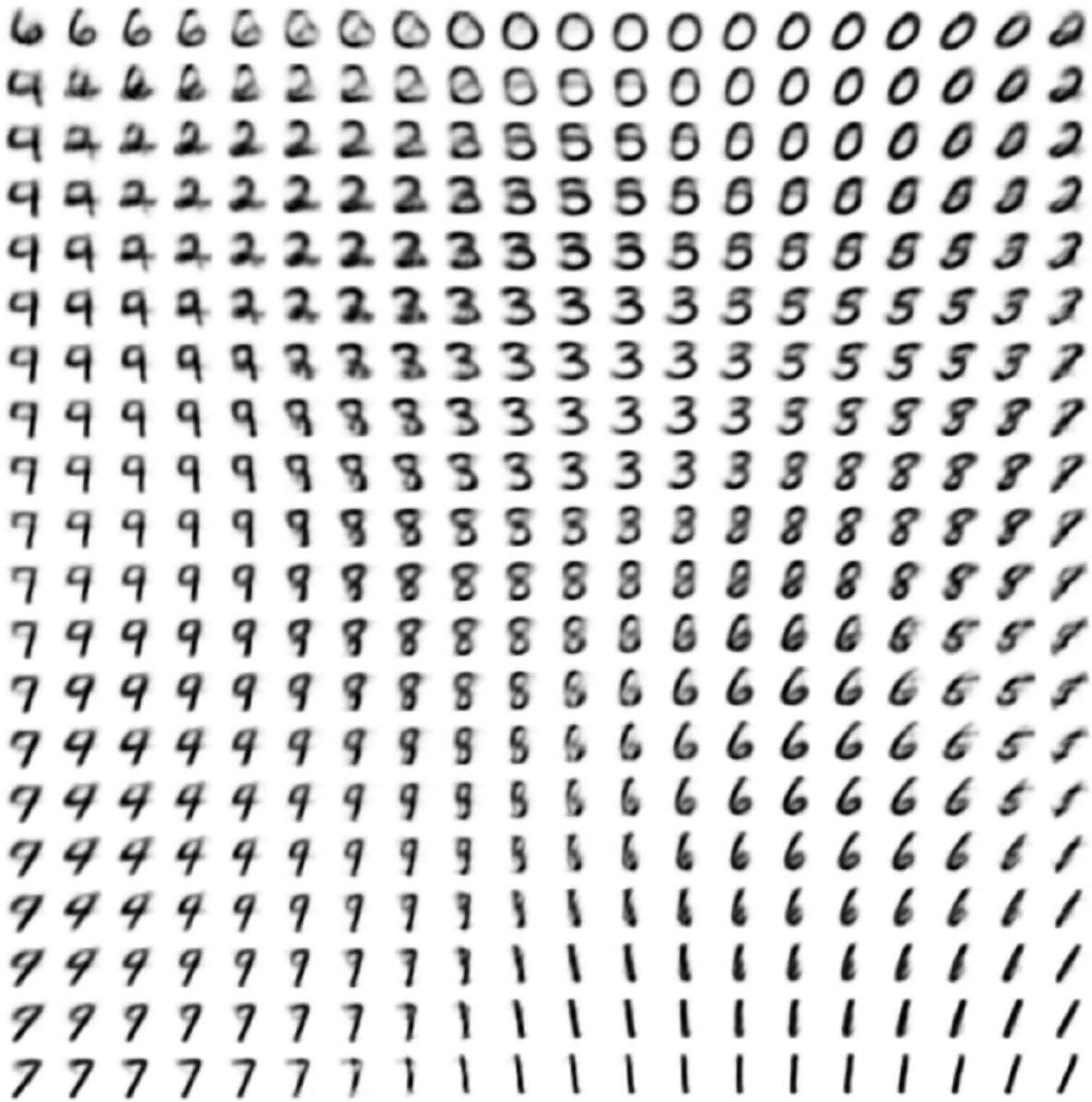
VAE interpolation

VAE interpolation

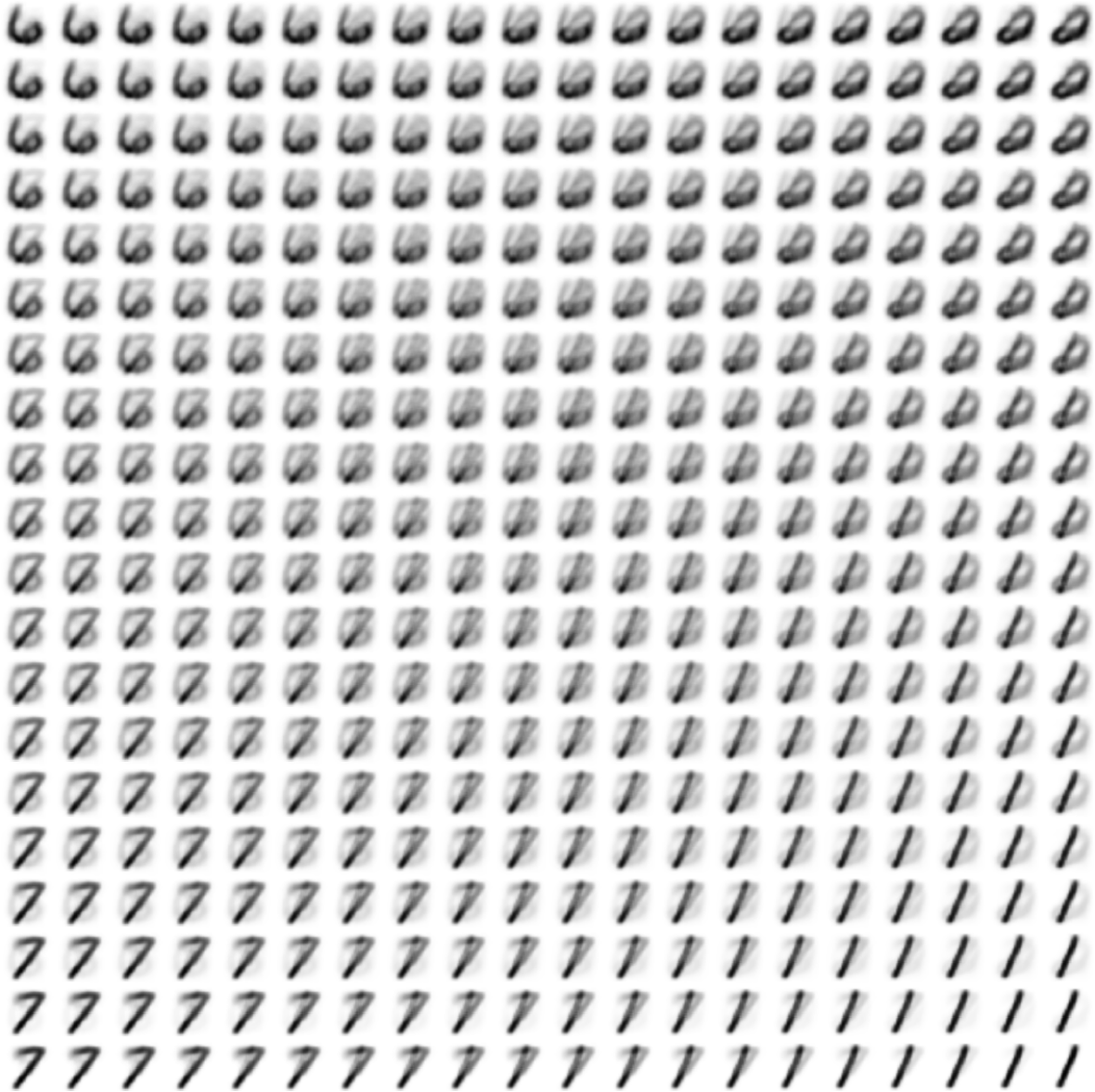
VAE interpolation



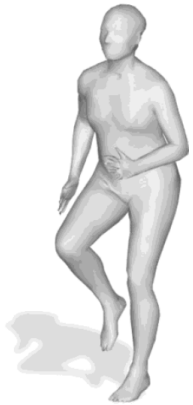
VAE interpolation



Euclidean interpolation

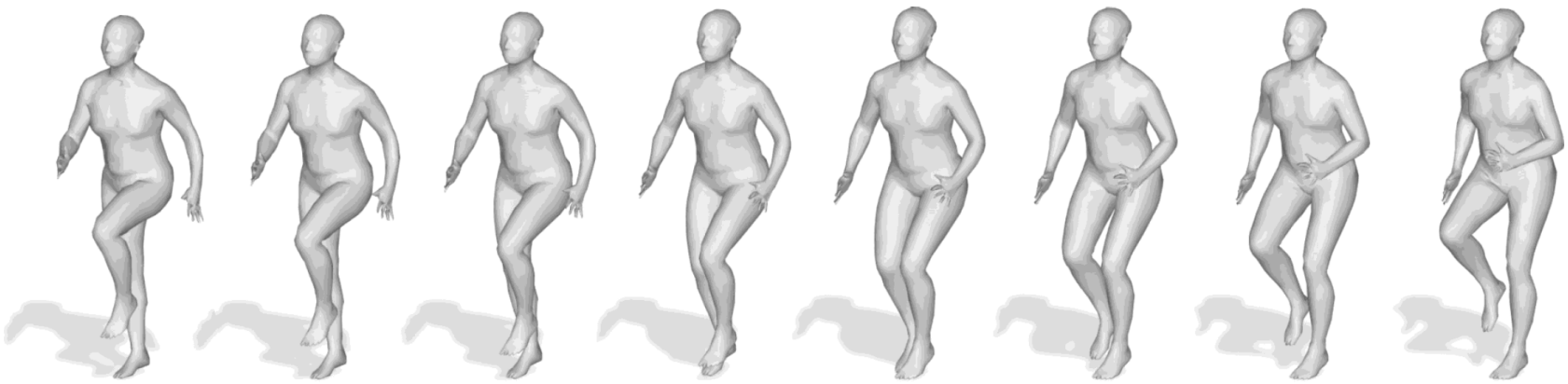


Example: Deformable 3D shapes



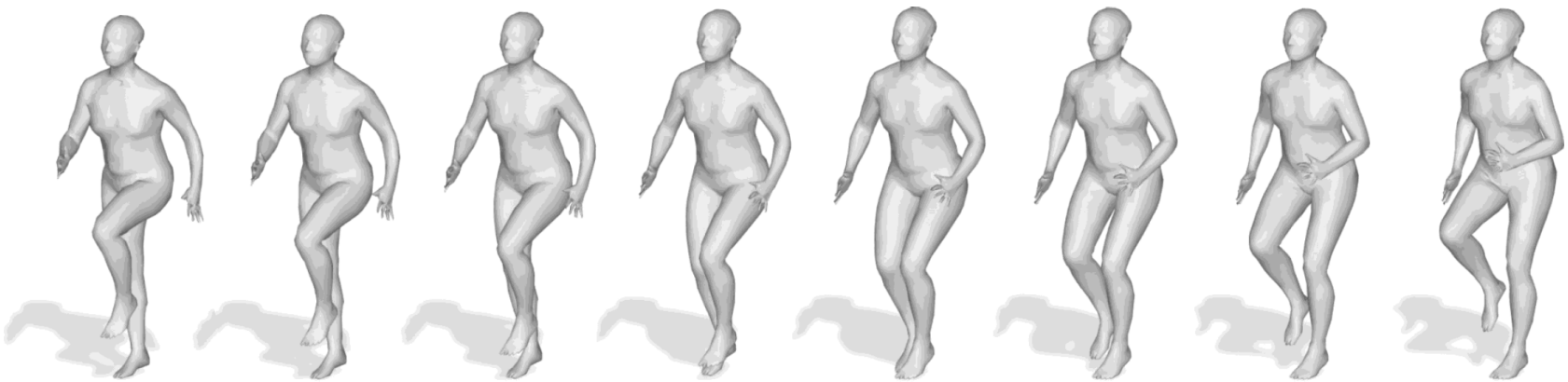
Cosmo, Norelli, Halimi, Kimmel, Rodolà, "LIMP: Learning Latent Shape Representations with Metric Preservation Priors", 2020

Example: Deformable 3D shapes



Cosmo, Norelli, Halimi, Kimmel, Rodolà, "LIMP: Learning Latent Shape Representations with Metric Preservation Priors", 2020

Example: Deformable 3D shapes

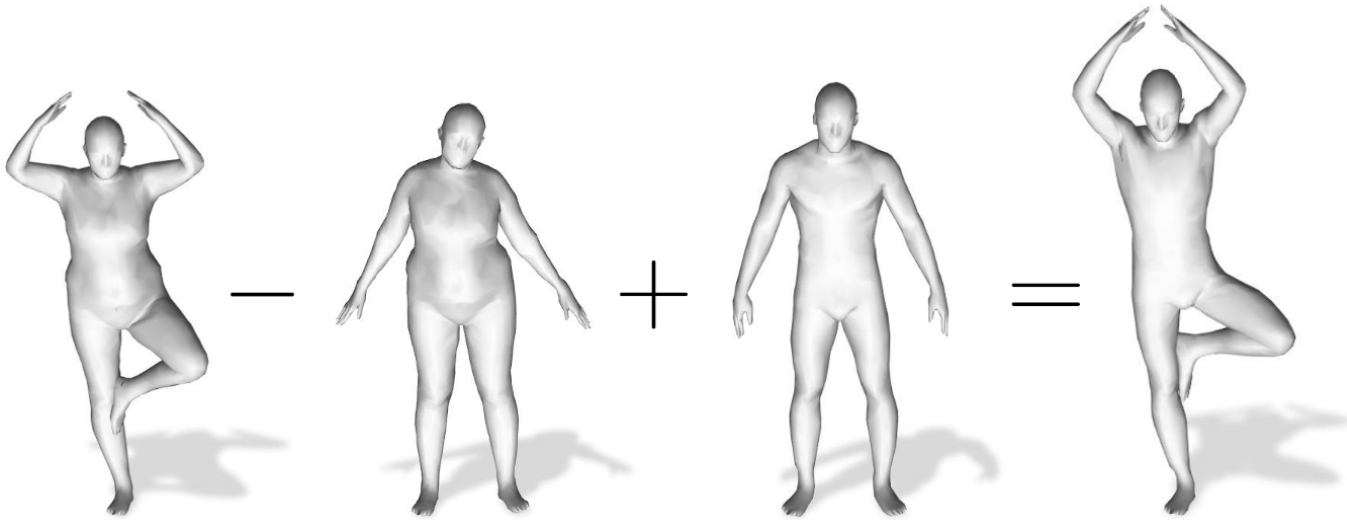


The latent space can be explored along different dimensions (disentanglement):

Cosmo, Norelli, Halimi, Kimmel, Rodolà, "LIMP: Learning Latent Shape Representations with Metric Preservation Priors", 2020

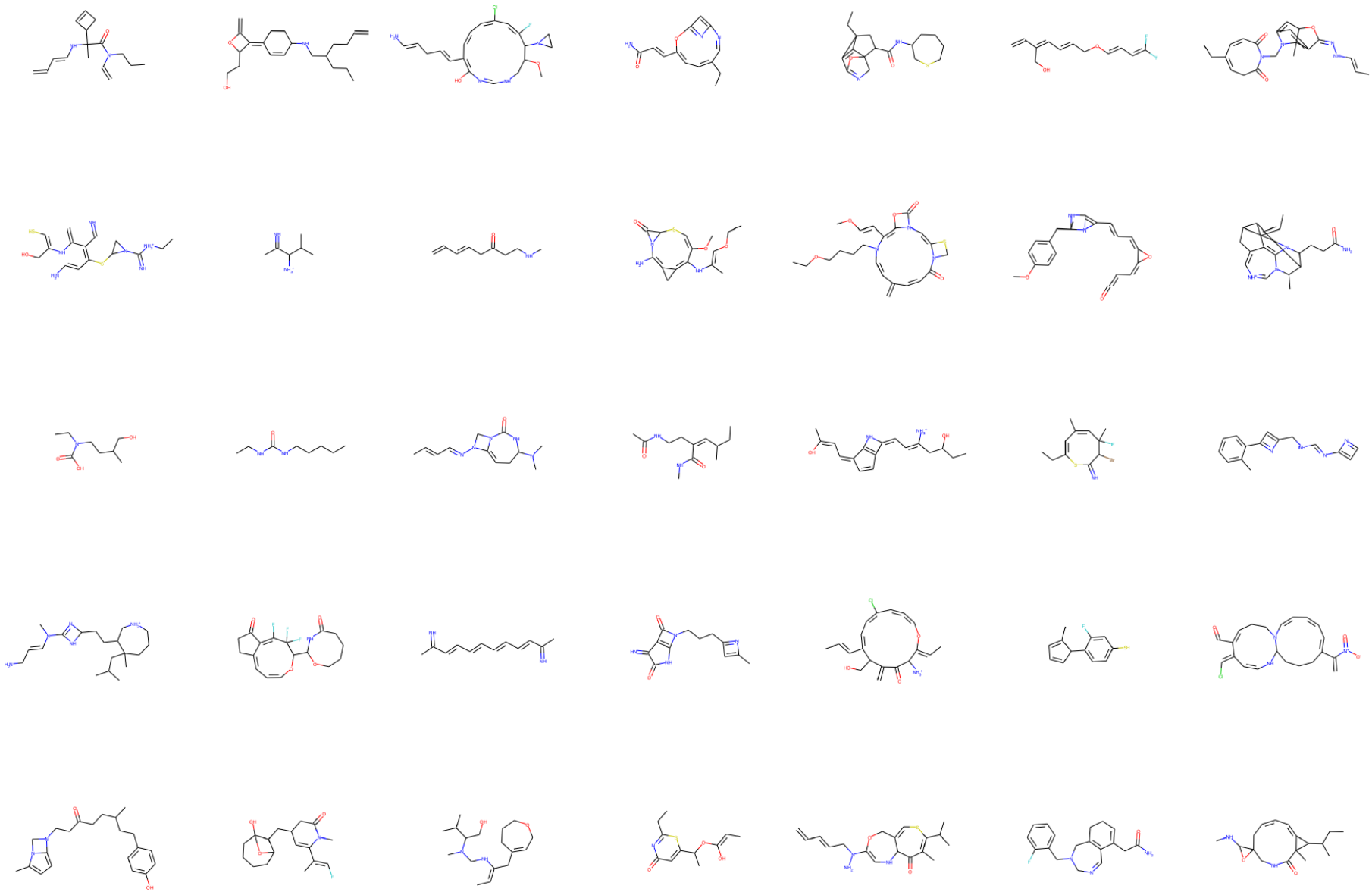
Analogies

Algebraic manipulation of codes leads to the following:



In NLP this is done by manipulating word embeddings:

King – Man + Woman = Queen



Bresson, "A Two-Step Graph Convolutional Decoder for Molecule Generation", 2019

Razavi et al, “Generating Diverse High-Fidelity Images with VQ-VAE-2”, 2019