Deep Learning: Introduction

Vibhav Gogate
The University of Texas at Dallas
Motivation

• Look at our brains
• Many levels of processing
• Each level is learning features or representations at increasing level of abstraction
• Example: Model of visual cortex
  – Our brain first identifies edges, then patches, surfaces, objects, etc. in that order
• Replicate this architecture = Deep Learning
Applications

• Computer Vision
• Speech
• Natural Language processing
• Video Processing (New???)
• Your own app here!
Deep models: Plenty of them

• Classification (does not work well)
  – Deep Multi-layer perceptrons

• Graphical models
  – Deep Directed networks (BNs)
  – Deep Boltzmann Machines (Markov networks)
  – Deep Belief Networks (BN+MN combination)

• Deep Auto-encoders

• Deep AND/OR graphs or Sum-product networks
Deep Directed Models

All CPDs are logistic (Sigmoid networks)

Inference is hard: Why?
– Correlated by explaining away

\[ p(h_1, h_2, h_3, v | \theta) = \prod_i \text{Ber}(v_i | \text{sigm}(h_1^T w_{0i})) \prod_j \text{Ber}(h_{1j} | \text{sigm}(h_2^T w_{1j})) \]
\[ \prod_k \text{Ber}(h_{2k} | \text{sigm}(h_3^T w_{2k})) \prod_l \text{Ber}(h_{3l} | w_{3l}) \]

Slow inference; Slow learning
Deep Boltzmann machines

- Inference is easier
  - Block Gibbs sampling
- Learning is harder
  - Partition function
- Greedy layer by layer learning works well in practice

\[
p(h_1, h_2, h_3, v | \theta) = \frac{1}{Z(\theta)} \exp \left( \sum_{ij} v_i h_{1j} W_{1ij} + \sum_{jk} h_{1j} h_{2j} W_{2jk} + \sum_{kl} h_{2k} h_{3l} W_{3kl} \right)
\]
Deep Belief networks

- Learning is slightly easier than Deep Boltzmann machines
- Inference is slightly harder

\[
p(h_1, h_2, h_3, v|\theta) = \prod_i \text{Ber}(v_i|\text{sigm}(h_1^T w_{1i}) \prod_j \text{Ber}(h_{1j}|\text{sigm}(h_2^T w_{2j}) \right. \\
\left. \frac{1}{Z(\theta)} \exp \left( \sum_{kl} h_{2k} h_{3l} W_{3kl} \right) \right)
\]
Deep Auto-Encoder

• Auto-encoder: A neural network used for dimensionality reduction and feature discovery
  – Trained to predict itself!

• Hidden layer is constrained to be a narrow bottleneck
  – Otherwise, it will learn trivial identity mapping

• $V \rightarrow H \rightarrow V$
Training a deep autoencoder. (a) First we greedily train some RBMs. (b) Then we construct the auto-encoder by replicating the weights. (c) Finally we fine-tune the weights using back-propagation.
Sum-Product Networks: Motivation

• Deep learning: Stack many layers
  E.g.: DBN [Hinton & Salakhutdinov, 2006]
  CDBN [Lee et al., 2009]
  DBM [Salakhutdinov & Hinton, 2010]

• Potentially much more powerful than shallow architectures [Bengio, 2009]

• But ...
  – Inference is even harder
  – Learning requires extensive effort
Sum-Product Networks: Deep Tractable Models

Probability: \( P(X) = \frac{S(X)}{Z} \)

\( X: X_1 = 1, X_2 = 0 \)

<table>
<thead>
<tr>
<th></th>
<th>( X_1 )</th>
<th>( \overline{X}_1 )</th>
<th>( X_2 )</th>
<th>( \overline{X}_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( X_1 )</td>
<td>1</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( \overline{X}_1 )</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( X_2 )</td>
<td></td>
<td></td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>( \overline{X}_2 )</td>
<td></td>
<td></td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Applications: Handwritten Digit Classification using DBNs

- 3 layers
- Top layer is connected to 10 label units representing 10 digits (Hack!)
Applications: Handwritten Digit Classification using DBNs

- First two hidden layers were trained in a greedy unsupervised fashion using 30 passes over the data
  - Few hours
- Top layer trained using as input activations of the lower hidden layer as well as the class labels
- Weights Fine tuned using top-down procedure

Errors
DBN: 1.25%
SVM with degree 9 poly kernel: 1.4%
1-nearest neighbor: 3.1%
Applications: Handwritten Digit Classification using DBNs
Applications: **Data visualization and feature discovery**

- Learn informative features from raw data
- Use these features as input to supervised learning algorithms

2d visualization of some bag of words data from the Reuters RCV1-v2 corpus. Results of using a deep auto-encoder.
Applications: Learning Audio Features and Image Features

A 2d convolutional RBM with max-pooling layers. The input signal is a stack of 2d images (e.g., color planes). Each input layer is passed through a different set of filters. Each hidden unit is obtained by convolving with the appropriate filter, and then summing over the input planes. The final layer is obtained by computing the local maximum within a small window.
Applications: Learning Audio Features and Image Features

faces, cars, airplanes, motorbikes

Filters learned in layers 2 and 3
Applications: Image Completion
Sum-Product networks

Original

SPN

DBM

DBN

PCA

Nearest Neighbor
What we will cover?

• Architectures
  – How many hidden layers to use?
  – How many nodes in each hidden layer?
  – Etc.

• Learning Algorithms
• Inference Algorithms
• Applications