Autoencoders

Dr. Josif Grabocka

ISMLL, University of Hildesheim

Autoencoders

Autoencoders

- An autoencoder maps a feature vector $x \in \mathbb{R}^M$ to itself.

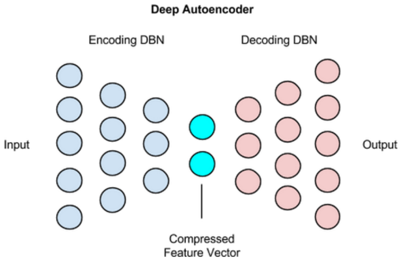

Figure 1: Illustration of an Autoencoder, Courtesy of `licdn.com`

- It is composed of two stages:
  - Encoding $f(x) = h, \quad f : \mathbb{R}^M \rightarrow \mathbb{R}^D$
  - Decoding $g(h) = \hat{x}, \quad g : \mathbb{R}^D \rightarrow \mathbb{R}^M$

Basic Autoencoders

- Formally a neural network of L layers with dimensions:

$$M = N_1 \geq N_2 \geq \cdots \geq N_{\frac{L}{2}-1} \geq N_{\frac{L}{2}} \leq N_{\frac{L}{2}+1} \leq \cdots \leq N_{L-1} \leq N_L = M$$

- The prediction model is a deep network:

$$\begin{aligned}
a_i^{(1)} &= W_{i,0}^{(1)} + \sum_{m=1}^{M} W_{i,m}^{(1)} x_{i,m}, \;\; h_i^{(1)} = f(a_i^{(1)}), \; i = 1, \ldots, M \\
&\vdots \\
a_i^{(\ell)} &= W_{i,0}^{(\ell)} + \sum_{j=1}^{N_{\ell-1}} W_{i,j}^{(\ell)} h_{i,j}^{(\ell-1)}, \;\; h_i^{(\ell)} = f(a_i^{(\ell)}), \;\; i = 1, \ldots, N_\ell \\
&\vdots \\
a_i^{(L)} &= W_{i,0}^{(L)} + \sum_{j=1}^{N_{L-1}} W_{i,j}^{(L)} h_{i,j}^{(L-1)}, \;\; \hat{x}_i^{(L)} = a_i^{(L)}, \; i = 1, \ldots, M
\end{aligned}$$

Learning Autoencoders

- The encoder function f is:

$$f\left(x; W^{(1)}, \ldots, W^{\left(\frac{L}{2}\right)}\right) = h^{\left(\frac{L}{2}\right)}$$

- The decoder function g is:

$$g\left(h^{\left(\frac{L}{2}\right)}; W^{\left(\frac{L}{2}+1\right)}, \ldots, W^{(L)}\right) = h^{(L)} = \hat{x}$$

- Ultimately the reconstruction loss is:

$$\underset{W}{\operatorname{argmin}} \sum_{x \in \mathcal{D}ata} \sum_{m=1}^{M} \left(x_m - g\left(f\left(x\right)\right)_m\right)^2$$

- Learn W through backpropagation (at the board).

Copy-Through Phenomenon

- Autoencoders can learn to copy through data.
  - What if $L = 2$, $N_1 = N_2 = M$ and $W^{(1)}_{i,j} = W^{(2)}_{i,j} = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{else} \end{cases}$
- On the other hand, for diverse applications such as dimensionality reduction and feature learning, it is important to extract salient latent features.
- Therefore models should be under-complete and should closely, but not exactly, reconstruct the input
- In that aspect, autoencoders need to be regularized

Sparse Autoencoders

- Remembering $f(x) = h$ and $g(h) = \hat{x}$ regularize code layer h:

$$\underset{f,g}{\operatorname{argmin}} \sum_{x \in \mathcal{D}ata} \sum_{m=1}^{M} \mathcal{L}\left(x_m, g\left(f\left(x\right)\right)_m\right) + \Omega(h)$$

- In terms of the actual network, the regularized loss is:

$$\underset{W}{\operatorname{argmin}} \sum_{x \in \mathcal{D}ata} \sum_{m=1}^{M} \left(x_m - h_m^{(L)}(W)\right)^2 + \Omega\left(h^{\left(\frac{L}{2}\right)}\right)$$

- $\Omega(h)$ derived through modeling the joint distribution:

$$\log p_{\text{model}}(x, h) \quad = \quad \log \prod_h p_{\text{model}}(h, x)$$

Sparse Autoencoders (2)

- Math triviality: $\log p_{\text{model}}(h, x) = \log p_{\text{model}}(h) + \log p_{\text{model}}(x \mid h)$
- A Laplacian prior can induce sparsity:

$$p_{\text{model}}(h_i) = \frac{\lambda}{2} e^{-\lambda |h_i|}$$

- Leading to the penalty:

$$\begin{aligned} \Omega(h) &= \lambda \sum_i |h_i| \\ -\log p_{\text{model}}(h) &= \sum_i \left(\lambda |h_i| - \log \frac{\lambda}{2} \right) = \Omega(h) + \text{const} \end{aligned}$$

- What is $\frac{\partial \Omega(h)}{\partial W}$?

Denoising Autoencoders

- Rather than adding a penalty, perturbate the input $x \rightarrow \tilde{x}$ and

$$\underset{f,g}{\operatorname{argmin}} \sum_{x \in \mathcal{D}ata} \sum_{m=1}^{M} \mathcal{L}\left(x, g\left(f\left(\tilde{x}\right)\right)\right)$$

- Corrupt through masking $\tilde{x}_m = \begin{cases} x_m & \text{if Bernoulli}(p) = 1 \\ 0 & \text{else} \end{cases}$
- Denote masked indices as $\mathcal{I} = \{m \mid \tilde{x}_m = 0\}$
- Optimize the subsequent weighted loss :

$$\underset{f,g}{\operatorname{argmin}} \sum_{x \in \mathcal{D}ata} \left(\alpha \sum_{m \in \mathcal{I}}^{M} \mathcal{L}\left(x, g\left(f\left(\tilde{x}\right)\right)\right) + (1-\alpha) \sum_{m \notin \mathcal{I}} \mathcal{L}\left(x, g\left(f\left(\tilde{x}\right)\right)\right) \right)$$

- How to back-propagate?

Contractive Autoencoders

- Regularize the code $h = f(x)$ penalizing derivatives of f:

$$\operatorname*{argmin}_{f,g} \sum_{x \in \mathcal{D}ata} \sum_{m=1}^{M} \mathcal{L}\left(x_m, g\left(f\left(x\right)\right)_m\right) + \Omega(h)$$

$$\Omega(h) = \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

- For a single layer autoencoder:

$$\left\| \frac{\partial f(x)}{\partial x} \right\|_F^2 = \sum_{i,m} \left(\frac{\partial h_i}{\partial x_m} \right)^2 = \sum_{i} \left(\frac{\partial h_i}{\partial a_i} \right)^2 \sum_{m} W_{i,m}^2$$

- What is $\frac{\partial \Omega(h)}{\partial W}$?

Convolutional Autoencoders

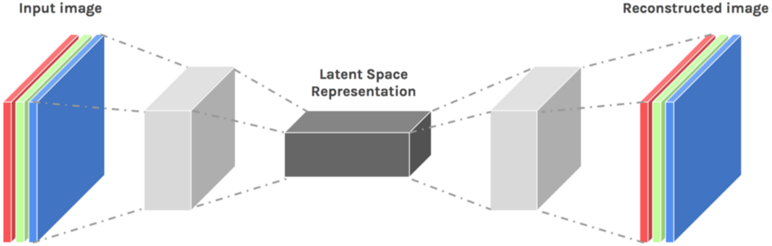

Figure 2: Convolutional Autoencoders, Courtesy: Manish Chablani

Convolutional Decoders

- Option 1: Resizing or Upsampling
- Option 2: Padding and/or Transposed Striding

, or: