Deep Learning for Computer Vision

Dr. Konda Reddy Mopuri
Mehta Family School of Data Science and Artificial Intelligence
IIT Guwahati
Aug-Dec 2022

Beyond Classification and Regression

1. Applications such as image synthesis, image-to-image transformations model high-dim signals

Beyond Classification and Regression

1. Applications such as image synthesis, image-to-image transformations model high-dim signals
2. These applications require to learn the meaningful degrees of freedom that constitute the signal

Beyond Classification and Regression

1. Applications such as image synthesis, image-to-image transformations model high-dim signals
2. These applications require to learn the meaningful degrees of freedom that constitute the signal
3. These degrees of freedom are of lesser dimensions than the signal

Example: Synthesizing Human faces

1. For generating new faces, it makes sense to capture a small number of degrees of freedom such as
   - skull size and shape
   - color of skin and eyes
   - features of nose and lips, etc.

Example: Synthesizing Human faces

1. For generating new faces, it makes sense to capture a small number of degrees of freedom such as
   - skull size and shape
   - color of skin and eyes
   - features of nose and lips, etc.
2. Even a comprehensive list of such things will be less than the number of pixels in the image (i.e. resolution)

Example: Synthesizing Human faces

1. For generating new faces, it makes sense to capture a small number of degrees of freedom such as
   - skull size and shape
   - color of skin and eyes
   - features of nose and lips, etc.
2. Even a comprehensive list of such things will be less than the number of pixels in the image (i.e. resolution)
3. If we can model these relatively small number of dimensions, we can synthesize a face with thousands of dimensions

Autoencoder

1. Neural network that maps a space to itself

Autoencoder

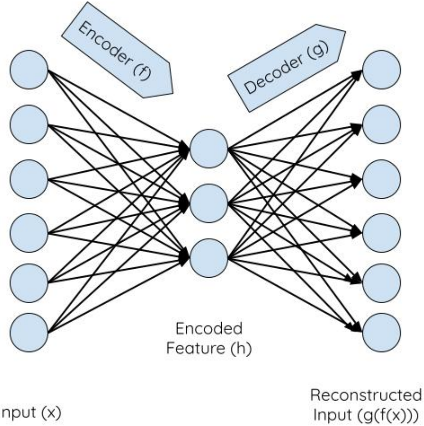
1. Neural network that maps a space to itself
2. Trained to copy its input to itself (close to, but not an identity function)

Autoencoder

1. Neural network that maps a space to itself
2. Trained to copy its input to itself (close to, but not an identity function)
3. Network consists of two parts: encoder (f) and decoder (g)

Autoencoder

1. Neural network that maps a space to itself
2. Trained to copy its input to itself (close to, but not an identity function)
3. Network consists of two parts: encoder (f) and decoder (g)



4.

Autoencoder

1. Original (input) space is of higher dimensions but the data lies in a manifold of smaller dimension

Autoencoder

1. Original (input) space is of higher dimensions but the data lies in a manifold of smaller dimension
2. Dimension of the latent space is a hyper-parameter chosen from prior knowledge, or through heuristics

Autoencoder

1. Original (input) space is of higher dimensions but the data lies in a manifold of smaller dimension
2. Dimension of the latent space is a hyper-parameter chosen from prior knowledge, or through heuristics

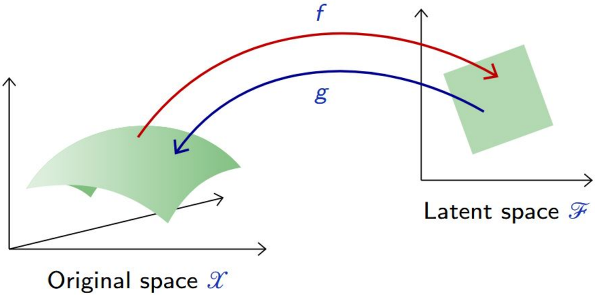


Figure credits: Francois Flueret

Autoencoder

1. Let p be the data distribution in the input space, autoencoder is characterized with the following loss

$$\mathbb{E}_{x \sim p} \left\| x - g \circ f(x) \right\|^2 \approx 0$$

Autoencoder

1. Let p be the data distribution in the input space, autoencoder is characterized with the following loss

$$\mathbb{E}_{x\sim p} \|x - g \circ f(x)\|^2 \approx 0$$

2. Training the autoencoder consists of finding the parameters for the encoder $(f(\cdot; w_f))$ and decoder $(g(\cdot; w_g)$ optimizing the following empirical loss

$$\hat{w}_f, \hat{w}_g = \underset{w_f, w_g}{\operatorname{argmin}} \frac{1}{N} \sum_n \|x_n - g(f(x_n; w_f); w_g)\|^2$$

Autoencoder

1. A simple example: f and g are linear functions $\rightarrow$ optimal solution is PCA

Autoencoder

1. A simple example: f and g are linear functions $\rightarrow$ optimal solution is PCA
2. Better results can be made possible with sophisticated transformations such as deep neural networks $\rightarrow$ Deep Autoencoders

Deep Autoencoders

```
AutoEncoder (
(encoder): Sequential (
(0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1)) (1): ReLU (inplace)
(2): Conv2d(32, 32, kernel_size=(5, 5), stride=(1, 1)) (3): ReLU (inplace)
(4): Conv2d(32, 32, kernel_size=(4, 4), stride=(2, 2)) (5): ReLU (inplace)
(6): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2)) (7): ReLU (inplace)
(8): Conv2d(32, 8, kernel_size=(4, 4), stride=(1, 1)) )
(decoder): Sequential (
(0): ConvTranspose2d(8, 32, kernel_size=(4, 4), stride=(1, 1)) (1): ReLU
(inplace)
(2): ConvTranspose2d(32, 32, kernel_size=(3, 3), stride=(2, 2)) (3): ReLU
(inplace)
(4): ConvTranspose2d(32, 32, kernel_size=(4, 4), stride=(2, 2)) (5): ReLU
(inplace)
(6): ConvTranspose2d(32, 32, kernel_size=(5, 5), stride=(1, 1)) (7): ReLU
(inplace)
(8): ConvTranspose2d(32, 1, kernel_size=(5, 5), stride=(1, 1)) ) )
```

Deep Autoencoders

Top row: original data samples
Bottom row: corresponding reconstructed samples (with linear layer of dimension 32)

Figure credits:blog.keras.io

Latent Representations

1. Consider two samples in the latent space and reconstruct the samples along the line joining these

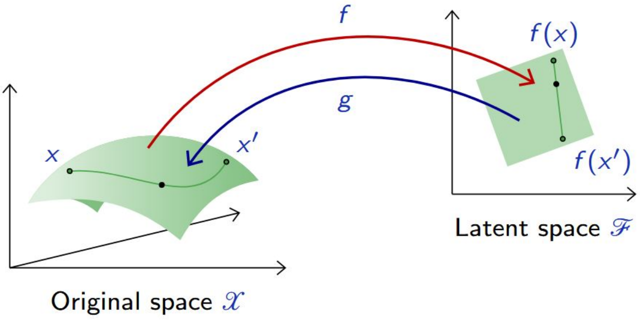


Figure credits: Francois Fleuret

Latent Representations

1. Consider two samples in the latent space and reconstruct the samples along the line joining these
2. $g(\alpha x + (1 - \alpha)x')$

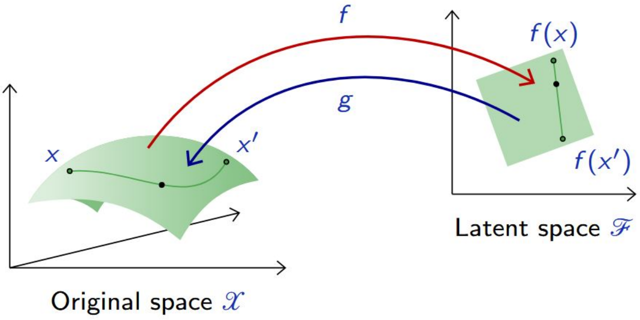
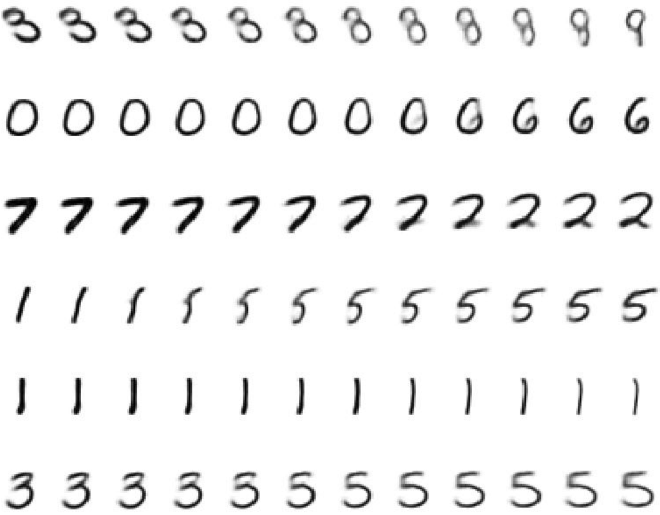


Figure credits: Francois Fleuret

Latent Representations



Generative Modeling by Autoencoder

1. Introduce a density model over the latent space

Generative Modeling by Autoencoder

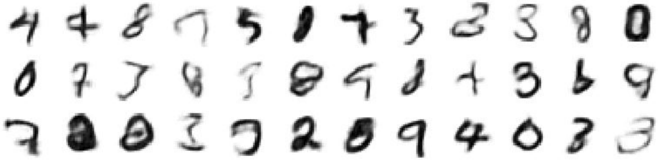
1. Introduce a density model over the latent space
2. Sample there and reconstruct using the decoder g

Generative Modeling by Autoencoder

1. Introduce a density model over the latent space
2. Sample there and reconstruct using the decoder g
3. For instance, use a Gaussian density for modeling the latent space from the training data (estimate mean and a diagonal covariance matrix)

Generative Modeling by Autoencoder

Autoencoder sampling ($d = 8$)



Autoencoder sampling ($d = 16$)

Figure credits: Francois Fleuret

Generative Modeling by Autoencoder

1. Reconstructions are not convincing

Generative Modeling by Autoencoder

1. Reconstructions are not convincing
2. Because the density model is too simple

Generative Modeling by Autoencoder

1. Reconstructions are not convincing
2. Because the density model is too simple
3. Good model still needs to capture the empirical distribution on the data although in a lower dimensional space

Besides dimensionality reduction

1. Autoencoders can capture the dependencies across signal components

Besides dimensionality reduction

1. Autoencoders can capture the dependencies across signal components
2. This can help to restore the missing components from an input

Besides dimensionality reduction

1. In this scenario, we may ignore the encoder/decoder architecture

Besides dimensionality reduction

1. In this scenario, we may ignore the encoder/decoder architecture
2. Goal in this case is not to learn a ϕ such that $\phi(X) \approx X$

Besides dimensionality reduction

1. In this scenario, we may ignore the encoder/decoder architecture
2. Goal in this case is not to learn a ϕ such that $\phi(X) \approx X$
3. It is to learn a ϕ such that $\phi(\tilde{X}) \approx X$, where $\tilde{X}$ is a perturbed version of X

Besides dimensionality reduction

1. In this scenario, we may ignore the encoder/decoder architecture
2. Goal in this case is not to learn a ϕ such that $\phi(X) \approx X$
3. It is to learn a ϕ such that $\phi(\tilde{X}) \approx X$, where $\tilde{X}$ is a perturbed version of X
4. This is referred to as a **Denoising Autoencoder**

Denoising Autoencoder

1. This can be illustrated with an additive Gaussian noise in case of a 2D signal and MSE

$$\hat{w} = \underset{w}{\operatorname{argmin}} \frac{1}{N} \sum_{n=1}^{N} \|x_n - \phi(x_n + \epsilon_n; w)\|^2 ,$$

where x_n are data samples and ϵ_n are Gaussian random noise

Denoising Autoencoder

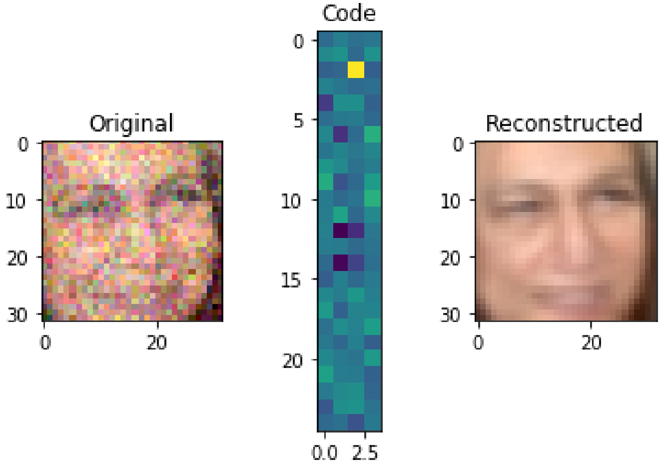


Figure credits: Ali Abdelal, https://stackabuse.com/

Weakness

1. Posterior $f_{x_n|x_n+\epsilon}$ may be multi-model

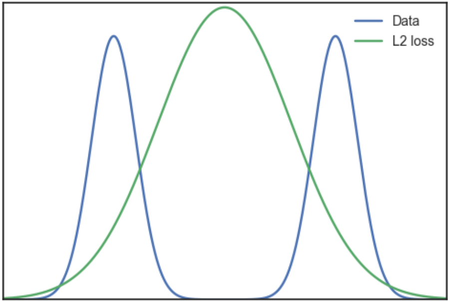


Figure credits:Patrick Langechuan Liu

Weakness

1. Posterior $f_{x_n|x_n+\epsilon}$ may be multi-model
2. L2 loss (used for training) assumes the underlying target distribution is Gaussian (thus unimodal)

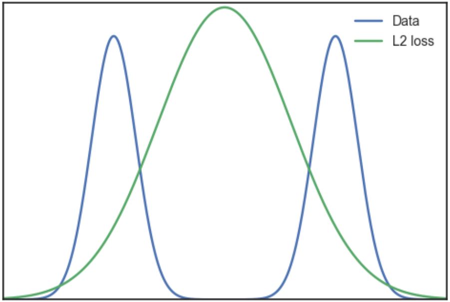


Figure credits:Patrick Langechuan Liu

Weakness

1. Posterior $f_{x_n|x_n+\epsilon}$ may be multi-model
2. L2 loss (used for training) assumes the underlying target distribution is Gaussian (thus unimodal)
3. L2 loss encourages the network to minimize loss across all modes

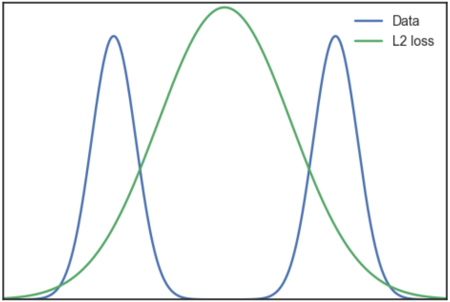


Figure credits:Patrick Langechuan Liu

Weakness

1. Posterior $f_{x_n|x_n+\epsilon}$ may be multi-model
2. L2 loss (used for training) assumes the underlying target distribution is Gaussian (thus unimodal)
3. L2 loss encourages the network to minimize loss across all modes
4. In image reconstruction applications, this leads to blurry results