Topics in AI (CPSC 532S): Multimodal Learning with Vision, Language and Sound

Lecture 13: Autoencoders [part 2]
Assignment 4 due Monday, March 8th

Group formation — tomorrow

Project proposals — Monday, March 15th

Assignment 5 will be out, Tuesday, March 16th
— Will have relatively long timeline, so you balance with the project
Unsupervised Learning

We have access to \( \{x_1, x_2, x_3, \cdots, x_N\} \) but not \( \{y_1, y_2, y_3, \cdots, y_N\} \)

*slide from Louis-Philippe Morency*
Restricted Boltzmann Machines (in one slide)

Model the **joint probability** of hidden state and observation

\[
p(x, h; \theta) = \frac{\exp(-E(x, h; \theta))}{Z}
\]

\[
Z = \sum_x \sum_h \exp(-E(x, h; \theta))
\]

\[
E = -xW h - b^T x - a^T h
\]

\[
E = -\sum_i \sum_j w_{i,j} x_i h_j - \sum_i b_i x_i - \sum_j a_j h_j
\]

Objective, maximize likelihood of the data

*slide from Louis-Philippe Morency*
Autoencoders
Autoencoders

Self (i.e. self-encoding)

*slide from Louis-Philippe Morency
Autoencoders

Self (i.e. self-encoding)

— Feed forward network intended to reproduce the input

— Encoder/Decoder architecture
  Encoder: $f = \sigma(Wx)$
  Decoder: $g = \sigma(W'h)$
Autoencoders

Self (i.e. self-encoding)

— Feed forward network intended to reproduce the input
— Encoder/Decoder architecture
  Encoder: $f = \sigma(Wx)$
  Decoder: $g = \sigma(W'h)$
— Score function

\[ x' = f(g(x)) \]
\[ \mathcal{L}(x', x) \]
Autoencoders

A standard neural network architecture (linear layer followed by non-linearity)

— Activation depends on type of data
  (e.g., sigmoid for binary; linear for real valued)

— Often use tied weights

\[ W' = W \]
Autoencoders: Hidden Layer Dimensionality

**Smaller** than the input

- Will compress the data, reconstruction of the data far from the training distribution will be difficult

- Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)
Autoencoders: Hidden Layer Dimensionality

**Smaller** than the input

— Will compress the data, reconstruction of the data far from the training distribution will be difficult

— Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)

Side note, this is useful for **anomaly detection**

*slide from Louis-Philippe Morency*
**Autoencoders**: Hidden Layer Dimensionality

**Smaller** than the input

- Will compress the data, reconstruction of the data far from the training distribution will be difficult
- Linear-linear encoder-decoder with Euclidian loss is actually equivalent to PCA (under certain data normalization)

**Larger** than the input

- No compression needed
- Can trivially learn to just copy, no structure is learned (unless you regularize)
- Does not encourage learning of meaningful features (unless you regularize)

*slide from Louis-Philippe Morency*
Autoencoders

A standard neural network architecture (linear layer followed by non-linearity)

— Activation depends on type of data
  (e.g., sigmoid for binary; linear for real valued)

— Often use tied weights

\[ W' = W \]
**De-noising Autoencoder**

**Idea:** add noise to input but learn to reconstruct the original

- Leads to better representations
- Prevents copying

**Note:** different noise is added during each epoch
What **can we do** with them?

- Good for compression (better than PCA)
- Disregard the decoder and use the middle layer as a representation
- Fine-tune the autoencoder for a task
Stacked (deep) Autoencoders and Denoising Autoencoders

What can we do with them?

— Good for compression (better than PCA)
— Disregard the decoder and use the middle layer as a representation
— **Fine-tune** the autoencoder for a task

*slide from Louis-Philippe Morency*
Context Encoders

[Pathak et al., 2016]
Context Encoders

[Pathak et al., 2016]
Context Encoders

[Pathak et al., 2016]
Context Encoders

[Pathak et al., 2016]
Context Encoders

[Pathak et al., 2016]
## Context Encoders

*Pathak et al., 2016*

<table>
<thead>
<tr>
<th>Pretraining Method</th>
<th>Supervision</th>
<th>Pretraining time</th>
<th>Classification</th>
<th>Detection</th>
<th>Segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ImageNet [26]</td>
<td>1000 class labels</td>
<td>3 days</td>
<td>78.2%</td>
<td>56.8%</td>
<td>48.0%</td>
</tr>
<tr>
<td>Random Gaussian</td>
<td>initialization</td>
<td>&lt; 1 minute</td>
<td>53.3%</td>
<td>43.4%</td>
<td>19.8%</td>
</tr>
<tr>
<td>Autoencoder</td>
<td>-</td>
<td>14 hours</td>
<td>53.8%</td>
<td>41.9%</td>
<td>25.2%</td>
</tr>
<tr>
<td>Agrawal <em>et al.</em> [1]</td>
<td>egomotion</td>
<td>10 hours</td>
<td>52.9%</td>
<td>41.8%</td>
<td>-</td>
</tr>
<tr>
<td>Doersch <em>et al.</em> [7]</td>
<td>context</td>
<td>4 weeks</td>
<td>55.3%</td>
<td>46.6%</td>
<td>-</td>
</tr>
<tr>
<td>Wang <em>et al.</em> [39]</td>
<td>motion</td>
<td>1 week</td>
<td>58.4%</td>
<td>44.0%</td>
<td>-</td>
</tr>
<tr>
<td>Ours</td>
<td>context</td>
<td>14 hours</td>
<td>56.5%</td>
<td>44.5%</td>
<td>29.7%</td>
</tr>
</tbody>
</table>
Spatial Context Networks

[ Wu, Sigal, Davis, 2017 ]
Spatial Context Networks

- Initiation
  - Random Gaussian
  - Wang et al. [32]
  - Doersch et al. [3]
  - *Doersch et al. [3]
  - Pathak et al. [21]
  - Zhang et al. [36]
  - ImageNet [21]
  - *ImageNet
  - SCN-EdgeBox

- Supervision
  - Random
  - Motion
  - Context
  - 1000 class labels
  - Context inpainting
  - Color
  - 1000 class labels

- Pretraining Time
  - < 1 minute
  - 1 week
  - 4 weeks
  - –
  - 14 hours
  - –
  - 3 days
  - 3 days
  - 10 hours

- Classification
  - 53.3
  - 58.4
  - 55.3
  - 65.4
  - 56.5
  - 65.6
  - 78.2
  - 76.9
  - 79.0

- Detection
  - 43.4
  - 44.0
  - 46.6
  - 50.4
  - 44.5
  - 46.9
  - 56.8
  - 58.7
  - 59.4

[ Wu, Sigal, Davis, 2017 ]
A Little Theory: Information Bottleneck [Tishbi et al., 1999]

Every layer could be treated as a random variable, then entire network is a Markov Chain

Data processing theorem: if the only connection between X and Z is through Y, the information that Z gives about X cannot be bigger than the information that Y gives about X.

\[ I(X;Y) \leq I(T_1;Y) \leq I(T_2;Y) \leq \cdots \leq I(\hat{Y};Y) \]
**A Little Theory: Information Bottleneck** [ Tishbi et al., 1999 ]

**Observation:** In the information plane layers first increase the mutual information between themselves and the output and then reduce information between themselves and the input (which leads to “forgetting” of irrelevant inputs and ultimately generalization)
A Little Theory: **Information Bottleneck** [Tishbi et al., 1999]

50 networks of same topology being optimized
A Little Theory: **Information Bottleneck** [Tishbi et al., 1999]

**Limitation:** Does not seem to work for non-Tanh activations (e.g., ReLU)