Deep Learning

7.2 Autoencoders

Dr. Konda Reddy Mopuri
kmopuri@iittp.ac.in
Dept. of CSE, IIT Tirupati

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

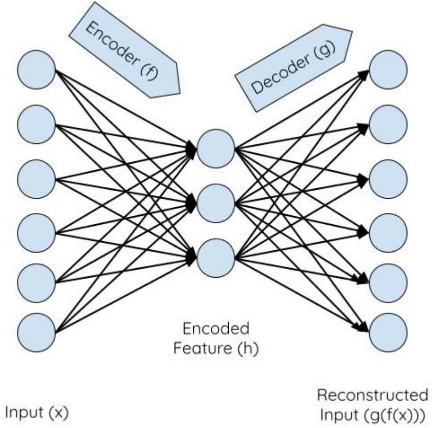
1. Neural network that maps a space to itself
2. Trained to copy its input to itself (close to, but not an identity function)

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

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

Autoencoder

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

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

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

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

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