Deep Learning Overview

- Fall 2016
- Yoonsuck Choe

What Is Deep Learning?
- Learning higher level abstractions/representations from data.
- Motivation: how the brain represents and processes sensory information in a hierarchical manner.

![Brain diagram](image)

From LeCun’s Deep Learning Tutorial

Intro to Neural Network: Backpropagation

Weight $w_{ji}$ is updated as: $w_{ji} \leftarrow w_{ji} + \eta \delta_j a_i$, where

- $a_i$: activity at input side of weight $w_{ji}$.
- $T_k$ is target value.

$\delta_k = (T_k - a_k) \sigma'(net_k)$

- Deeper weights (green line in figure above).

$$\delta_j = \sum_k w_{kj} \delta_k \sigma'(net_j)$$
Deep Learning

- Complex models with large number of parameters
  - Hierarchical representations
  - More parameters = more accurate on training data
  - Simple learning rule for training (gradient-based).

- Lots of data
  - Needed to get better generalization performance.
  - High-dimensional input need exponentially many inputs (curse of dimensionality).

- Lots of computing power: GPGPU, etc.
  - Training large networks can be time consuming.

The Rise of Deep Learning

- Andrew Ng & Jeff Dean (Google Brain team, 2012).
- Schmidhuber et al.'s deep neural networks (won many competitions and in some cases showed super human performance; 2011–). Recurrent neural networks using LSTM (Long Short-Term Memory).

Long History (in Hind Sight)

- Fukushima's Neocognitron (1980).
History: Fukushima’s Neocognitron

- Appeared in journal *Biological Cybernetics* (1980).
- Multiple layers with local receptive fields.
- S cells (trainable) and C cells (fixed weight).
- Deformation-resistant recognition.

History: LeCun’s Convolutional Neural Nets

- Convolution kernel (weight sharing) + Subsampling
- Fully connected layers near the end.
- Became a main-stream method in deep learning.

Motivating Deep Learning: Tensorflow Demo

- [http://playground.tensorflow.org](http://playground.tensorflow.org)
- Demo to explore why deep nnet is powerful and how it is limited.

Current Trends

- Deep belief networks (based on Boltzmann machine)
- Convolutional neural networks
- Deep Q-learning Network (extensions to reinforcement learning)
- Deep recurrent neural networks using (LSTM)
- Applications to diverse domains.
  - Vision, speech, video, NLP, etc.
- Lots of open source tools available.
Boltzmann Machine to Deep Belief Nets

• Haykin Chapter 11: Stochastic Methods rooted in statistical mechanics.

Boltzmann Machine: Energy

- Network state: $\mathbf{x}$ from random variable $\mathbf{X}$.
- $w_{ij} = w_{ji}$ and $w_{ii} = 0$.
- Energy (in analogy to thermodynamics):

$$E(\mathbf{x}) = -\frac{1}{2} \sum_i \sum_{j,i \neq j} w_{ji} x_i x_j$$

Boltzmann Machine: Prob. of a State $\mathbf{x}$

- Probability of a state $\mathbf{x}$ given $E(\mathbf{x})$ follows the Gibbs distribution:

$$P(\mathbf{X} = \mathbf{x}) = \frac{1}{Z} \exp \left( -\frac{E(\mathbf{x})}{T} \right)$$

- $Z$: partition function (normalization factor – hard to compute).
- $T$: temperature parameter.
- Low energy states are exponentially more probable.
- State $\mathbf{x}$ changed over time following the probability distribution $P(\mathbf{X} = \mathbf{x})$.  

• Stochastic binary machine: +1 or -1.
• Fully connected symmetric connections: $w_{ij} = w_{ji}$.
• Visible vs. hidden neurons, clamped vs. free-running.
• Goal: Learn weights to model prob. dist of visible units.
• Unsupervised. Pattern completion.
Boltzmann Learning Rule

- Learning based on correlation $\rho_{ji}^+$ (clamped) and $\rho_{ji}^-$ (free-running).

\[
\Delta w_{ji} = \eta \frac{\partial L(w)}{\partial w_{ji}} = \eta \left( \rho_{ji}^+ - \rho_{ji}^- \right)
\]

where $L(w)$ is the log likelihood of the pattern being any of the training patterns, and $\eta$ is the learning rate. This is gradient ascent.

Boltzmann Machine Summary

- Theoretically elegant.
- Very slow in practice (especially the unclamped phase).

Logistic (or Directed) Belief Net

- Similar to Boltzmann Machine, but with directed, acyclic connections.

\[
P(X_j = x_j | X_1 = x_1, ..., X_{j-1} = x_{j-1}) = P(X_j = x_j | \text{parents}(X_j))
\]

- Same learning rule:

\[
\Delta w_{ji} = \eta \frac{\partial L(w)}{\partial w_{ji}}
\]

- With dense connections, calculation of $P$ becomes intractable.

Deep Belief Net (1)

- Overcomes issues with Logistic Belief Net. Hinton et al. (2006)
- Based on Restricted Boltzmann Machine (RBM): visible and hidden layers, with layer-to-layer full connection but no within-layer connections.
- RBM Back-and-forth update: update hidden given visible, then update visible given hidden, etc., then train $w$ based on

\[
\frac{\partial L(w)}{\partial w_{ji}} = \rho_{ji}^{(0)} - \rho_{ji}^{(\infty)}
\]
Deep Belief Net (2)

Deep Belief Net = Layer-by-layer training using RBM.
Hybrid architecture: Top layer = undirected, lower layers directed.

1. Train RBM based on input to form hidden representation.
2. Use hidden representation as input to train another RBM.
3. Repeat steps 2-3.

* Similar approach: Stacked denoising autoencoders.

Applications: NIST digit recognition, etc.

Deep Convolutional Neural Networks (2)

- Learned kernels (first convolutional layer).
- Resembles mammalian RFs: oriented Gabor patterns, color opponency (red-green, blue-yellow).

Deep Convolutional Neural Networks (1)

- Krizhevsky et al. (2012)
- Applied to ImageNet competition (1.2 million images, 1,000 classes).
- Network: 60 million parameters and 650,000 neurons.
- Top-1 and top-5 error rates of 37.5% and 17.0%.
- Trained with backprop.

Deep Convolutional Neural Networks (3)

- Left: Hits and misses and close calls.
- Right: Test (1st column) vs. training images with closest hidden representation to the test data.
Deep Q-Network (DQN)


- Latest application of deep learning to a *reinforcement learning* domain (*Q* as in *Q*-learning).
- Applied to *Atari 2600* video game playing.

DQN Overview

- Input preprocessing
- Experience replay (collect and replay state, action, reward, and resulting state)
- Delayed (periodic) update of *Q*.
- Moving target $\hat{Q}$ value used to compute error (loss function $L$, parameterized by weights $\theta_i$).
  - Gradient descent: $\frac{\partial L}{\partial \theta_i}$

DQN Algorithm

**Algorithm 1**: deep Q-learning with experience replay.

Initialize replay memory $D$ to capacity $N$.

Initialize action-value function $Q$ with random weights $\theta$.

Initialize target action-value function $\hat{Q}$ with weights $\theta^* = \theta$.

For episode $e = 1, M$ do

  1. Initialize sequence $s_1 = \{s_t\}$ and preprocessed sequence $\phi_1 = \phi(s_1)$.

  2. For $t = 1, T$ do

     a. With probability $\epsilon$ select a random action $a_t$.

     b. Otherwise, select $a_t = \text{argmax}_a Q(\phi(s_t); a; \theta)$.

     c. Execute action $a_t$ in emulator and observe reward $r_t$ and image $x_{t+1}$.

     d. Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$.

     e. Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $D$.

     f. Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $D$.

     g. Set $y_j = \begin{cases} r_j + \gamma \max_{a'} \hat{Q} (\phi_{j+1}; a'; \theta^*) & \text{if episode terminates at step } j + 1 \\ \hat{Q} (\phi_{j+1}; a_t; \theta) & \text{otherwise} \end{cases}$

     h. Perform a gradient descent step on $\left(y_j - Q(\phi_j, a_t; \theta)\right)^2$ with respect to the network parameters $\theta$.

    Every $C$ steps reset $\hat{Q} = Q$.

End For

End For
DQN Results

- Superhuman performance on over half of the games.

DQN Operation

- Value vs. game state; Game state vs. action value.

Deep Recurrent Neural Networks

Feedforward
- No memory of past input.

Recurrent
- Good: Past input affects present output.
- Bad: Cannot remember far into the past.
RNN Training: Backprop in Time

- Can unfold recurrent loop: Make it into a feedforward net.
- Use the same backprop algorithm for training.
- Again, cannot remember too far into the past.

Fig from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

- Captures info
- Keeps info
- Releases info
- gate is close
- gate is open

- Long-term retention possible with LSTM.

From http://www.machinelearning.ru/wiki/images/6/6c/RNN_and_LSTM_16102015.pdf

Long Short-Term Memory in Action

- Unfold in time and use backprop as usual.

Fig from http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Long Short-Term Memory

- LSTM to the rescue (Hochreiter and Schmidhuber, 2017).
- Built-in recurrent memory that can be written (Input gate), reset (Forget gate), and outputted (Output gate).

From http://www.machinelearning.ru/wiki/images/6/6c/RNN_and_LSTM_16102015.pdf
LSTM Applications

- Applications: Sequence classification, Sequence prediction, Sequence translation.

From http://machinelearning.ru

Deep Learning Applications: Vision

- ConvNet sweeping image recognition challenges.

From LeCun’s Deep Learning Tutorial
Deep Learning Applications: Speech

- Deep learning led to major improvement in speech recognition.
  
  From LeCun's Deep Learning Tutorial

Deep Learning Applications: NLP

- Based on encoding/decoding and attention.
  

Deep Learning Applications: Speech

- ConvNet applied to speech recognition.
- Use spectrogram and treat it like a 2D image.

From LeCun's Deep Learning Tutorial

Deep Learning Applications: NLP

- Google's LSTM-based machine translation.

Limitations

• Discriminative vs. generative learning.
  – Discriminative: $P(\text{class} | \text{data})$. Can easily be fooled with adversarial input.
  – Generative: $P(\text{class, data}) = P(\text{class} | \text{data})P(\text{data})$. Explicitly models the data.

• Deep neural nets mostly use discriminative learning, so can be fooled by adversarial input. Generative adversarial learning can overcome this (Goodfellow et al. arXiv:1406.2661 (2014)).

Summary


• Deep convolutional networks: High computational demand, over the board great performance.


• Deep recurrent neural networks: sequence learning. LSTM a powerful mechanism.

• Diverse applications. Top performance.

• Flood of deep learning tools available.

Deep Learning Tools

• Kaffe: UC Berkeley’s deep learning tool box
  
• TensorFlow (Google)
  
• Deep learning modules for Torch (Facebook)
  
• Microsoft CNTK (Computational Network Tool Kit)
  
• Other: Apache Mahout (MapReduce-based ML)