Deep Learning Architectures Survey [1]

Byung Kang

Department of Computer Sciences
Applied Algorithms Lab.

September 27, 2011

Outline

What is Deep Learning?

- Learning the “representation” of objects
- Ability to use hierarchical abstraction
- i.e. Gradually constructing representation in higher levels
- e.g. Pixels -> Lines -> Shapes -> Objects -> Scenery -> Situation

[- 0.90 41.11 68.51 34.25 -0.10 0 0.05]
[0.84 109.62 109.62 34.25 0.37 0 -0.04]
[0.76 68.51 164.44 34.25 -0.42 0 0.16]
[0.17 246.66 123.33 34.25 0.85 0 -0.04]
[0.16 178.14 54.81 34.25 0.38 0 -0.14]

Figure: How we see an image (left) vs. how machines see an image (right).

History

- Boltzman Machines - graphical model for joint probability representation
- But this soon got replaced by artificial neural network
  - Training procedure for Boltzman machines were too complex
  - ANN's back-propagation was much easier and more intuitive
- ANN retired - increasing layers gave worse results
- Shallow architectures became popular (e.g. SVMs, decision trees, etc.)
- Recent revival (late 2000s) of deep architecture
- Simple layer-wise greedy pre-training proved to be effective for deep ANNs

Restricted Boltzman Machines (RBMs)

- One of the early graphical models
- An energy-based 2-layer *generative* model
- Train to model the distribution $p(x)$, for training set $\{x_i\}$

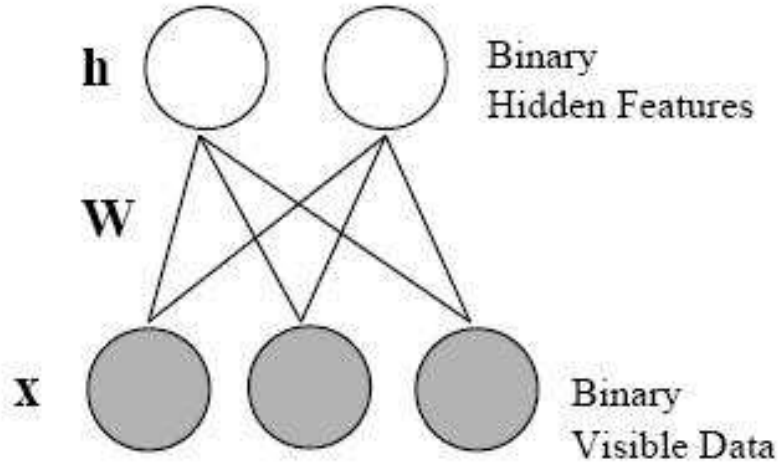
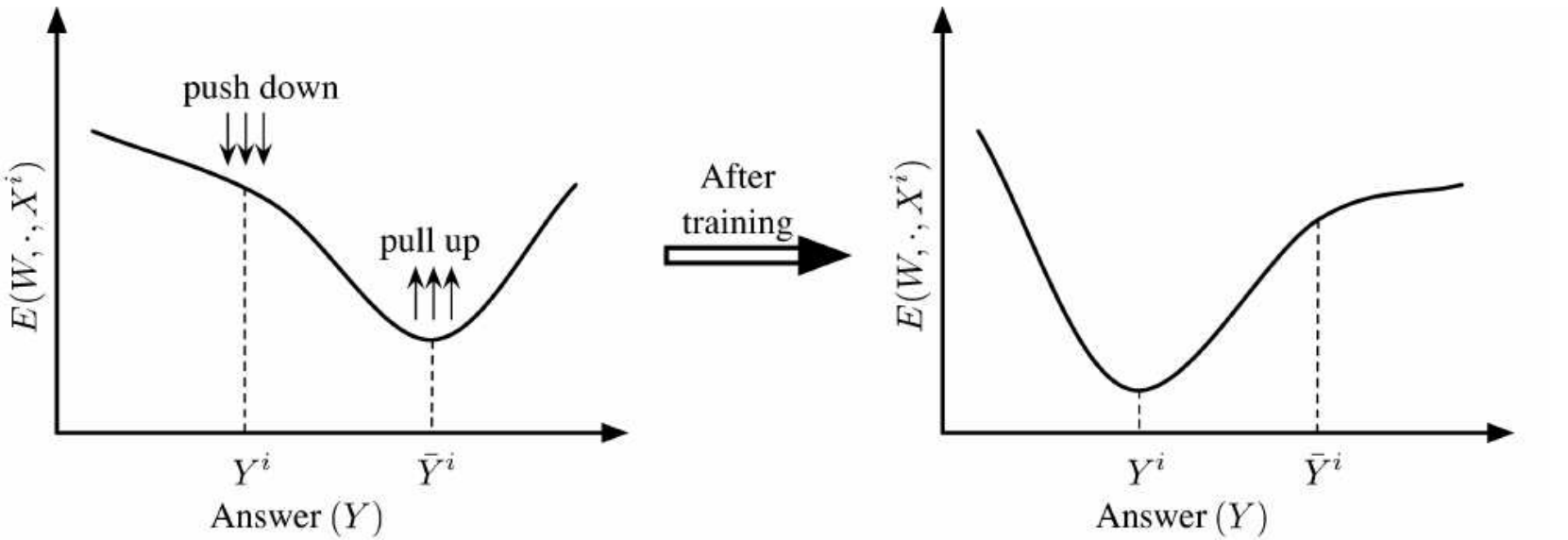


Figure: The bottom nodes are the “visible” ones and the top nodes are the latent variables.

RBMs are “restricted” because they lack connections among same-layer nodes.

Sidenote: Energy-Based Models

- Distribution of the form: $p(x) = \frac{exp(-Energy(x))}{Z}$
- The higher the energy, the less probable x is.
- Any probability distribution can be cast as energy-based models.



RBM Mechanism

- Energy has form:

$$Energy(x, h) = -b^T x - c^T h - h^T W x$$

  where W is the weight matrix on the connections, and b, c are the weight vectors on nodes.
- Because there are no connections on the same level, the conditionals are factorized:

$$p(h|x) = \prod_i p(h_i|x) \quad \text{(same for } p(x|h)\text{)}$$

- Most commonly used values for x, h are binary ($x_i, h_i \in \{0, 1\}$) - see next slide

Binary RBMs

For binary RBMs, the posterior has a well-known sigmoidal form:

$$\begin{aligned} p(h|x) &= \frac{\exp(b^T x + c^T h + h^T W x)}{\sum_{h'} \exp(b^T x + c^T h + h^T W x))} \\ &= \frac{\prod_i \exp(c_i h_i + h_i W_i x)}{\prod_i \sum_{h'} \exp(c_i h'_i + h'_i W_i x)} \\ &= \prod_i \frac{\exp(c_i h_i + h_i W_i x)}{\sum_{h'} \exp(c_i h'_i + h'_i W_i x)} \end{aligned}$$

The term in the product is precisely a sigmoid function. Hence,

$$p(h_i = 1|x) = \frac{\exp(c_i + W_i x)}{1 + \exp(c_i + W_i x)} = \sigma(c_i + W_i x).$$

A similar form can be derived for $p(x_j = 1|h) = \sigma(b_j + W'_j h)$

Sampling from RBMs

How do we sample from the distribution represented by the RBM (i.e. $p(x)$)?

Gibbs sampling:

1. Set $x_0 \sim \hat{p}(x)$
2. Set $h_0 \sim p(h|x_0)$
3. For $i = 1$ to k
   - $x_i \sim p(x|h_{i-1})$
   - $h_i \sim p(h|x_i)$
4. Return (x_k, h_k)

, where $\hat{p}$ is the empirical distribution of the training set.
This sampling procedure is needed for DBN training (later slides).

Training RBMs

- Q:Then, how do we train the RBM to represent the distribution over training samples $\{x_i\}$?
- A: Maximize the log probability $\log p(x)$ w.r.t. the parameters W, b, c.

First, define $FreeEnergy(x) \equiv -\log \sum_h \exp(-Energy(x,h))$ so that

$$p(x) = \sum_h p(x,h) = \sum_h \frac{\exp(-Energy(x,h))}{Z} = \frac{\exp(-FreeEnergy(x))}{Z},$$

$$\frac{\partial \log p(x)}{\partial \theta} = -\frac{\partial FreeEnergy(x)}{\partial \theta} + \sum_{x'} p(x') \frac{\partial FreeEnergy(x')}{\partial \theta}$$

The second term is just $E_p[\frac{\partial FreeEnergy(x')}{\partial \theta}]$.
i.e. We need to sample from the true distribution p.

Contrastive Divergence

Approximate the second term in two levels:

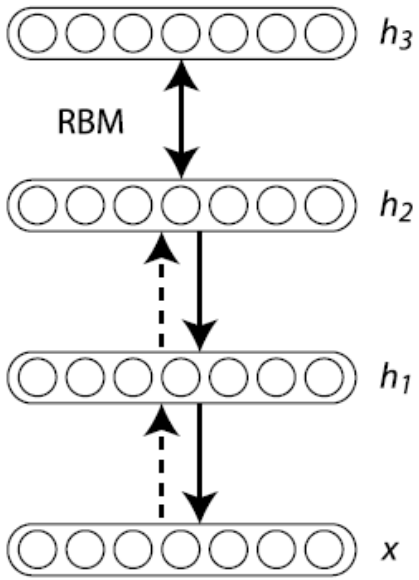
- First approximation: just take a single MCMC sample
- Second approximation: just take k steps of MCMC.
- This is known as the k-step Contrastive Divergence (CD-k)

Surprisingly, CD-1 works well empiricallly.
i.e., when training for sample x,

- Sample $h \sim p(h|x)$
- Sample $x' \sim p(x'|h)$
- Compute the gradient around x' to approximate $E_p[\frac{\partial FreeEnergy(x')}{\partial \theta}]$

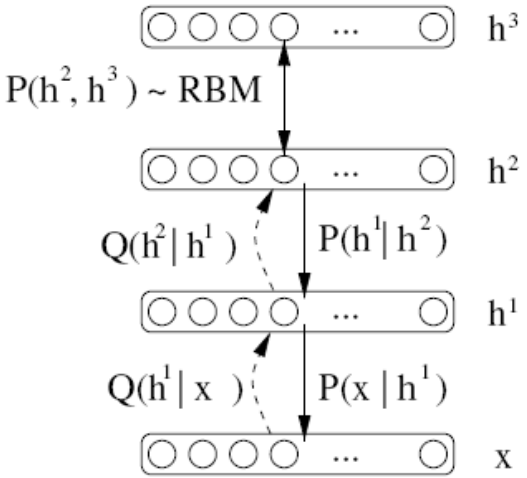
RBMs as Building-Blocks

- Vertically stack RBMs to get a DBN
- Each layer represents a latent feature of the domain
- Still models a joint distribution of all layers (generative model)
- Latent nodes of the previous layer becomes the input to the next layer
- Top-most layer is a full RBM (serves as a joint prior $p(h^k, h^{k-1})$)



Greedy Layer-Wise Training

- Treat each layer as an RBM and train accordingly
- Start from the bottom layer, and train upwards
- Training done in unsupervised fashion (as in RBM)



Training DBNs in Detail

- When training the k-th layer, use the latent nodes of $k-1$ as inputs
- Since we're learning greedily, there's no guarantee for optimality
- But this "pre-training" leads to good initialization for later supervised tasks

Given the empirical distribution $\hat{p}(\cdot)$,

- For each layer k,
  - Init W^k, b^k, c^k
    - For $i = 1$ to $k-1$
    - Sample $h_j^i \sim p(h_j^i | h^{i-1})$
  - Train k-th RBM with h^{k-1} to update W^k, b^k, c^k

Sampling from DBNs

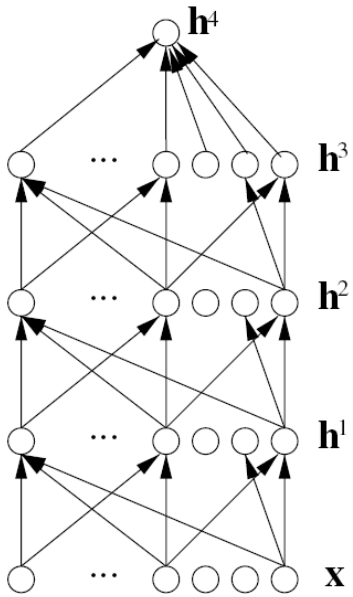
Since DBN also represents a probability distribution over samples, we should be able to produce samples from that distribution.

1. Sample a visible vector h^{l-1} from the top-level RBM. (c.f. Gibbs sampling)
2. For $k = l - 1$ downto 1, sample h^{k-1} from $p(h^{k-1}|h^k)$ (sigmoid in case of binary RBM)
3. Return $x = h^0$

NOTE: When performing Gibbs sampling, fewer steps might be needed if the initial seed is sampled from the empirical distribution $\hat{p}$.

Leverage a Discriminative Model

- DBNs are generative models, but we can use it for discriminative tasks
- e.g. A deep multi-layered neural network (figure below)
- Initialize the deep NN's weight with those of trained DBN
- Training DBNs adjusts the weights to account for latent features
- Now, fine-tune the weights using supervised signal




Yoshua Bengio.
Learning deep architectures for ai.
Foundations and Trends in Machine Learning, 2, 2009.