Lecture 16

Introduction to Autoencoders
Figure 1: Training neural autoencoder with noisy-clean speech pairs.

Figure 1. Schematic representation of the semi-adversarial neural network architecture designed to derive perturbations that are able to confound gender classifiers while still allowing biometric matchers to perform well. The overall network consists of three sub-components: a convolutional autoencoder (subnetwork I), an auxiliary gender classifier (subnetwork II), and an auxiliary matcher (subnetwork III).


"About half (52%) of U.S. adults said they decided recently not to use a product or service because they were worried about how much personal information would be collected about them."

https://www.pewresearch.org/fact-tank/2020/04/14/half-of-americans-have-decided-not-to-use-a-product-or-service-because-of-privacy-concerns/
Lecture Overview

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders
Feature Extraction & Dimensionality Reduction

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders
Unsupervised Learning

Working with datasets *without* considering a/the *target* variable

Some Applications and Goals:

- Finding hidden structures in data
- Data compression
- Clustering
- Retrieving similar objects
- Exploratory data analysis
- Generating new examples
Principal Component Analysis (PCA)

1) Find directions of maximum variance
Principal Component Analysis (PCA)

2) Transform features onto directions of maximum variance
Principal Component Analysis (PCA)

3) Usually consider a subset of vectors of most variance (dimensionality reduction)
An Hourglass-Shaped Multilayer Perceptron

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders
A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder

Encoder

Decoder

Inputs

hidden units / embedded space / latent space / bottleneck

Outputs

= reconstructed inputs
A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder

\[ \mathcal{L}(\mathbf{x}, \mathbf{x}') = \| \mathbf{x} - \mathbf{x}' \|_2^2 = \sum_i (x_i - x_i')^2 \]

If we don't use non-linear activation functions and minimize the MSE, this is very similar to PCA.

However, the latent dimensions will not necessarily be orthogonal and will have ~ same variance.
A Basic Fully-Connected (Multilayer-Perceptron) Autoencoder

**Question:**
If we can achieve the same with PCA, which is essentially a kind of matrix factorization that is more efficient than Backprop + SGD, why bother with autoencoders?
Potential Autoencoder Applications

After training, disregard this part

Use embedding as input to classic machine learning methods (SVM, KNN, Random Forest, ...)

Or, similar to transfer learning, train autoencoder on large image dataset, then fine tune encoder part on your own, smaller dataset and/or provide your own output (classification) layer

Latent space can also be used for visualization (EDA, clustering), but there are better methods for that
A Simple Autoencoder

Reshape
28*28 => 784

Encoder
32 dim

fully connected layer + leaky relu
784 => 32

fully connected layer + sigmoid
32 => 784

32 dim

Decoder

Reshape
784 => 28*28

original

reconstructed

https://github.com/rasbt/deeplearning-models/blob/master/pytorch_ipynb/autoencoder/ae-basic.ipynb
Convolutional Autoencoders
& Transposed Convolutions / Deconvolutions

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders
A Convolutional Autoencoder

1 or more convolutional layers

Encoder

1 or more "de"convolutional layers

Decoder

original

reconstructed

8  6  0  9  1  9  3  7  1  3  1  7  8  6  2
8  6  0  9  1  9  3  7  1  3  1  7  8  6  2
Transposed Convolution

• Allows us to increase the size of the output feature map compared to the input feature map

• Synonyms:
  ▶ often also (incorrectly) called "deconvolution"  
    (mathematically, deconvolution is defined as the inverse of convolution, which is different from transposed convolutions)
  ▶ the term "unconv" is sometimes also used
  ▶ fractionally strided convolution is another (better?) term for that
Regular Convolution:

output

input


Transposed Convolution  \(^{(\text{stride} = 2)}\)

output

input

A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output. (https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8)
Transposed Convolution (3x3 kernel, stride=2)

A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output. (https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8)

Transposed Convolution (emulated with direct convolution):

Regular Convolution:  \((\text{stride} = 1)\)

<table>
<thead>
<tr>
<th>Output</th>
<th>Input</th>
</tr>
</thead>
</table>

Figure 2.1: (No padding, unit strides) Convolving a \(3 \times 3\) kernel over a \(4 \times 4\) input using unit strides (i.e., \(i = 4, k = 3, s = 1\) and \(p = 0\)).

Transposed Convolution (emulated with direct convolution):

<table>
<thead>
<tr>
<th>Output</th>
<th>Input</th>
</tr>
</thead>
</table>

Transposed Convolution

\[
\text{output} = s(n - 1) + k - 2p
\]
Deconvolution and Checkerboard Artifacts

https://distill.pub/2016/deconv-checkerboard/

A good interactive article highlighting the dangers of transposed conv.

In short, recommends replacing transposed conv. by upsampling (interpolation) followed by regular convolution.
Regular Convolution:


Transposed Convolution (stride = 2)

A Conv2DTranspose with 3x3 kernel and stride of 2x2 applied to a 2x2 input to give a 5x5 output. (https://medium.com/apache-mxnet/transposed-convolutions-explained-with-ms-excel-52d13030c7e8)
Implementing a Convolutional Autoencoder for Handwritten Digits

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders
Beyond "Regular" Fully-Connected or Convolutional Autoencoders

1. Dimensionality Reduction
2. Fully-connected Autoencoders
3. Convolutional Autoencoders
4. A Convolutional Autoencoder in PyTorch
5. Other Types of Autoencoders
Autoencoders and Dropout

Add dropout layers to force networks to learn redundant features
Denoising Autoencoder

Add dropout after the input, or add noise to the input to learn to denoise images

Sparse Autoencoder

Add L1 penalty to the loss to learn sparse feature representations

\[ \mathcal{L} = \| x - Dec(Enc(x)) \|_2^2 + \sum_i |Enc_i(x)| \]
Variational Autoencoder

\[ L^{[i]} = -\mathbb{E}_{z \sim q_w(z \mid x^{[i]})} \left[ \log p_w(x^{[i]} \mid z) \right] + \text{KL} \left( q_w(z \mid x^{[i]}) \parallel p(z) \right) \]

Expected neg. log likelihood term; wrt to encoder distribution

Kullback-Leibler divergence term where \( p(z) = \mathcal{N}(\mu = 0, \sigma^2 = 1) \)