The Origin of Deep Learning

Lili Mou

Jan, 2015

Acknowledgment

Most of the materials come from G. E. Hinton's online course.

Outline

Introduction

Preliminary

Boltzmann Machines and RBMs

Deep Belief Nets

- The Wake-Sleep Algorithm
- BNs and RBMs

Examples

Introduction

A Little History about Neural Networks

- Perceptions
- Multi-layer neural networks
    - Model capacity
    - Inefficient representations
    - The curse of dimensionality
- Early attempts for deep neural networks
    - Optimization and generalization
- One of the few successful deep architectures: Convolutional Neural networks
- Successful pretraining methods
    - Deep belief nets
    - Autoencoders

Stacked Boltzmann Machines and Deep belief nets are successful pretraining architectures for deep neural networks.

Learning a deep neural network has two stages:

1. Pretrain the model unsupervisedly
2. Initialize the weights in feed-forward neural networks as have been pretrained

Preliminary

A **bayesian network** is

- A directed acyclic graph G (DAG), whose nodes correspond to the random variables $X_1, X_2, \cdots, X_n$.
- For each node X_i, we have a conditional probabilistic distribution (CPD) $\mathrm{P}(X_i | \mathrm{Par}_{\mathrm{G}}(X_i))$

The joint distribution is

$$\mathrm{P}(\boldsymbol{X}) = \prod_{i=1}^{n} \mathrm{P}(X_i | \mathrm{Par}_{\mathrm{G}}(X_i))$$

Is $\boldsymbol{X}$ independent of $\boldsymbol{Y}$ given evidence $\boldsymbol{Z}$?

- $X \rightarrow Y$
- $X \leftarrow Y$
- $X \rightarrow W \rightarrow Y$
- $X \leftarrow W \leftarrow Y$
- $X \leftarrow W \rightarrow Y$
- $X \rightarrow W \leftarrow Y$

Definition (Active trail) $X_1 - X_2 - X_n$ is **active** give Z iff

- For any v-structure $X_{i-1} \rightarrow X_i \leftarrow X_{i+1}$, we have X_i or one of its descendents $\in Z$
- No other X_i is in Z

Definition (d-separation): X and Y are **d-separated** given evidence Z if there is no active trail from X to Y given Z.

Theorem d-separation $\Rightarrow$ (conditional) independency

Reasoning Patterns

3 typical reasoning patterns:

1. **Causal Reasoning** (Prediction)
   Predict "downstream" effects of various factors.
2. **Evidential Reasoning** (Explanation)
   Reason from effects to causes
3. **Intercausal Reasoning** (Explaining away)
   More tricky. The intuition is:
   Both flu and cancer cause fevers. When we have a fever, if we know that we have a flu, the probability of a cancer declines.

Factors: $\Phi = \{\phi_1(D_1), \cdots, \phi_k(D_k)\}$
ϕ_i is a factor over scope $D_i \subseteq \{X_1 \cdots X_n\}$

Unnormalized measure: $\tilde{\mathrm{P}}_\Phi(X_1, \cdots, X_n) = \prod_{i=1}^{k} \phi_i(D_i)$

Partition function: $Z_\Phi = \sum_{X_1, \cdots, X_n} \tilde{\mathrm{P}}_\Phi(X_1, \cdots, X_n)$

Joint probabilistic distribution: $\mathrm{P}_\Phi(X_1, \cdots, X_n) = \frac{1}{Z} \tilde{P}_\Phi(X_1, \cdots, X_n)$

Factors: $\Phi = \{\phi_1(X, Y), \cdots, \phi_k(X, Y)\}$

Unnormalized measure: $\tilde{\mathrm{P}}_\Phi(Y|X) = \prod_{i=1}^{k} \phi_i(D_i)$

Partition function: $Z_\Phi(X) = \sum_Y \tilde{\mathrm{P}}_\Phi(Y|X)$

Conditional probabilistic distribution: $\mathrm{P}_\Phi(Y|X) = \frac{1}{Z_\phi(X)} \tilde{P}_\Phi(Y|X)$

- Exact inference, e.g., clique tree calibration
- Sampling methods, e.g., Gibbs sampling, Metropolis Hastings algorithms
- Variational inference

Gibbs sampling

Posterior distributions $\mathrm{P}(X_i|X_{-i})$ known, give an unbiased sample of $\mathrm{P}(\boldsymbol{X})$

Wait until mixed $\{$

  Loop over i $\{$

    $x_i \sim \mathrm{P}(X_i|X_{-i})$

  $\}$

$\}$

In probabilistic graphical models, $\mathrm{P}(X_i|X_{-i})$ is easy to compute, which is only related to local factors.

- **Supervised learning**, all variables are known in the training set
  - Maximum likelihood estimation
  - Max a posteriori
- **Unsupervised/Semisupervised learning**, some variables are unknown for at least some training samples
  - Expectation maximum

    Loop {

    1. Estimate the probability of the unknown variables
    2. Estimate the parameters of the Bayesian network (MLE/MAP)

    }

N.B.: Inference is required for un-/semi-supervised learning

Rewrite the joint probability as

$$\mathrm{P}(X) = \frac{1}{Z} \exp\left\{\sum_i \theta_i f_i(X)\right\}$$

MLE estimation with gradient based methods

$$\frac{\partial \log P}{\partial \theta_i} = \mathbb{E}_{\text{data}}[f_i] - \mathbb{E}_{\text{model}}[f_i]$$

N.B.: Inference is required during each iteration of learning

Boltzmann Machines and RBMs

We have visible variables $\boldsymbol{v}$ and hidden variables $\boldsymbol{h}$. Both $\boldsymbol{v}$ and $\boldsymbol{h}$ are binary

Define energy function

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_i b_i v_i - \sum_j b_k h_k - \sum_{i<j} w_{i,j} v_i v_j - \sum_{i,k} w_{ik} v_i h_k - \sum_{k<l} w_{kl} h_k h_l$$

Define probability

$$\mathrm{P}(\boldsymbol{v}, \boldsymbol{h}) = \frac{1}{Z} \exp\{-E(\boldsymbol{v}, \boldsymbol{h})\}$$

where

$$Z = \sum_{\boldsymbol{v}, \boldsymbol{h}} \mathrm{P}(\boldsymbol{v}, \boldsymbol{h})$$

Two types of factors

1. Biases.

x_i	$\phi(x_i)$
0	1
1	$\log b_i$

2. Pairwise factors.

x_i	x_j	$\phi(x_i, y_j)$
0	0	1
0	1	1
1	0	1
1	1	$\log w_{i,j}$

$$\nabla_{\theta_i}(-\log \mathcal{L}(\Theta)) = \mathbb{E}_{\text{data}}[f_i] - \mathbb{E}_{\text{model}}[f_i]$$

Training BMs

- Brute-force (intractable)
- MCMC

$$p_i \equiv \mathrm{P}(s_i | \boldsymbol{s}_{-i}) = \sigma(-\sum_j s_j w_{ji} - b_i)$$

- Warm starting

  Keep the $\boldsymbol{h}^{(i)}$ as "particles" for each training sample $\boldsymbol{x}^{(i)}$

  Run the Markov chain only a few times after each weight update

- Mean-field approximation

$$p_i = \sigma(-\sum_j p_j w_{ji} - b_i)$$

Restricted Boltzmann Machines (RBM)

To achieve better training, we add constraints to our model

1. Only one hidden layer and one visible layer
2. No hidden connections
3. No visible connections (which also marginalized out in BMs)

Our model becomes a complete bipartite graph

Nice properties:

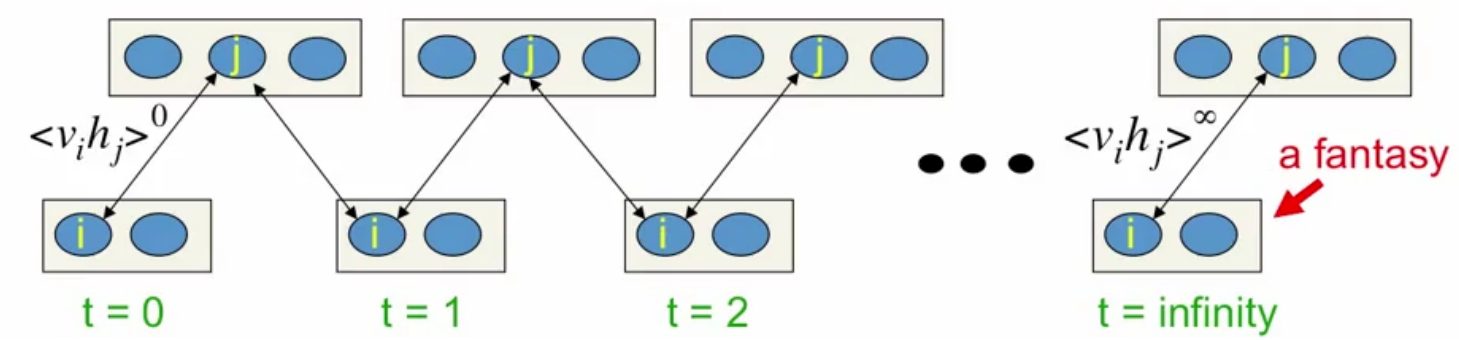
- $h_i \perp h_j | \boldsymbol{v}$
- $v_i \perp v_j | \boldsymbol{h}$

The learning will be much faster

- $\mathbb{E}_{\text{data}}[v_i h_j]$ can be computed within exactly one matrix multiplication
- $\mathbb{E}_{\text{model}}[v_i h_j]$ is also easier to compute because the Gibbs sampling becomes

$$\boldsymbol{h}|\boldsymbol{v} \Rightarrow \boldsymbol{v}|\boldsymbol{h} \Rightarrow \boldsymbol{h}|\boldsymbol{v} \cdots$$

Contrastive Divergence Learning Algorithm



- MCMC:

$$\Delta w_{ij} = \alpha(< v_i h_j >^0 - < v_i h_j >^\infty)$$

- CK-k

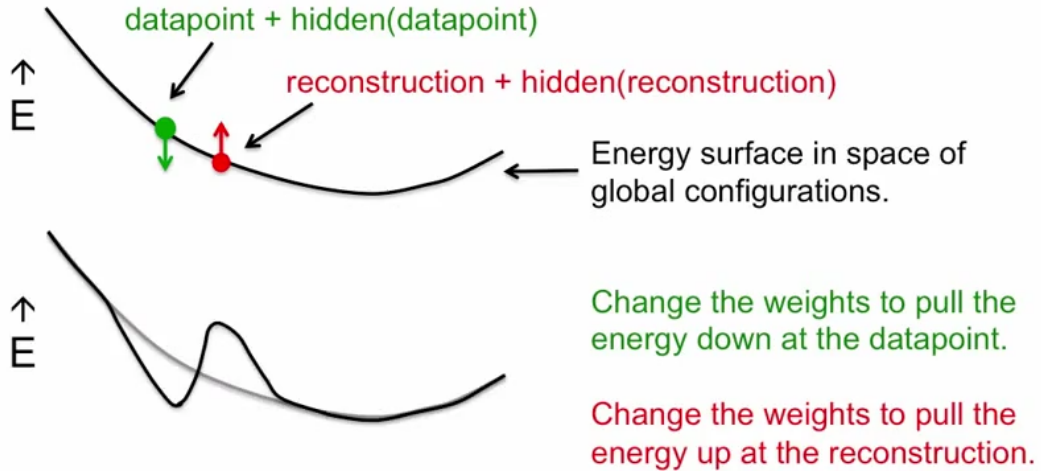
$$\Delta w_{ij} = \alpha(< v_i h_j >^0 - < v_i h_j >^k)$$

The crazy idea: Do things wrongly and hope it works.

Why does it work at all? When does it fail?

Why does it work at all?

- Start from the data, Markov chain wanders away from the current data towards things that it likes more.
- Perhaps, we only need one or a few steps to know the **direction** to the stationary distribution.



When does it fail?

- When the current low energy space is far away from any data points.

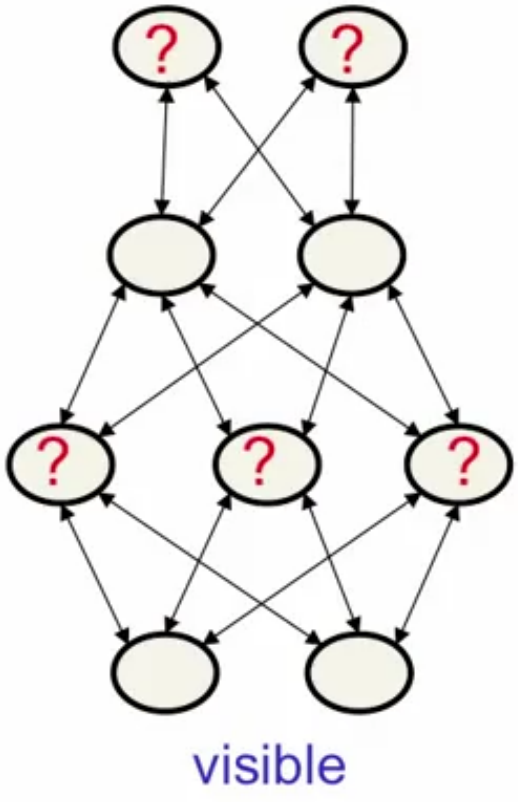
A good compromise

- CD-1 $\Rightarrow$ CD-3 $\Rightarrow$ CD-k

Exploring "Deep" Structures

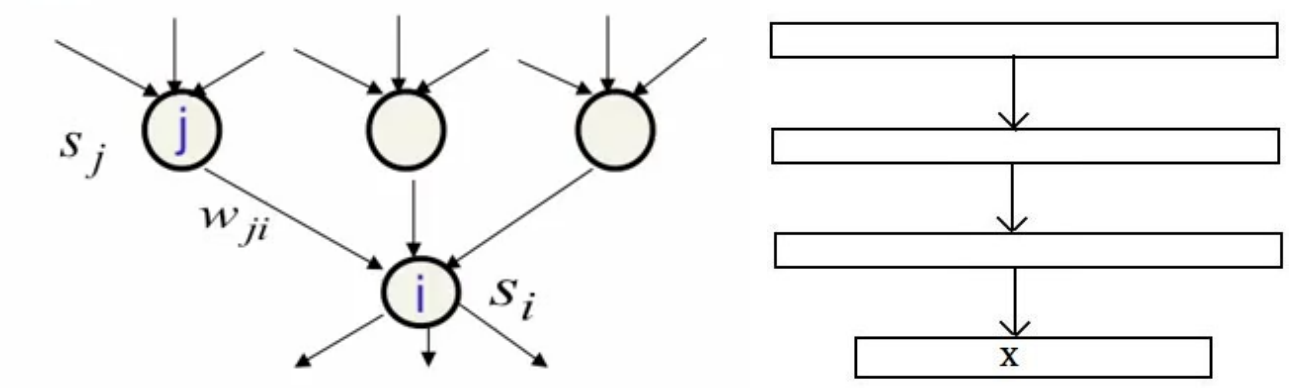
- Deep Boltzmann machines (DBM): Boltzmann machines with multiple hidden layers
- Learning: Following the learning rules of BMs, sample hidden units once a half of the net

What if we train layer by layer greedily?



Deep Belief Nets

Consider a generative process that generates our data $\boldsymbol{x}$ in a causal fashion



The conditional probabilistic distribution is in the form

$$p_i \equiv \mathrm{P}(s_i = 1) = \sigma(-b_i - \sum_j s_j w_{ji}) = \frac{1}{1 + \exp\left\{-b_i - \sum_j s_j w_{ji}\right\}}$$

where w_{ji} and b_i are the weights to be learned.

Learning DBNs

The learning process would be easy if we could get unbiased samples s_i

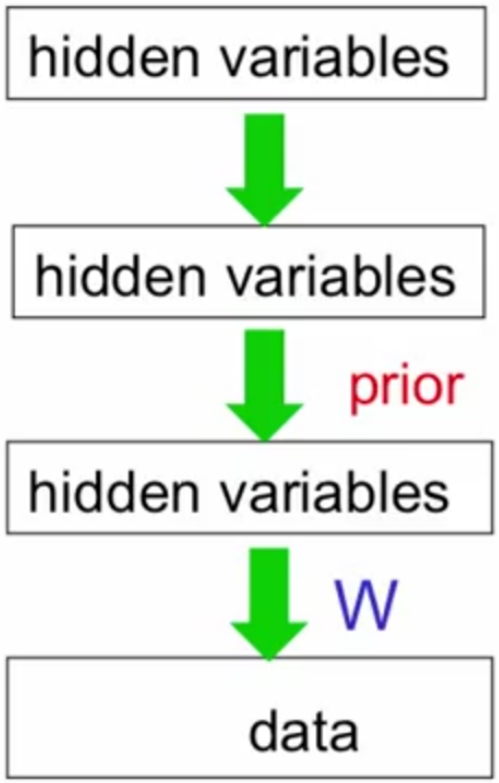
$$\Delta w_{ji} = \alpha s_j (s_i - p_i)$$

However, it is **intractable** to get unbiased samples for a *deep, densely-connected* Bayesian network

- Posterior not factorial even with only one hidden layer (the phenomenon of "explaining away")
- Posterior depends on the prior as well as likelihood

$\Rightarrow$ All the weights interact.

- Even worse, to compute the prior of one layer, we need to marginalize out all the hidden layers above.



Some Algorithms for Learning DBNs

- Using Markov chain Monto Carlo methods (Neal, 1992)
- Variational methods to get approximate samples from the posterior (1990's)

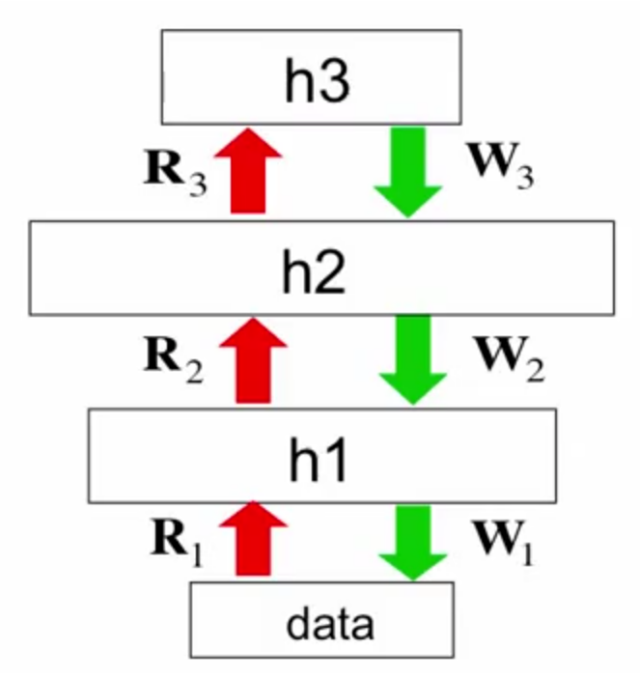
The crazy idea: Do things wrongly and hope it works!

The Wake-Sleep Algorithm

The Wake-Sleep Algorithm (Hinton, 1995)

Basic idea: ignore "explaining away"

- **Wake phase**: Use recognition weights $R = \mathrm{P}(\boldsymbol{Y}|\boldsymbol{X})$ to perform a bottom-up pass, and learn the generative weights
- **Sleep phase**: Use generative weights $W = \mathrm{P}(\boldsymbol{X}|\boldsymbol{Y})$ to generate samples from the model, and learn the recognition weights



N.B.: The recognition weights are not part of the generative model

Recall: $\mathbb{E}_{\text{data}}$ and $\mathbb{E}_{\text{data}}$

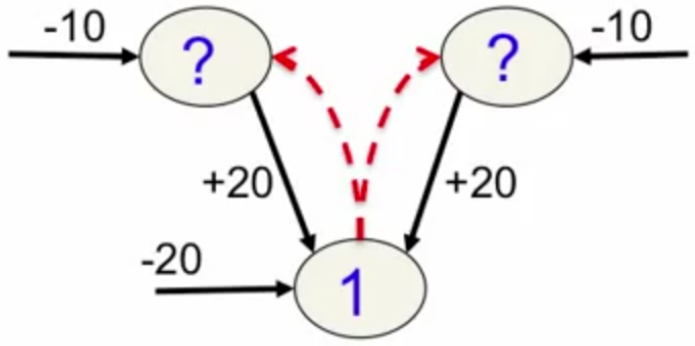
Mode averaging

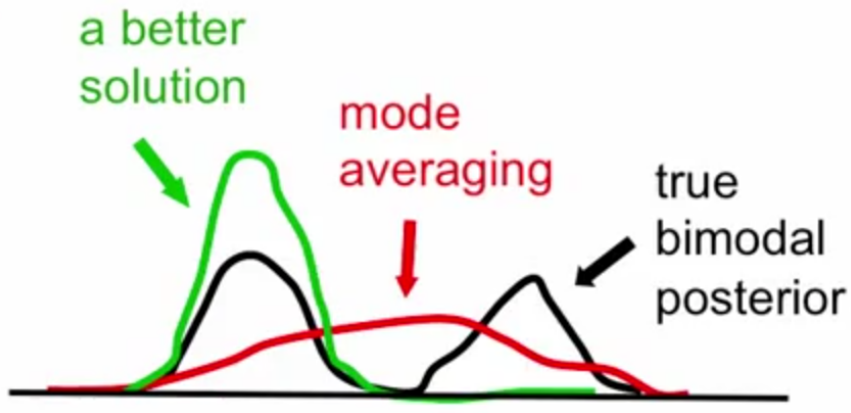
If we generate from the model, half the instances of a 1 at the data layer will be caused by a (1,0) at the hidden layer and half will be caused by a (0,1).

- So the recognition weights will learn to produce (0.5, 0.5)
- This represents a distribution that puts half its mass on 1,1 or 0,0: very improbable hidden configurations.

Its much better to just pick one mode.

- This is the best recognition model you can get if you assume that the posterior over hidden states factorizes





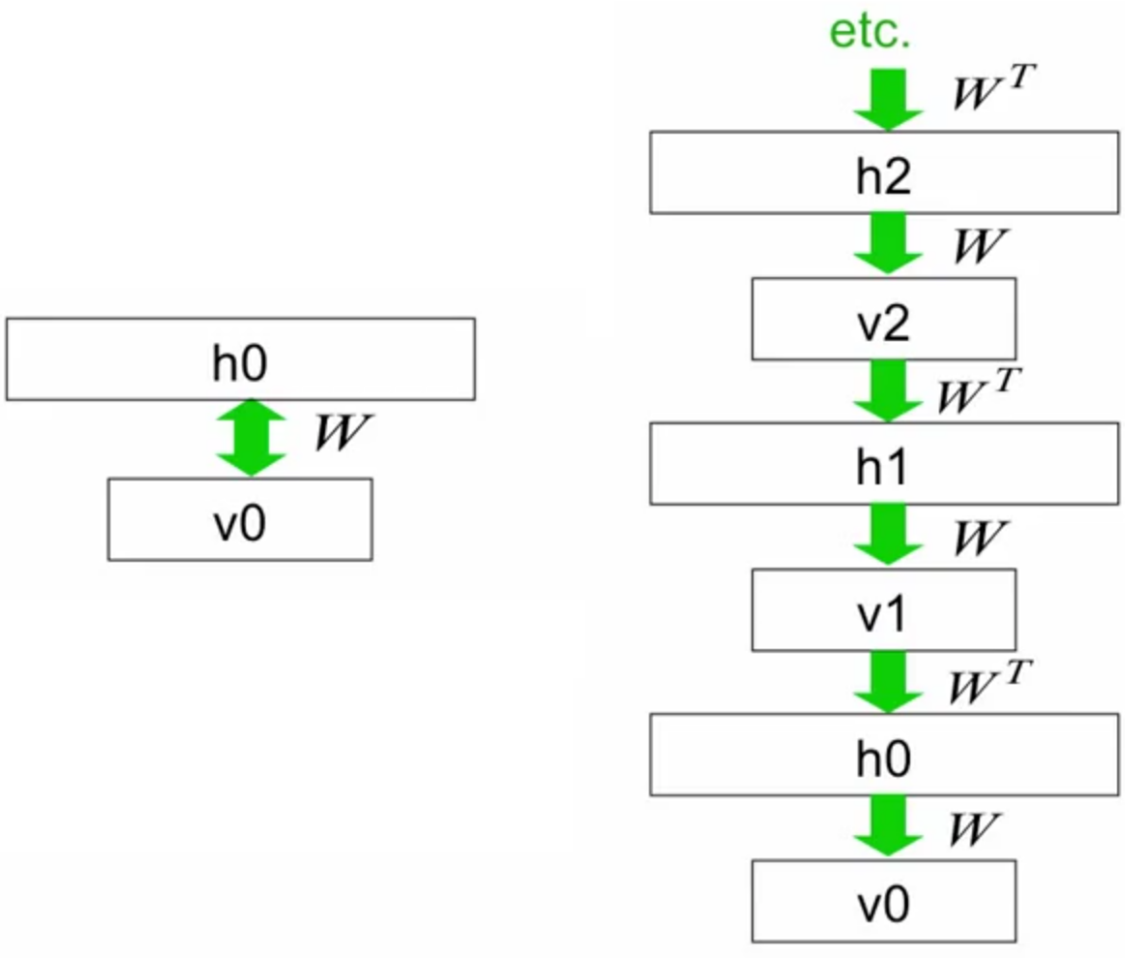
BNs and RBMs

RBM

||

BN with infinite layers with tied weights

The Intuition



Complementarity Prior

Factorial posterior, i.e., $\mathrm{P}(\boldsymbol{x}|\boldsymbol{y}) = \prod_i \mathrm{P}(x_i|\boldsymbol{y})$ and $\mathrm{P}(\boldsymbol{y}|\boldsymbol{x}) = \prod_i \mathrm{P}(y_i|\boldsymbol{x})$

$\Leftrightarrow$ Independent set $\{y_i \perp y_k|\boldsymbol{x}, \quad x_i \perp x_j|\boldsymbol{y}\}$

$\Leftrightarrow$ (Assume positivity)

$$\mathrm{P}(\boldsymbol{x}, \boldsymbol{y}) = \frac{1}{Z} \exp\left\{\sum_{i,j} \psi_{i,j}(x_i, y_j) + \sum_i \gamma_i(x_i) + \sum_j \alpha_j(y_j)\right\}$$

Can be extended to infinite layers within a-few-line derivations

CD Learning Revisit

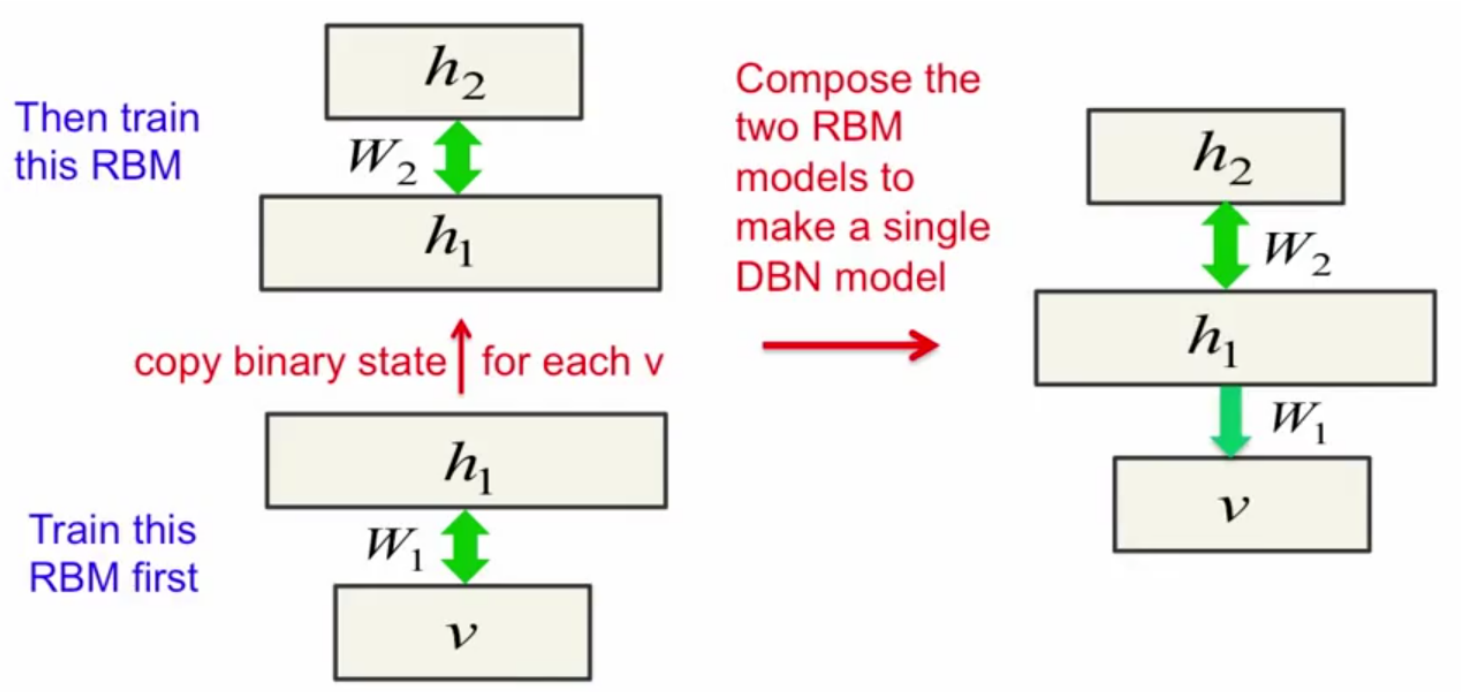
- CD-k: Ignore the "explaining away" phenomenon by ignoring the small derivatives contributed by the tied weights in higher layers
- When weights are small, Markov chain will be mixed soon.
- When weights get larger, run more iterations of CD.

- Good news:

  For multi-layer feature learning (in DBNs), it is just OK to use CD-1.

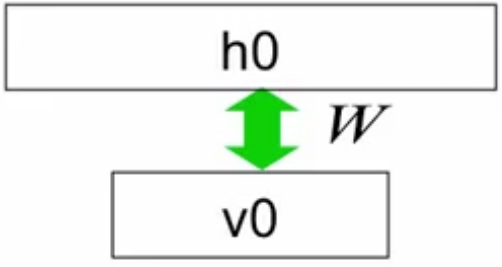
  (why?)

What if we train stacked RBMs layer by layer greedily?

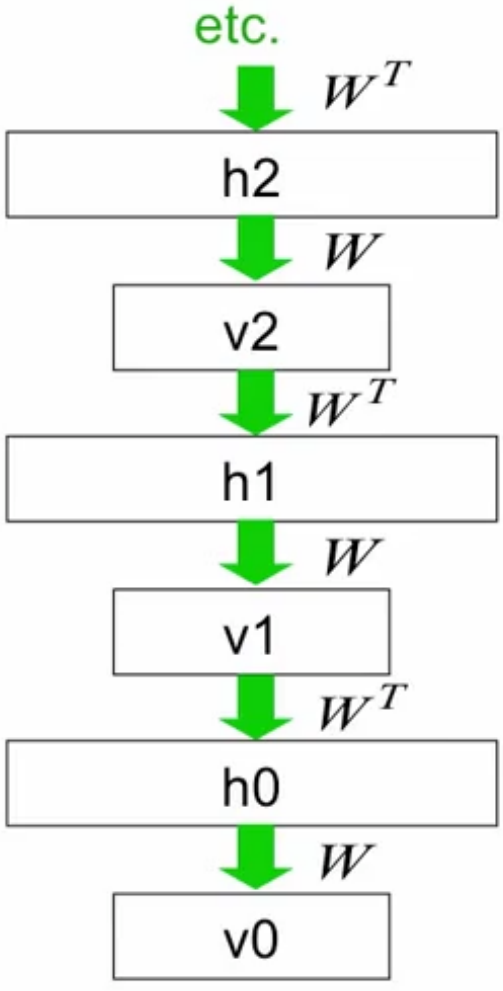


Learning a deep directed network

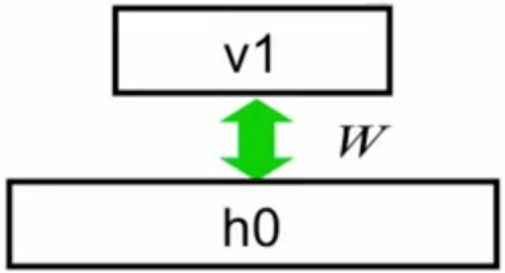
- First learn with all the weights tied. This is exactly equivalent to learning an RBM.

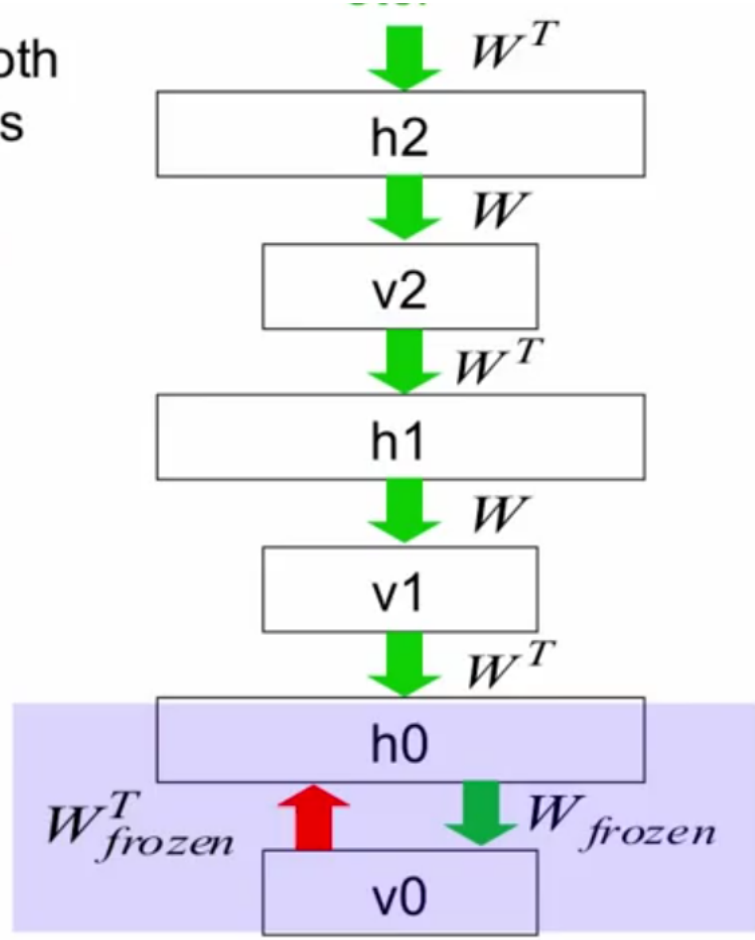


  - Think of the symmetric connections as a shorthand notation for an infinite directed net with tied weights.



- Then freeze the first layer of weights in both directions and learn the remaining weights (still tied together).
  - This is equivalent to learning another RBM, using the aggregated posterior distribution of h0 as the data.





Stacked RBM and BNs

- Prior does not cancel exactly the "explaining away" effect
- Do things wrongly anyway
- It can be proved that, a variational bound of the likelihood will improve whenever we add a new hidden layer.

Deep Belief Nets, a hybrid model of directed and undirected graph

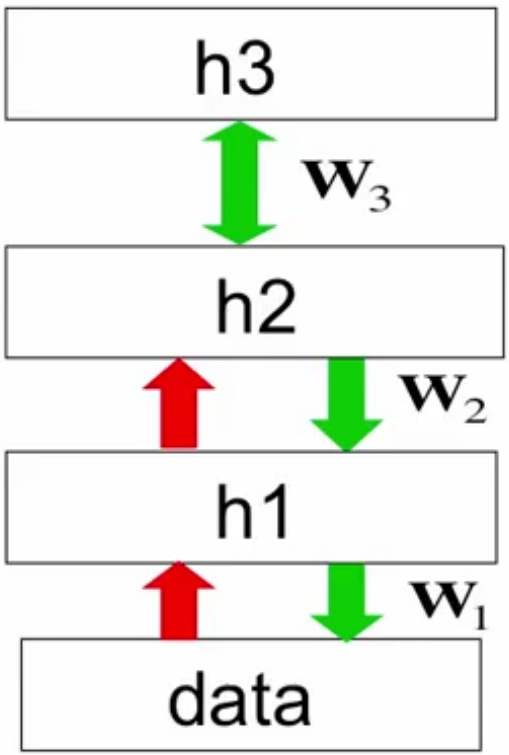
Learning, given data $\mathcal{D}$

- Learn stacked RBMs layer by layer greedily with CD-1
- Fine-tuning by the wake and sleep algorithm (why?)

Reconstructing data

- Start from random configurations of the top two layers
- Run the Markov chain for a long time
- Generate lower layers (including the data) in a causal fashion

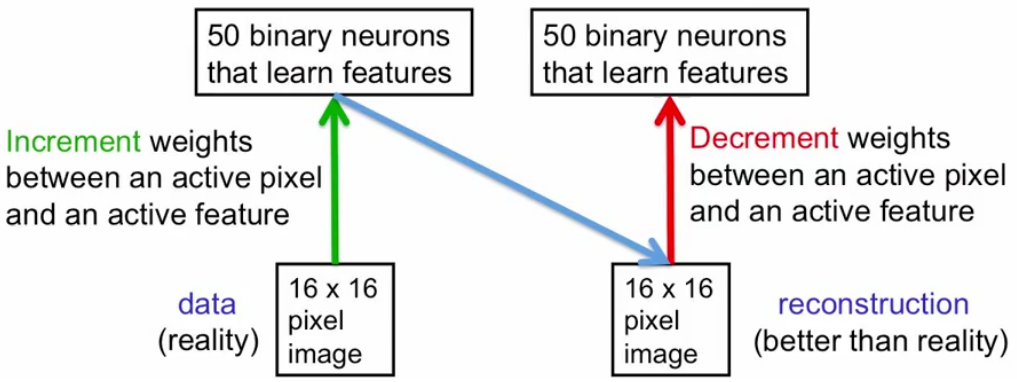
$\Rightarrow$ Theoretically incorrect, but works well

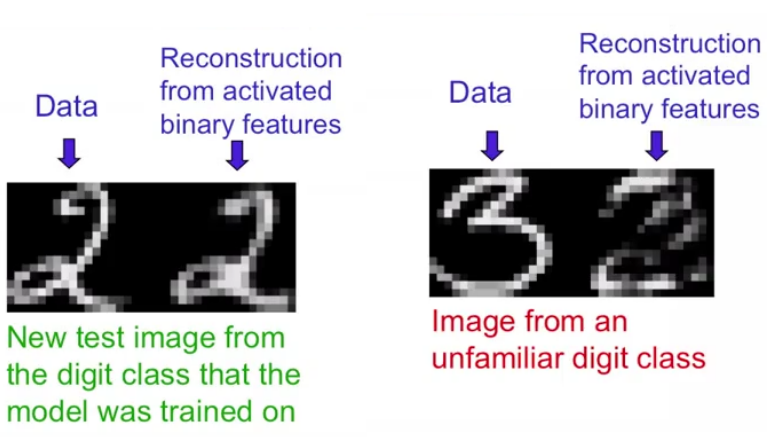


Examples

RBM Modeling MNIST Data

Modeling '2'

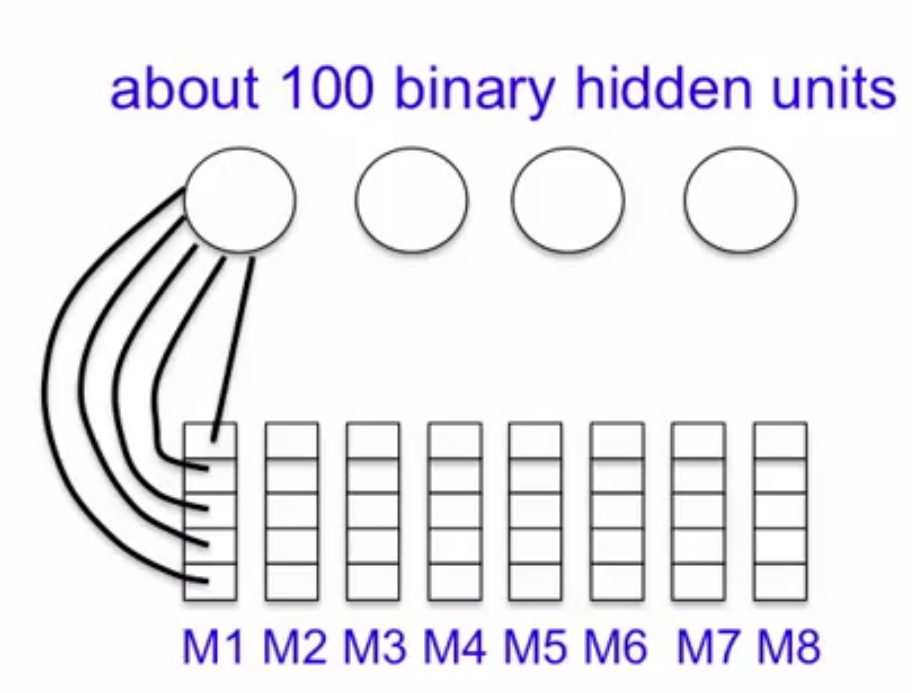




Modeling all 10 digits

...

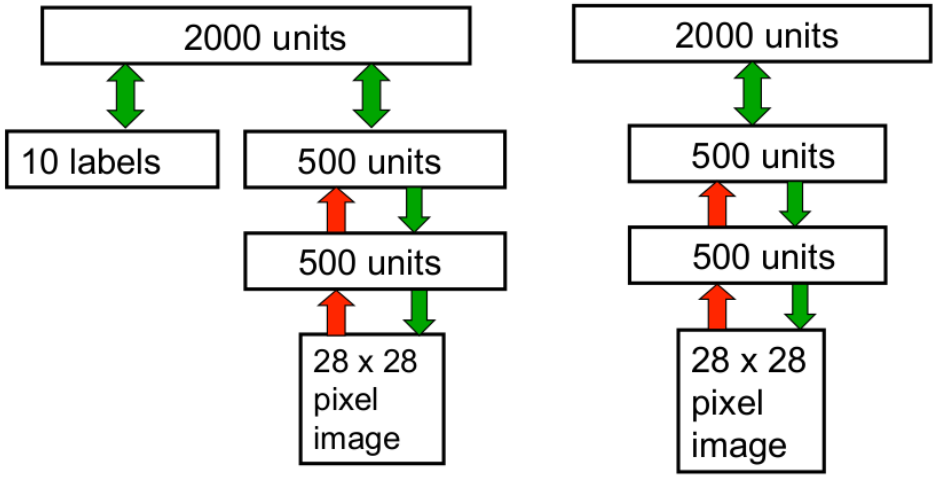
RBM for Collaborative Filtering



- RBM works about as good as matrix factorization, but gives different errors
- Combining RBM and MF is a big win ($1,000,000)

DBM Modeling MNIST Data

DBN Classifying MNIST Digits



- Pretraining
  - Training stacked RBMs layer by layer greedily
  - Fine-tuning with wake and sleep algorithm
- Supervised learning
  - Regard the DBN as a feed-forward neural network
  - Fine-tuning the connection weights

Results on the permutation-invariant MNIST task

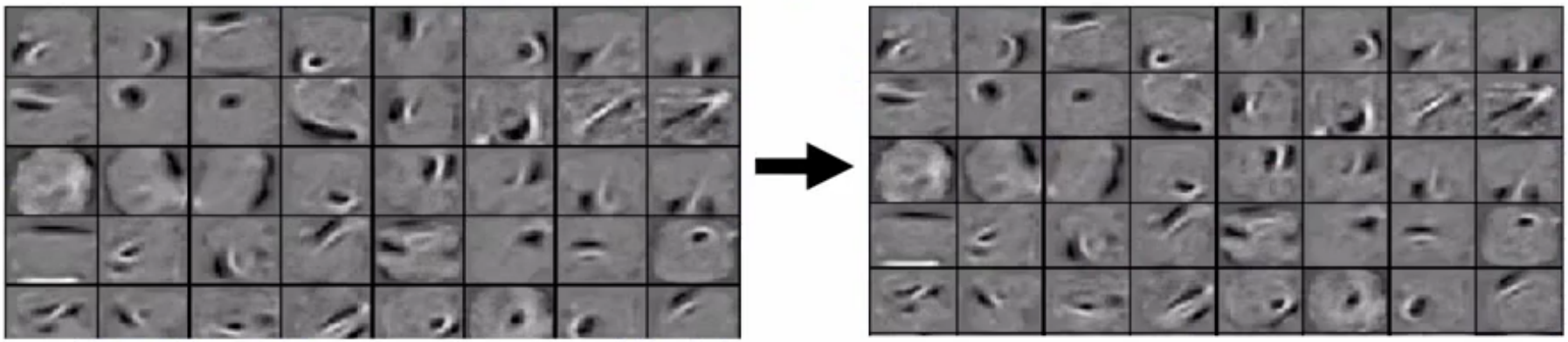
	Error rate
Backprop net with one or two hidden layers (Platt; Hinton)	1.6%
Backprop with L2 constraints on incoming weights	1.5%
Support Vector Machines (Decoste & Schoelkopf, 2002)	1.4%
Generative model of joint density of images and labels (+ generative fine-tuning)	1.25%
Generative model of unlabelled digits followed by gentle backpropagation (Hinton & Salakhutdinov, 2006)	1.15%→1.0%

Unsupervised "pre-training" also helps for models that have more data and better priors

- Ranzato et. al. (NIPS 2006) used an additional 600,000 distorted digits.
- They also used convolutional multilayer neural networks.

Back-propagation alone:	0.49%
Unsupervised layer-by-layer pre-training followed by backprop:	0.39% (record at the time)

What happens when fine-tuning?



Before fine-tuning

After fine-tuning

Thanks for listening!