김지섭 (jkim@bi.snu.ac.kr)
서울대학교 컴퓨터공학부
2015/05/7 인공지능 실습수업
History of Neural Network Research

Neural network
Back propagation

1986

Deep belief net
Science

2006

The New York Times

Google

Hong Kong

Microsoft

IMAGENET

2011

2012

deep learning results

• Unsupervised & Layer-wise pre-training

<table>
<thead>
<tr>
<th>Rank</th>
<th>Name</th>
<th>Error rate</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>U. Toronto</td>
<td>0.15315</td>
<td>Deep Conv Net</td>
</tr>
<tr>
<td>2</td>
<td>U. Tokyo</td>
<td>0.26172</td>
<td>Hand-crafted features and learning models.</td>
</tr>
<tr>
<td>3</td>
<td>U. Oxford</td>
<td>0.26979</td>
<td>Bottleneck.</td>
</tr>
<tr>
<td>4</td>
<td>Xerox/INRIA</td>
<td>0.27058</td>
<td></td>
</tr>
</tbody>
</table>

Object recognition over 1,000,000 images and 1,000 categories (2 GPU)

Microsoft

Deep Networks Advance State of Art in Speech

Deep Learning leads to breakthrough in speech recognition at MSR.

Slides from Wanli Ouyang wlouyang@ee.cuhk.edu.hk
Contents

● Neural Networks
  ▪ Multilayer Perceptron 구조
  ▪ Back Propagation 학습 알고리즘

● Deep Belief Network
  ▪ Restricted Boltzmann Machines
  ▪ Deep Learning (Deep Belief Network)

● Convolutional Neural Networks (CNN)
  ▪ CNN 구조 및 학습
  ▪ 응용 사례
NEURAL NETWORKS

Slides by Jiseob Kim
jkim@bi.snu.ac.kr
Classification Problem

● 데이터 $x$가 주어졌을 때 해당되는 레이블 $y$를 찾는 문제
  ▪ ex1) $x$: 사람의 얼굴 이미지, $y$: 사람의 이름
  ▪ ex2) $x$: 혈당 수치, 혈압 수치, 심박수, $y$: 당뇨병 여부
  ▪ ex3) $x$: 사람의 목소리, $y$: 목소리에 해당하는 문장

● $x$: D차원 벡터, $y$: 정수 (Discrete)

● 대표적인 패턴 인식 알고리즘
  ▪ Support Vector Machine
  ▪ Decision Tree
  ▪ K-Nearest Neighbor
  ▪ Multi-Layer Perceptron (Artificial Neural Network; 인공신경망)
Artificial Neural Networks

The Perceptron

\[
\text{output} = \begin{cases} 
1 & \text{if } \sum_{i=0}^{n} w_i x_i > 0 \\
-1 & \text{otherwise}
\end{cases}
\]

\[
\sum_{i=0}^{n} w_i x_i = w \cdot x
\]
Perceptron (2/2)

\[ w_1 x_1 + w_2 x_2 + b = 0 \]

\[ \begin{cases} > 0: & \bigcirc \\ < 0: & \bigtriangleup \end{cases} \]
**Parameter Learning in Perceptron**

**start:**
The weight vector $w$ is generated randomly.

**test:**
A vector $x \in P \cup N$ is selected randomly,
If $x \in P$ and $w \cdot x > 0$ goto test,
If $x \in P$ and $w \cdot x \leq 0$ goto **add**, 
If $x \in N$ and $w \cdot x < 0$ go to test,
If $x \in N$ and $w \cdot x \geq 0$ go to **subtract**.

**add:**
Set $w = w + x$, goto test

**subtract:**
Set $w = w - x$, goto test

---

**Equation of hyperplane:**

$$X \cdot W - \theta = 0$$

- **$X \cdot W - \theta > 0$** on this side
- **$X \cdot W - \theta < 0$** on this side

**$\frac{W}{|W|}$** Unit vector normal to hyperplane
Sigmoid Unit

\[
\frac{\partial \sigma(x)}{\partial x} = \sigma(x)(1 - \sigma(x))
\]

Sigmoid function is Differentiable

\[
net = \sum_{i=0}^{n} w_i x_i
\]

\[
o = \sigma(net) = \frac{1}{1 + e^{-net}}
\]

Classic Perceptron

Sigmoid Unit
Learning Algorithm of Sigmoid Unit

- Loss Function
  \[ \varepsilon = (d - f)^2 \]

- Gradient Descent Update
  \[
  \frac{\partial \varepsilon}{\partial W} = -2(d - f) \frac{\partial f}{\partial s} X = -2(d - f) f(1 - f) X
  \]
  \[
  f(s) = \frac{1}{1 + e^{-s}}
  \]
  \[
  f'(s) = f(s)(1 - f(s))
  \]
  \[
  W \leftarrow W + c(d - f) f(1 - f) X
  \]
Need for Multiple Units and Multiple Layers

- Multiple boundaries are needed (e.g. XOR problem) → Multiple Units
- More complex regions are needed (e.g. Polygons) → Multiple Layers
Structure of Multilayer Perceptron (MLP; Artificial Neural Network)
• **Loss Function**
  - We have the same Loss Function
  - But the # of parameters are now much more (Weight for each layer and each unit)
  - To use Gradient Descent, we need to calculate the gradient for all the parameters

• **Recursive Computation of Gradients**
  - Computation of loss-gradient of the top-layer weights is the same as before
  - Using the chain rule, we can compute the loss-gradient of lower-layer weights recursively (Back Propagation)
Back Propagation Learning Algorithm (1/3)

- Gradients of top-layer weights and update rule
  \[ \varepsilon = (d - f)^2 \]
  \[ \frac{\partial \varepsilon}{\partial W} = -2(d - f) \frac{\partial f}{\partial s} X = -2(d - f)f(1 - f)X \]
  \[ W \leftarrow W + c(d - f)f(1 - f)X \]

- Store intermediate value \textbf{delta} for later use of chain rule
  \[ \delta^{(k)} = \frac{\partial \varepsilon}{\partial s^{(j)}} = (d - f) \frac{\partial f}{\partial s^{(j)}} \]
  \[ = (d - f)f(1 - f) \]
Back Propagation Learning Algorithm (2/3)

- **Gradients of lower-layer weights**

  \[
  \frac{\partial \varepsilon}{\partial W_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} \frac{\partial s_i^{(j)}}{\partial W_i^{(j)}} = \frac{\partial \varepsilon}{\partial s_i^{(j)}} X^{(j-1)}
  \]

  \[
  = -2(d - f) \frac{\partial f}{\partial s_i^{(j)}} X^{(j-1)} = -2\delta_i^{(j)} X^{(j-1)}
  \]

  **Weighted sum**

  \[
  S_i^{(j)} = X^{(j-1)} \cdot W_i^{(j)}
  \]

  **Local gradient**

  \[
  \frac{\partial \varepsilon}{\partial s_i^{(j)}} = \frac{\partial (d - f)^2}{\partial s_i^{(j)}} = -2(d - f) \frac{\partial f}{\partial s_i^{(j)}}
  \]

  **Gradient Descent Update rule for lower-layer weights**

  \[
  W_i^{(j)} \leftarrow W_i^{(j)} + \eta \delta_i^{(j)} X^{(j-1)}
  \]
Applying chain rule, recursive relation between delta’s:

\[ \delta_i^{(j)} = f_i^{(j)}(1 - f_i^{(j)}) \sum_{l=1}^{m_{j+1}} \delta_i^{(j+1)} w_{il} \]

Algorithm: Back Propagation

1. Randomly Initialize weight parameters
2. Calculate the activations of all units (with input data)
3. Calculate top-layer delta
4. Back-propagate delta from top to the bottom
5. Calculate actual gradient of all units using delta’s
6. Update weights using Gradient Descent rule
7. Repeat 2~6 until converge
Applications

- Almost All Classification Problems
  - Face Recognition
  - Object Recognition
  - Voice Recognition
  - Spam mail Detection
  - Disease Detection
  - etc.
Limitations and Breakthrough

● Limitations
  ▪ Back Propagation barely changes lower-layer parameters (Vanishing Gradient)
  ▪ Therefore, Deep Networks cannot be fully (effectively) trained with Back Propagation

● Breakthrough
  ▪ Deep Belief Networks (Unsupervised Pre-training)
  ▪ Convolutional Neural Networks (Reducing Redundant Parameters)
  ▪ Rectified Linear Unit (Constant Gradient Propagation)
DEEP BELIEF NETWORKS

Slides by Jiseob Kim
jkim@bi.snu.ac.kr
Motivation

- 아이디어:
  - Greedy Layer-wise training
  - Pre-training + Fine tuning
  - Contrastive Divergence
Restricted Boltzmann Machine (RBM)

Energy-Based Model

\[ P(x, h) = \frac{e^{-E(x, h)}}{\sum_{x, h} e^{-E(x, h)}} \]
\[ P(x) = \frac{\sum_{h} e^{-E(x, h)}}{\sum_{x, h} e^{-E(x, h)}} \]

Joint \( (x, h) \) Probability
Marginal \( (x) \) Probability, or Likelihood

\[ P(x_j = 1|h) = \sigma(b_j + W' \cdot h) \]
\[ P(h_i = 1|x) = \sigma(c_i + W_i \cdot x) \]

Energy function

- \( E(x, h) = b' \cdot x + c' \cdot h + h' \cdot W \cdot x \)

Remark:
- Conditional Independence

\[ P(h \mid x) = \prod_i P(h_i \mid x) \]
\[ P(x \mid h) = \prod_j P(x_j \mid h) \]

- Conditional Probability is the same as Neural Network
Unsupervised Learning of RBM

- **Maximum Likelihood**
  - Use Gradient Descent

\[
L(X; \theta) = \sum_{x,h} e^{-E(x,h)} \sum_{x,h} e^{-E(x,h)}
\]

\[
\frac{\partial L(X; \theta)}{\partial w_{ij}} = \int p(x, \theta) \frac{\partial \log f(x; \theta)}{\partial \theta} dx - \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \log f(x^{(k)}; \theta)}{\partial \theta}
\]

\[
= \langle x_i h_j \rangle_{p(x, \theta)} - \langle x_i h_j \rangle_X = \langle x_i h_j \rangle_{\infty} - \langle x_i h_j \rangle_{0}
\]

\[
\approx \langle x_i h_j \rangle_{1} - \langle x_i h_j \rangle_{0}
\]

- Distribution of Dataset
- Distribution of Model

\[
\langle v_i h_j \rangle^0, \langle v_i h_j \rangle^1, \langle v_i h_j \rangle^2, \ldots \langle v_i h_j \rangle^\infty
\]

a fantasy
Contrastive Divergence (CD) Learning of RBM parameters

- **k-Contrastive Divergence Trick**
  - From the previous slide, to get distribution of model, we need to calculate many Gibbs sampling steps.
  - And this is per a single parameter update.
  - Therefore, we take the sample after only k-steps where in practice, $k=1$ is sufficient.

Take this as a sample of Model distribution.
Effect of Unsupervised Training

Unsupervised Training makes RBM successfully catch the essential patterns

RBM trained on MNIST hand-written digit data:

Each cell shows the pattern each hidden node encodes
Deep Belief Network (DBN)

- **Deep Belief Network (Deep Bayesian Network)**
  - Bayesian Network that has similar structure to Neural Network
  - Generative model
  - Also, can be used as classifier (with additional classifier at top layer)
  - Resolves gradient vanishing by Pre-training
  - There are two modes (Classifier & Auto-Encoder), but we only consider Classifier here
Learning Algorithm of DBN

- **DBN as a stack of RBMs**

1. Regard each layer as RBM
2. Layer-wise Pre-train each RBM in Unsupervised way
3. Attach the classifier and Fine-tune the whole Network in Supervised way
Viewing Learning as Wake-Sleep Algorithm

Training algorithms of DBNs and RBMs

- Training data: training set size: $n$ (in millions), training data dimension: $d$ (= number of observable nodes, in hundreds or thousands).
- RBM training as the basic module: maximum likelihood + stochastic gradient ascent
- Layer-wise greedy training: $P(h^2, h^3) \sim \text{RBM}$

Joint Wake-Sleep algorithm: Samplings+ Updates

Wake Sampling | Sleep Sampling | Sleep Updates | Wake Updates
Effect of Unsupervised Pre-Training in DBN (1/2)

Erhan et. al. AISTATS’2009
Effect of Unsupervised Pre-Training in DBN (2/2)

without pre-training

with pre-training

![Graph showing the effect of pre-training on test classification error for different numbers of layers.](image)
Internal Representation of DBN
Higher layers have more abstract representations

- Interpolating between different images is not desirable in lower layers, but natural in higher layers

(a) Interpolating between an example and its 200-th nearest neighbor (see caption below).

(b) Interpolating between an example and its nearest neighbor.

(c) Sequences of points interpolated at different depths

Bengio et al., ICML 2013
As DBN is a generative model, we can also regenerate the data:
- From the top layer to the bottom, conduct Gibbs sampling to generate the data samples.

Lee, Ng et al., ICML 2009
Nowadays, CNN outperforms DBN for Image or Speech data

However, if there is no topological information, DBN is still a good choice

Also, if the generative model is needed, DBN is used

Generate Face patches
Tang, Srivastava, Salakhutdinov, NIPS 2014
Motivation

**Idea:**
- Fully connected 네트워크 구조는 학습해야할 파라미터 수가 너무 많음
- 이미지 데이터, 음성 데이터 (spectrogram)과 같이 각 feature들 간의 위상적, 기하적 구조가 있는 경우 Local한 패턴을 학습하는 것이 효과적

![Image 1](image1.png) ![Image 2](image2.png)

- DBN의 경우 다른 data
- CNN의 경우 같은 data
● Convolution과 Pooling (Subsampling)을 반복하여 상위 Feature를 구성
● Convolution은 Local 영역에서의 특정 Feature를 얻는 과정
● Pooling은 Dimension을 줄이면서도, Translation-invariant한 Feature를 얻는 과정

http://parse.ele.tue.nl/education/cluster2
Convolution Layer

- The Kernel Detects pattern:

- The Resulting value Indicates:
  - How much the pattern matches at each region
Max-Pooling Layer

- The Pooling Layer summarizes the results of Convolution Layer
  - e.g.) 10x10 result is summarized into 1 cell

- The Result of Pooling Layer is Translation-invariant
• Higher layer catches more specific, abstract patterns

• Lower layer catches more general patterns
Parameter Learning of CNN

- CNN is just another Neural Network with sparse connections
- Learning Algorithm:
  - Back Propagation on Convolution Layers and Fully-Connected Layers
Image Net Competition Ranking
(1000-class, 1 million images)

1. Clarifi (0.117): Deep Convolutional Neural Networks (Zeiler)
2. NUS: Deep Convolutional Neural Networks
3. ZF: Deep Convolutional Neural Networks
4. Andrew Howard: Deep Convolutional Neural Networks
5. OverFeat: Deep Convolutional Neural Networks
6. UvA-Euvision: Deep Convolutional Neural Networks
7. Adobe: Deep Convolutional Neural Networks
8. VGG: Deep Convolutional Neural Networks
10. decaf: Deep Convolutional Neural Networks
11. IBM Multimedia Team: Deep Convolutional Neural Networks
12. Deep Punx (0.209): Deep Convolutional Neural Networks
13. MIL (0.244): Local image descriptors + FV + linear classifier (Hidaka et al.)
14. Minerva-MSRA: Deep Convolutional Neural Networks

From Kyunghyun Cho’s dnn tutorial
Krizhevsky et al.: the winner of ImageNet 2012 Competition

1000-class problem, top-5 test error rate of 15.3%
Application (Speech Recognition)

Input: Spectrogram of Speech

Convolutional Neural Network

CNN outperforms all previous methods that uses GMM of MFCC
Good learning resources

- **Webpages:**
  - Geoffrey E. Hinton’s readings (with source code available for DBN) http://www.cs.toronto.edu/~hinton/csc2515/deeprefs.html
  - Notes on Deep Belief Networks http://www.quantumg.net/dbns.php
  - MLSS Tutorial, October 2010, ANU Canberra, Marcus Frean http://videolectures.net/mlss2010au_freen_deepbeliefnets/
  - Deep Learning Tutorials http://deeplearning.net/tutorial/
  - Hinton’s Tutorial, http://videolectures.net/mlss09uk_hinton_dbn/
  - CUHK MMLab project : http://mmlab.ie.cuhk.edu.hk/project_deep_learning.html

- **People:**
  - Geoffrey E. Hinton’s http://www.cs.toronto.edu/~hinton
  - Andrew Ng http://www.cs.stanford.edu/people/ang/index.html
  - Ruslan Salakhutdinov http://www.utstat.toronto.edu/~rsalakhu/
  - Yee-Whye Teh http://www.gatsby.ucl.ac.uk/~ywteh/
  - Yoshua Bengio www.iro.umontreal.ca/~bengioy
  - Yann LeCun http://yann.lecun.com/
  - Marcus Frean http://ecs.victoria.ac.nz/Main/MarcusFrean
  - Rob Fergus http://cs.nyu.edu/~fergus/pmwiki/pmwiki.php

- **Acknowledgement**
  - Many materials in this ppt are from these papers, tutorials, etc (especially Hinton and Frean’s). Sorry for not listing them in full detail.
Graphical model for Statistics

- **Conditional independence** between random variables
- Given C, A and B are independent:
  - $P(A, B|C) = P(A|C)P(B|C)$
- $P(A, B, C) = P(A, B|C)P(C) = P(A|C)P(B|C)P(C)$
- Any two nodes are conditionally independent given the values of their parents.

http://www.eecs.qmul.ac.uk/~norman/BBNs/Independence_and_conditional_independence.htm
Directed and undirected graphical models

- **Directed graphical model**
  - $P(A,B,C) = P(A|C)P(B|C)P(C)$
  - Any two nodes are *conditionally independent* given the values of their parents.

- **Undirected graphical model**
  - $P(A,B,C) = P(B,C)P(A,C)$
  - Also called Markov Random Field (MRF)

\[ P(A,B,C,D) = P(D|A,B)P(B|C)P(A|C)P(C) \]
Modeling undirected model

● Probability:

\[
P(x; \theta) = \frac{f(x; \theta)}{\sum_x f(x; \theta)} = \frac{f(x; \theta)}{Z(\theta)}
\]

Example: \( P(A, B, C) = P(B, C)P(A, C) \)

\[
P(A, B, C; \theta) = \frac{\exp(w_1 BC + w_2 AC)}{\sum_{A,B,C} \exp(w_1 BC + w_2 AC)}
\]

\[
= \frac{\exp(w_1 BC) \exp(w_2 AC)}{Z(w_1, w_2)}
\]

\[
\sum_x P(x; \theta) = 1
\]
More directed and undirected models

Hidden Markov model

MRF in 2D
More directed and undirected models

\[ P(A, B, C, D) = P(A)P(B)P(C|B)P(D|A, B, C) \]

\[ P(y_1, y_2, y_3, h_1, h_2, h_3) = P(h_1)P(h_2| h_1)P(h_3| h_2)P(y_1| h_1)P(y_2| h_2)P(y_3| h_3) \]
More directed and undirected models

(a) HMM
(b) RBM
(c) DBN
Extended reading on graphical model

- Zoubin Ghahramani’s video lecture on graphical models:
  - [http://videolectures.net/mlss07_ghahramani_grafm/](http://videolectures.net/mlss07_ghahramani_grafm/)
Product of Experts

\[
P(x; \theta) = \frac{\prod_m f_m(x_m; \theta_m)}{\sum_x \prod_m f_m(x_m; \theta_m)} = \frac{e^{-E(x; \theta)}}{\sum_x e^{-E(x; \theta)}} = \frac{f(x; \theta)}{Z(\theta)},
\]

Energy function

\[
E(x; \theta) = -\sum_m \log f_m(x_m; \theta_m)
\]

Partition function

\[
E(x; w) = w_1 AB + w_2 BC + w_3 AD + w_4 BE + w_3 CF + ...
\]

MRF in 2D
Product of Experts

\[ \prod_{i=1}^{15} \left[ \lambda_i e^{(x-u_i)^T \Sigma (x-u_i)} + c(1 - \lambda_i) \right] \]


**Products of experts versus Mixture model**

- **Products of experts:**
  - "and" operation
  - Sharper than mixture
  - Each expert can constrain a different subset of dimensions.

- **Mixture model, e.g. Gaussian Mixture model**
  - "or" operation
  - A weighted sum of many density functions

\[
P(x; \theta) = \frac{\prod_{m} f_m(x_m; \theta_m)}{\sum_{x} \prod_{m} f_m(x_m; \theta_m)}
\]
Outline

• Basic background on statistical learning and Graphical model

• Contrastive divergence and Restricted Boltzmann machine
  ▪ Product of experts
  ▪ Contrastive divergence
  ▪ Restricted Boltzmann Machine

• Deep belief net
**Contrastive Divergence (CD)**

- **Probability:**
- **Maximum Likelihood and gradient descent**

$$\max_{\theta} \left\{ \prod_{k=1}^{K} P(x^{(k)}; \theta) \right\} \iff \max_{\theta} L(X; \theta) = \max_{\theta} \left\{ \log \prod_{k=1}^{K} P(x^{(k)}; \theta) \right\}$$

$$\theta_{t+1} = \theta_t + \lambda \frac{\partial L(X; \theta)}{\partial \theta} \quad \text{or} \quad \frac{\partial L(X; \theta)}{\partial \theta} = 0$$

$$\frac{1}{K} \frac{\partial L(X; \theta)}{\partial \theta} = \frac{\partial}{\partial \theta} \left\{ \log Z(\theta) - \frac{1}{K} \sum_{k=1}^{K} \log f(x^{(k)}; \theta) \right\}$$

$$= \int p(x, \theta) \frac{\partial \log f(x; \theta)}{\partial \theta} dx - \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \log f(x^{(k)}; \theta)}{\partial \theta}$$

$$= \left\langle \frac{\partial \log f(x; \theta)}{\partial \theta} \right\rangle_{p(x, \theta)} - \left\langle \frac{\partial \log f(x; \theta)}{\partial \theta} \right\rangle_x$$

**Model dist.** 
**Data dist.** 
**Expectation**
Contrastive Divergence (CD)

- **Gradient of Likelihood:**

\[
\frac{\partial L(X; \theta)}{\partial \theta} = \int p(x, \theta) \frac{\partial \log f(x; \theta)}{\partial \theta} dx - \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \log f(x^{(k)}; \theta)}{\partial \theta}
\]

- Intractable

<table>
<thead>
<tr>
<th>Tractable Gibbs Sampling</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sample</strong> (p(z_1, z_2, \ldots, z_M))</td>
</tr>
<tr>
<td>1. Initialize ({z_i : i = 1, \ldots, M})</td>
</tr>
<tr>
<td>2. For (\tau = 1, \ldots, T):</td>
</tr>
<tr>
<td>- Sample (z_1^{(\tau+1)} \sim p(z_1, z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})).</td>
</tr>
<tr>
<td>- Sample (z_2^{(\tau+1)} \sim p(z_2, z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})).</td>
</tr>
<tr>
<td>- Sample (z_j^{(\tau+1)} \sim p(z_j, z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau+1)}, \ldots, z_M^{(\tau)})).</td>
</tr>
<tr>
<td>- Sample (z_M^{(\tau+1)} \sim p(z_M, z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})).</td>
</tr>
</tbody>
</table>

Fast contrastive divergence \(T=1\)

\[
\theta_{t+1} = \theta_t + \lambda \frac{\partial L(X; \theta)}{\partial \theta}
\]

- **Accuracy vs. Speed**

- Accurate but slow gradient
- Approximate but fast gradient

\[
P(A, B, C) = P(A|C)P(B|C)P(C)
\]
Gibbs Sampling for graphical model

Figure 11.12 The Gibbs sampling method requires samples to be drawn from the conditional distribution of a variable conditioned on the remaining variables. For graphical models, this conditional distribution is a function only of the states of the nodes in the Markov blanket. For an undirected graph this comprises the set of neighbours, as shown on the left, while for a directed graph the Markov blanket comprises the parents, the children, and the co-parents, as shown on the right.

1. Initialize $\{z_i : i = 1, \ldots, M\}$
2. For $\tau = 1, \ldots, T$:
   - Sample $z_1^{(\tau+1)} \sim p(z_1 | z_2^{(\tau)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - Sample $z_2^{(\tau+1)} \sim p(z_2 | z_1^{(\tau+1)}, z_3^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_j^{(\tau+1)} \sim p(z_j | z_1^{(\tau+1)}, \ldots, z_{j-1}^{(\tau+1)}, z_{j+1}^{(\tau)}, \ldots, z_M^{(\tau)})$.
   - $\vdots$
   - Sample $z_M^{(\tau+1)} \sim p(z_M | z_1^{(\tau+1)}, z_2^{(\tau+1)}, \ldots, z_{M-1}^{(\tau+1)})$.

More information on Gibbs sampling:
Pattern recognition and machine learning (PRML)
Convergence of Contrastive divergence (CD)

- The fixed points of ML are not fixed points of CD and vice versa.
  - CD is a biased learning algorithm.
  - But the bias is typically very small.
  - CD can be used for getting close to ML solution and then ML learning can be used for fine-tuning.

- It is not clear if CD learning converges (to a stable fixed point). At 2005, proof is not available.

- Further theoretical results? Please inform us

Basic background on statistical learning and Graphical model

Contrastive divergence and Restricted Boltzmann machine
- Product of experts
- Contrastive divergence
- Restricted Boltzmann Machine

Deep belief net
Boltzmann Machine

- Undirected graphical model, with hidden nodes.

$$P(x; \theta) = \frac{\prod_m f_m(x_m; \theta_m)}{\sum_x \prod_m f_m(x_m; \theta_m)} = \frac{e^{-E(x; \theta)}}{\sum_x e^{-E(x; \theta)}} = \frac{f(x; \theta)}{Z(\theta)},$$

$$E(x; \theta) = -\sum_{i<j} w_{ij} x_i x_j - \sum_i \lambda_i x_i$$

$$\theta : \{w_{ij}, \lambda_i\}$$

Boltzmann machine: \( E(x, h) = b' x + c' h + h' Wx + x' Ux + h' Vh \)
restricted boltzmann machine (rbm)

- undirected, loopy, layer

\[ P(x, h) = \frac{e^{-E(x,h)}}{\sum_{x,h} e^{-E(x,h)}} \]

\[ P(x) = \frac{\sum_h e^{-E(x,h)}}{\sum_{x,h} e^{-E(x,h)}} \]

- \( E(x,h) = b' \cdot x + c' \cdot h + h' \cdot Wx \)

\[ P(h \mid x) = \prod_i P(h_i \mid x) \]

\[ P(x \mid h) = \prod_j P(x_j \mid h) \]

\[ P(x_j = 1 \mid h) = \sigma(b_j + W \cdot_j \cdot h) \]

\[ P(h_i = 1 \mid x) = \sigma(c_i + W_i \cdot x) \]

read the manuscript for details
**Restricted Boltzmann Machine (RBM)**

\[
P(x; \theta) = \frac{\sum_h e^{-(b'x + c'h + h'Wx)}}{\sum_{x,h} e^{-(b'x + c'h + h'Wx)}} = \frac{f(x; \theta)}{Z(\theta)}
\]

- \( E(x,h) = b'x + c'h + h'Wx \)
- \( x = [x_1 \ x_2 \ ...]^T, \ h = [h_1 \ h_2 \ ...]^T \)
- **Parameter learning**
  - Maximum Log-Likelihood

\[
\max_{\theta} \left\{ \prod_{k=1}^{K} P(x^{(k)}; \theta) \right\} \iff \min_{\theta} L(X; \theta) = \min_{\theta} \left\{ -\log \prod_{k=1}^{K} P(x^{(k)}; \theta) \right\}
\]

CD for RBM

- CD for RBM, very fast!

\[ P(x; \theta) = \frac{\sum_h e^{-(b'x + c' h + h'Wx)}}{\sum_{x,h} e^{-(b'x + c' h + h'Wx)}} = \frac{f(x; \theta)}{Z(\theta)} \]

\[ \theta_{t+1} = \theta_t + \lambda \frac{\partial L(X; \theta)}{\partial \theta} \]

\[ \frac{\partial L(X; \theta)}{\partial \omega_{ij}} = \int p(x, \theta) \frac{\partial \log f(x; \theta)}{\partial \theta} dx - \frac{1}{K} \sum_{k=1}^{K} \frac{\partial \log f(x^{(k)}; \theta)}{\partial \theta} \]

\[ = \langle x_i h_j \rangle_{p(x, \theta)} - \langle x_i h_j \rangle_x = \langle x_i h_j \rangle_\infty - \langle x_i h_j \rangle_0 \]

\[ \approx \langle x_i h_j \rangle_1 - \langle x_i h_j \rangle_0 \rightarrow \text{CD} \]

\[ P(x_j = 1|h) = \sigma(b_j + W_j \cdot h) \quad P(h_i = 1|x) = \sigma(c_i + W_i \cdot x) \]
### CD for RBM

\[
\frac{\partial L(X; \theta)}{\partial w_{ij}} \approx \langle x_i h_j \rangle_1 - \langle x_i h_j \rangle_0
\]

**for all hidden units** \(i\) **do**
- compute \(Q(h_{1i} = 1|x_1)\) (for binomial units, \(\text{sigm}(c_i + \sum_j W_{ij} x_{1j})\))
- sample \(h_{1i} \in \{0, 1\}\) from \(Q(h_{1i}|x_1)\)
**end for**

**for all visible units** \(j\) **do**
- compute \(P(x_{2j} = 1|h_1)\) (for binomial units, \(\text{sigm}(b_j + \sum_i W_{ij} h_{1i})\))
- sample \(x_{2j} \in \{0, 1\}\) from \(P(x_{2j} = 1|h_1)\)
**end for**

**for all hidden units** \(i\) **do**
- compute \(Q(h_{2i} = 1|x_2)\) (for binomial units, \(\text{sigm}(c_i + \sum_j W_{ij} x_{2j})\))
**end for**

\[
P(x_j = 1|h) = \sigma(b_j + W' \cdot j \cdot h)
\]

\[
P(h_i = 1|x) = \sigma(c_i + W_i \cdot x)
\]
RBM for classification

\( y \): classification label

\[
p(y, x, h) \propto e^{-E(y, x, h)}
\]

\[
E(y, x, h) = -h^T W x - b^T x - c^T h - d^T \tilde{y} - h^T U \tilde{y}
\]

\[
p(y|x) = \frac{e^{d_y} \prod_{j=1}^{n} \left(1 + e^{c_j + U_{jy} + \sum_i W_{ji} x_i}\right)}{\sum_{y^*} e^{d_{y^*}} \prod_{j=1}^{n} \left(1 + e^{c_j + U_{jy^*} + \sum_i W_{ji} x_i}\right)}
\]

\[
\frac{\partial \log p(y_i|x_i)}{\partial \theta} = \sum_j \text{sigm}(o_{y_j}(x_i)) \frac{\partial o_{y_j}(x_i)}{\partial \theta}
\]

\[
- \sum_{j, y^*} \text{sigm}(o_{y^*j}(x_i)) p(y^*|x_i) \frac{\partial o_{y^*j}(x_i)}{\partial \theta}
\]

\[
o_{yj}(x) = c_j + \sum_k W_{jk} x_k + U_{jy}
\]

RBM itself has many applications

- Multiclass classification
- Collaborative filtering
- Motion capture modeling
- Information retrieval
- Modeling natural images
- Segmentation

Y Li, D Tarlow, R Zemel, Exploring compositional high order pattern potentials for structured output learning, CVPR 2013
• Basic background on statistical learning and Graphical model
• Contrastive divergence and Restricted Boltzmann machine

• Deep belief net (DBN)
  ▪ Why deep learning?
  ▪ Learning and inference
  ▪ Applications
A belief net is a directed acyclic graph composed of random variables.
Deep Belief Net

- Belief net that is deep
- A generative model
  - $P(x,h_1,\ldots,h_l) = p(x|h_1) \ p(h_1|h_2) \ldots \ p(h_{l-2}|h_{l-1}) \ p(h_{l-1},h_l)$
- Used for unsupervised training of multi-layer deep model.

Pixels $\Rightarrow$ edges $\Rightarrow$ local shapes $\Rightarrow$ object parts

$$P(x,h_1,h_2,h_3) = p(x|h_1) \ p(h_1|h_2) \ p(h_2,h_3)$$
Why Deep learning?

- The mammal brain is organized in a deep architecture with a given input percept represented at multiple levels of abstraction, each level corresponding to a different area of cortex.
- An architecture with insufficient depth can require many more computational elements, potentially exponentially more (with respect to input size), than architectures whose depth is matched to the task.
- Since the number of computational elements one can afford depends on the number of training examples available to tune or select them, the consequences are not just computational but also statistical: poor generalization may be expected when using an insufficiently deep architecture for representing some functions.

Why Deep learning?

- Linear regression, logistic regression: depth 1
- Kernel SVM: depth 2
- Decision tree: depth 2
- Boosting: depth 2
- The basic conclusion that these results suggest is that when a function can be compactly represented by a deep architecture, it might need a very large architecture to be represented by an insufficiently deep one. (Example: logic gates, multi-layer NN with linear threshold units and positive weight).

Example: sum product network (SPN)

- \( N \cdot 2^{N-1} \) parameters
- \( O(N) \) parameters

\[
\begin{align*}
&\sum \prod \text{network (SPN)} \\
\xrightarrow{\text{2}^{N-1}}
\end{align*}
\]
Depth of existing approaches

- **Boosting (2 layers)**
  - L 1: base learner
  - L 2: vote or linear combination of layer 1

- **Decision tree, LLE, KNN, Kernel SVM (2 layers)**
  - L 1: matching degree to a set of local templates.
  - L 2: Combine these degrees

- **Brain: 5-10 layers**

\[ b + \sum_{i} \alpha_i K(x, x_i) \]
Why decision tree has depth 2?

- Rely on partition of input space.
- Local estimator. Rely on partition of input space and use separate params for each region. Each region is associated with a leaf.
- Need as many as training samples as there are variations of interest in the target function. Not good for highly varying functions.
- Num. training sample is exponential to Num. dim in order to achieve a fixed error rate.
• **Inference problem:** Infer the states of the unobserved variables.

• **Learning problem:** Adjust the interactions between variables to make the network more likely to generate the observed data.

\[
\mathcal{P}(x,h_1,h_2,h_3) = p(x|h_1) \ p(h_1|h_2) \ p(h_2,h_3)
\]
- Inference problem (the problem of explaining away):

\[ P(A,B|C) = P(A|C)P(B|C) \]

- An example from manuscript

\[ P(h_{11}, h_{12} | x_1) \neq P(h_{11} | x_1) P(h_{12} | x_1) \]

Sol: Complementary prior
Inference problem (the problem of explaining away)

- Sol: Complementary prior

```
\begin{align*}
  h_4 & \quad 30 \\
  h_3 & \quad 500 \\
  h_2 & \quad 1000 \\
  h_1 & \quad 2000 \\
  x &
\end{align*}
```
Deep Belief Net

- Explaining away problem of Inference (see the manuscript)
  - Sol: Complementary prior, see the manuscript
- Learning problem
  - Greedy layer by layer RBM training (optimize lower bound) and fine tuning
  - Contrastive divergence for RBM training

\[ P(h_i = 1 | x) = \sigma(c_i + W_i \cdot x) \]
Why greedy layerwise learning work?

Optimizing a lower bound:

\[
\log P(x) = \log \sum_h P(x, h_1) \\
\geq \sum_{h_1} \{Q(h_1 | x)[\log P(h_1) + \log P(h_1 | x)] - Q(h_1 | x) \log Q(h_1 | x)\}
\]

When we fix parameters for layer 1 and optimize the parameters for layer 2, we are optimizing the \(P(h_2)\) in (1)
RBM can be considered as DBN that has infinitive layers.
Pretrain, fine-tune and inference – (autoencoder)
Pretrain, fine-tune and inference - 2

Pretraining

Fine-tuning

Feature Representation F(X|W)

Input X

GP

target y

W_1^T

W_2^T

W_3

W_1

W_2

W_3^T

1000

1000

1000

1000
How many layers should we use?

- There might be no universally right depth
  - Bengio suggests that several layers is better than one
  - Results are robust against changes in the size of a layer, but top layer should be big
  - A parameter. Depends on your task.
  - With enough narrow layers, we can model any distribution over binary vectors [1]


Copied from http://videolectures.net/mlss09uk_hinton_dbn/
Effect of Unsupervised Pre-training

Erhan et. al. AISTATS’2009
Effect of Depth

without pre-training

with pre-training

number of layers

test classification error (perc)

number of layers

test classification error (perc)
Why unsupervised pre-training makes sense

If image-label pairs were generated this way, it would make sense to try to go straight from images to labels. For example, do the pixels have even parity?

If image-label pairs are generated this way, it makes sense to first learn to recover the stuff that caused the image by inverting the high bandwidth pathway.
Layer-wise pretraining is efficient but not optimal.

- It is possible to train parameters for all layers using a wake-sleep algorithm.
  - Bottom-up in a layer-wise manner
  - Top-down and reffiting the earlier models
Fine-tuning with a contrastive version of the “wake-sleep” algorithm

After learning many layers of features, we can fine-tune the features to improve generation.

1. **Do a stochastic bottom-up pass**
   - Adjust the top-down weights to be good at reconstructing the feature activities in the layer below.

2. **Do a few iterations of sampling in the top level RBM**
   - Adjust the weights in the top-level RBM.

3. **Do a stochastic top-down pass**
   - Adjust the bottom-up weights to be good at reconstructing the feature activities in the layer above.
Include lateral connections

- RBM has no connection among layers
- This can be generalized.
- Lateral connections for the first layer [1].
  - Sampling from $P(h|x)$ is still easy. But sampling from $P(x|h)$ is more difficult.
- Lateral connections at multiple layers [2].
  - Generate more realistic images.
  - CD is still applicable, with small modification.

Without lateral connection

real data

samples from model
With lateral connection

real data

samples from model
My data is real valued ...

- Make it \([0, 1]\) linearly: \(x = ax + b\)
- Use another distribution

\[
- \log p(v, h) = \sum_i \frac{(v_i - c_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij} + \text{const}
\]

\[
p(h_j = 1|v) = f(b_j + \sum_i v_i w_{ij}),
\]

\[
p(v_i|h) = \mathcal{N}(c_i + \sum_i h_j w_{ij}, 1),
\]

\[
\Delta w_{ij} \propto \langle v_i h_j \rangle_{\text{data}} - \langle v_i h_j \rangle_{\text{recon}}
\]

\[
\Delta b_j \propto \langle h_j \rangle_{\text{data}} - \langle h_j \rangle_{\text{recon}}
\]
My data has temporal dependency ...

- **Static:**

\[- \log p(v, h) = \sum_i \frac{(v_i - c_i)^2}{2\sigma_i^2} - \sum_j b_j h_j - \sum_{i,j} \frac{v_i}{\sigma_i} h_j w_{ij} + \text{const}\]

- **Temporal**

\[p(v_t, h_t | v_{<t}, \theta) = \exp \left( -E (v_t, h_t | v_{<t}, \theta) \right) / Z\]

\[E = \sum_i \frac{\left(\hat{a}_{i,t} - v_{i,t}\right)^2}{2\sigma_i^2} - \sum_j \hat{b}_{j,t} h_{j,t} - \sum_{i,j} W_{ij} \frac{v_{i,t}}{\sigma_i} h_{j,t}\]

\[\hat{a}_{i,t} = a_i + \sum_k A_{ki} v_{k,<t}\]

\[\hat{b}_{j,t} = b_j + \sum_k B_{kj} v_{k,<t}\]

\[\sigma_i = 1\]
Consider DBN as...

- A statistical model that is used for unsupervised training of fully connected deep model
- A directed graphical model that is approximated by fast learning and inference algorithms
- A directed graphical model that is fine tuned using mature neural network learning approach -- BP.
Outline

- Basic background on statistical learning and Graphical model
- Contrastive divergence and Restricted Boltzmann machine
- **Deep belief net (DBN)**
  - Why DBN?
  - Learning and inference
  - Applications
Applications of deep learning

- Hand written digits recognition
- Dimensionality reduction
- Information retrieval
- Segmentation
- Denoising
- Phone recognition
- Object recognition
- Object detection
- ...

Welling, M. etc., Exponential Family Harmoniums with an Application to Information Retrieval, NIPS 2004
A. R. Mohamed, etc., Deep Belief Networks for phone recognition, NIPS 09 workshop on deep learning for speech recognition.
Nair, V. and Hinton, G. E. 3-D Object recognition with deep belief nets. NIPS09

..............................
Object recognition

• NORB
  - logistic regression 19.6%, kNN (k=1) 18.4%, Gaussian kernel SVM 11.6%, convolutional neural net 6.0%, convolutional net + SVM hybrid 5.9%. DBN 6.5%.
  - With the extra unlabeled data (and the same amount of labeled data as before), DBN achieves 5.2%.
Learning to extract the orientation of a face patch (Salakhutdinov & Hinton, NIPS 2007)
The training and test sets

100, 500, or 1000 labeled cases
11,000 unlabeled cases

face patches from new people
The root mean squared error in the orientation when combining GP’s with deep belief nets

<table>
<thead>
<tr>
<th>Labels</th>
<th>GP on the pixels</th>
<th>GP on top-level features</th>
<th>GP on top-level features with fine-tuning</th>
</tr>
</thead>
<tbody>
<tr>
<td>100 labels</td>
<td>22.2</td>
<td>17.9</td>
<td>15.2</td>
</tr>
<tr>
<td>500 labels</td>
<td>17.2</td>
<td>12.7</td>
<td>7.2</td>
</tr>
<tr>
<td>1000 labels</td>
<td>16.3</td>
<td>11.2</td>
<td>6.4</td>
</tr>
</tbody>
</table>

Conclusion: The deep features are much better than the pixels. Fine-tuning helps a lot.
Deep Autoencoders (Hinton & Salakhutdinov, 2006)

- They always looked like a really nice way to do non-linear dimensionality reduction:
  - But it is very difficult to optimize deep autoencoders using backpropagation.
- We now have a much better way to optimize them:
  - First train a stack of 4 RBM’s
  - Then “unroll” them.
  - Then fine-tune with backprop.
Deep Autoencoders (Hinton & Salakhutdinov, 2006)
A comparison of methods for compressing digit images to 30 real numbers.
Representation of DBN
Deep belief net (DBN)
- is a network with deep layers, which provides strong representation power;
- is a generative model;
- can be learned by layerwise RBM using Contrastive Divergence;
- has many applications and more applications is yet to be found.

Generative models explicitly or implicitly model the distribution of inputs and outputs. Discriminative models model the posterior probabilities directly.
A very controversial topic

Model
- DBN is generative, SVM is discriminative. But fine-tuning of DBN is discriminative.

Application
- SVM is widely applied.
- Researchers are expanding the application area of DBN.

Learning
- DBN is non-convex and slow
- SVM is convex and fast (in linear case).

Which one is better?
- Time will say.
- You can contribute

Hinton: The superior classification performance of discriminative learning methods holds only for domains in which it is not possible to learn a good generative model. This set of domains is being eroded by Moore’s law.