Unsupervised Learning:
Deep Auto-encoder
Credits

• This lecture contains public domain materials from
  • Guy Nolan
  • Stefania Raimondo
  • Jason Su
  • Hung-yi Lee
Auto-encoder

encoding: $28 \times 28 = 784$

$\text{code} \rightarrow \text{NN Encoder} \rightarrow \text{code}$

Usually $< 784$

Can reconstruct the original object

$\text{code} \rightarrow \text{NN Decoder} \rightarrow \text{image}$

Compact representation of the input object

Learn together
Recap: PCA

Minimize \((x - \hat{x})^2\)

As close as possible

Input layer

\[ x \]

Hidden layer (linear)

\[ W \]

Output layer

\[ \hat{x} \]

Output of the hidden layer is the code
Deep Auto-encoder

• Of course, the auto-encoder can be deep

Initialize by RBM layer-by-layer

Symmetric is not necessary.

Deep Auto-encoder

Original Image

PCA

Deep Auto-encoder

784 → 30 → 784

784 → 1000 → 500 → 250 → 30 → 250 → 500 → 1000 → 784

0 / 2 3 4

0 / 2 3 4

q

q
DEEP AUTOENCODER EXAMPLE

https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html - By Andrej Karpathy
Auto-encoder

- De-noising auto-encoder

More: Contractive auto-encoder


Deep Auto-encoder - Example

Pixel -> tSNE

t-Distributed Stochastic Neighbor Embedding (t-SNE)
Auto-encoder – Text Retrieval

**Vector Space Model**

**Bag-of-word**

Word string: “This is an apple”

<table>
<thead>
<tr>
<th>Word</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>this</td>
<td>1</td>
</tr>
<tr>
<td>is</td>
<td>1</td>
</tr>
<tr>
<td>a</td>
<td>0</td>
</tr>
<tr>
<td>an</td>
<td>1</td>
</tr>
<tr>
<td>apple</td>
<td>1</td>
</tr>
<tr>
<td>pen</td>
<td>0</td>
</tr>
</tbody>
</table>

Semantics are not considered.
Auto-encoder – Text Retrieval

The documents talking about the same thing will have close code.

- Bag-of-word (document or query)
- LSA: project documents to 2 latent topics
Auto-encoder for CNN

Deconvolution

Unpooling

Deconvolution

Unpooling

Deconvolution

As close as possible

Convolutions

Pooling

Deconvolution

Unpooling

Deconvolution

Pooling

code
CNN - Unpooling

Alternative: simply repeat the values

Source of image:
https://leonardoaraujosantos.gitbooks.io/artificial-intelligence/content/image_segmentation.html
Actually, deconvolution is convolution.
**Auto-encoder – Pre-training DNN**

- Greedy Layer-wise Pre-training *again*
Auto-encoder – Pre-training DNN

• Greedy Layer-wise Pre-training again
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training again
Auto-encoder – Pre-training DNN

- Greedy Layer-wise Pre-training *again*

```
| Input | 784 |
|------------------|
```

```
| Output | 10  |
|------------------|
```

```
| Output | 500 |
|------------------|
```

```
| Output | 1000 |
|------------------|
```

```
| Output | 1000 |
|------------------|
```

```
| Output | 1000 |
|------------------|
```

```
| Output | 500 |
|------------------|
```

```
| Output | 10 |
|------------------|
```

```
| W^1 | Random init |
|------------------|
```

```
<table>
<thead>
<tr>
<th>W^2</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>W^3</th>
</tr>
</thead>
</table>
```

```
<table>
<thead>
<tr>
<th>W^4</th>
</tr>
</thead>
</table>
```

Find-tune by backpropagation
SIMPLE LATENT SPACE INTERPOLATION - KERAS

Encoder

Encoder

$z_1$

$z_2$
SIMPLE LATENT SPACE INTERPOLATION - KERAS

\[ Z_i = \alpha \cdot z_1 + (1 - \alpha) \cdot z_2 \]
SIMPLE LATENT SPACE INTERPOLATION — KERAS CODE EXAMPLE
SIMPLE LATENT SPACE — INTERPOLATION - KERAS
**Intuition:**
- We still aim to encode the input and to NOT mimic the identity function.
- We try to undo the effect of *corruption* process stochastically applied to the input.

*A more robust model*
DENOISING AUTOENCODERS

Use Case:
- Extract robust representation for a NN classifier.
DENOISING AUTOENCODERS

Instead of trying to mimic the identity function by minimizing:

\[ L(x, g(f(x))) \]

where \( L \) is some loss function

A DAE instead minimizes:

\[ L(x, g(f(\tilde{x}))) \]

where \( \tilde{x} \) is a copy of \( x \) that has been corrupted by some form of noise.
DENOISING AUTOENCODERS

Idea: A robust representation against noise:

- Random assignment of subset of inputs to 0, with probability \( v \).
- Gaussian additive noise.
DENOISING AUTOENCODERS

- Reconstruction $\hat{x}$ computed from the corrupted input $\tilde{x}$.
- Loss function compares $\hat{x}$ reconstruction with the noiseless $x$.

❖ The autoencoder cannot fully trust each feature of $x$ independently so it must learn the correlations of $x$'s features.
❖ Based on those relations we can predict a more ‘not prune to changes’ model.

➢ We are forcing the hidden layer to learn a generalized structure of the data.
DENOISING AUTOENCODERS - PROCESS

Taken some input $x$  
Apply Noise  
$\hat{x}$
DENOISING AUTOENCODERS - PROCESS

\[ \tilde{x} \xrightarrow{\text{Encode And Decode}} g(f(\tilde{x})) \]
DENOISING AUTOENCODERS - PROCESS

\[ g(f(\tilde{x})) \]
**DENOISING AUTOENCODERS - PROCESS**

\[ \hat{x} \quad \text{Compare} \quad x \]
DENOISING AUTOENCODERS

prior: examples concentrate near a lower dimensional “manifold”
DENOISING CONVOLUTIONAL AE — KERAS

- 50 epochs.
- Noise factor 0.5
- 92% accuracy on validation set.
Motivation:

- We want to harness the feature extraction quality of a AE for our advantage.
- For example: we can build a deep supervised classifier where it’s input is the output of a SAE.
- The benefit: our deep model’s W are not randomly initialized but are rather “smartly selected”
- Also using this unsupervised technique lets us have a larger unlabeled dataset.
- Building a SAE consists of two phases:
  1. Train each AE layer one after the other.
  2. Connect any classifier (SVM / FC NN layer etc.)
STACKED AE

$x$ → SAE → Classifier → $y$
STACKED AE — TRAIN PROCESS

First Layer Training (AE 1)

\[
x \xrightarrow{f_1(x)} z_1 \xrightarrow{g_1(z_1)} \hat{x}
\]
STACKED AE — TRAIN PROCESS

Second Layer Training (AE 2)

\[ x \rightarrow f_1(x) \rightarrow z_1 \rightarrow f_2(z_1) \rightarrow z_2 \rightarrow g_2(z_2) \rightarrow \hat{z}_1 \]
STACKED AE — TRAIN PROCESS

Add any classifier

\[ x \xrightarrow{f_1(x)} z_1 \xrightarrow{f_2(z_1)} z_2 \xrightarrow{\text{Classifier}} \text{Output} \]
SegNet Architecture

(Convolutional Encoder-Decoder)

Pooling Indices

- Conv + Batch Normalisation + ReLU
- Pooling
- Upsampling
- Softmax

(Badrinarayanan, Kendall, & Cipolla, 2015)
SegNet Evaluation

• SUN RGB-D Results – complex indoor scenes
• CamVid - out-door road scenes
• Pascal VOC 2012 – few classes, varying backgrounds

• Demo

https://mi.eng.cam.ac.uk/projects/segnet/#demo
SegNet CamVid Results

SegNet-Basic with only local contrast normalized RGB as input (median freq. balancing)

SegNet with only local contrast normalized RGB as input (pre-trained encoder, median freq. balancing)

SegNet with only local contrast normalized RGB as input (pretrained encoder, median freq. balancing + large training set)

(Badrinarayanan, Kendall, & Cipolla, 2015)
SegNet SUN RGB-D Results

(Badrinarayanan, Kendall, & Cipolla, 2015)
U-Net

- Current state of the art
  - Very popular in MICCAI 2016
  - Works well with low data
- Influenced by the previous
  - Up-conv 2x2 = bilinear up-sampling then 2x2 convolution
  - 2D slices
  - 3x3 convolutions

Fig. 1. U-net architecture (example for 32x32 pixels in the lowest resolution). Each blue box corresponds to a multi-channel feature map. The number of channels is denoted on top of the box. The x-y-size is provided at the lower left edge of the box. White boxes represent copied feature maps. The arrows denote the different operations.

Fig. 2. Overlap-tile strategy for seamless segmentation of arbitrary large images (here segmentation of neuronal structures in EM stacks). Prediction of the segmentation in the yellow area, requires image data within the blue area as input. Missing input data is extrapolated by mirroring.

https://lmb.informatik.uni-freiburg.de/people/ronneber/u-net/
Credits

• This lecture contains public domain materials from
  – Guy Nolan
  – Stefania Raimondo
  – Jason Su
  – Hung-yi Lee