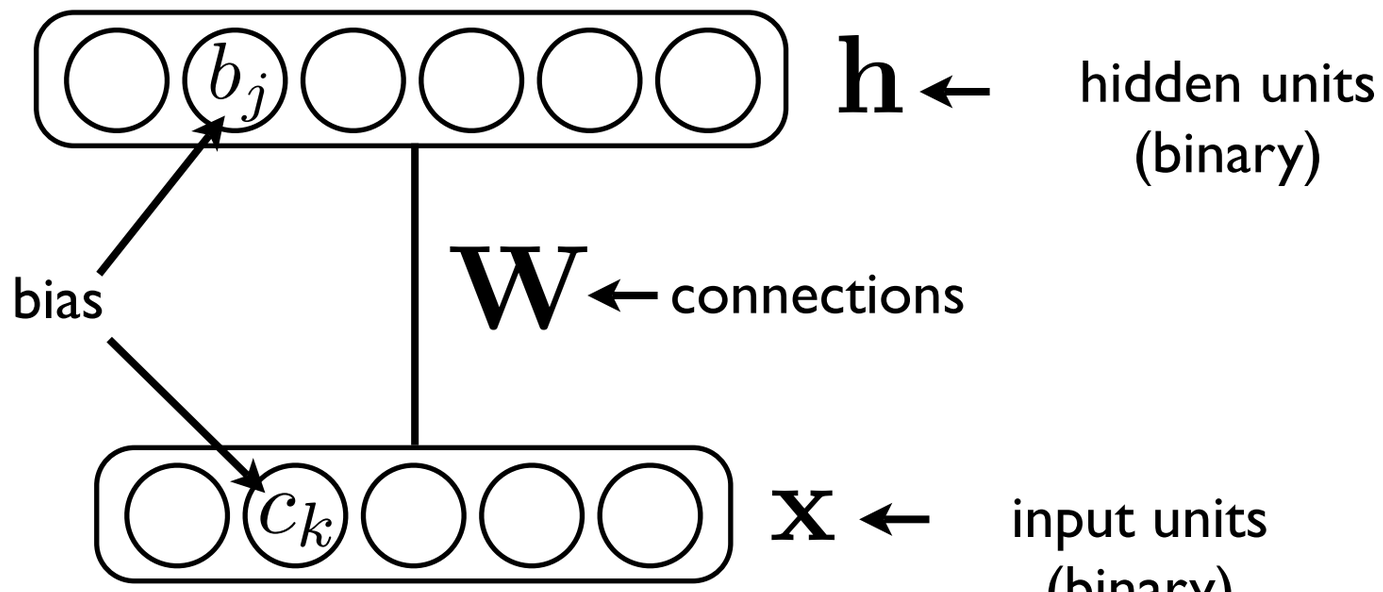
Au-delà de la Machine de Boltzmann Restreinte

Hugo Larochelle
University of Toronto

Introduction

- Restricted Boltzmann Machines (RBMs) are useful feature extractors
- They are mostly used to initialize deep feed-forward neural networks
- Can the Boltzmann machine modeling framework be useful on its own?

Restricted Boltzmann Machine



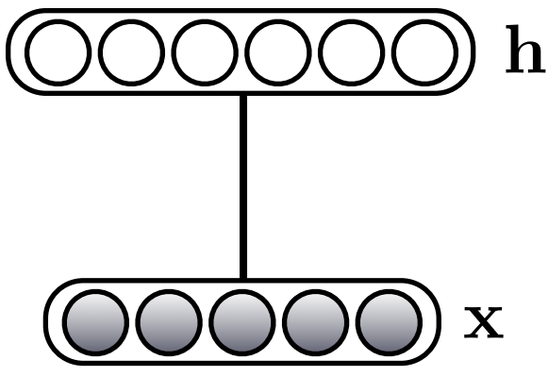
Energy function :

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$
$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution: $p(\mathbf{x}, \mathbf{h}) = \exp(-E(\mathbf{x}, \mathbf{h}))/Z$

intractable in general

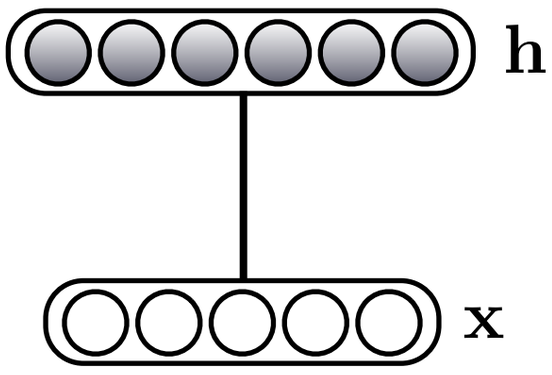
Inference



$$p(\mathbf{h}|\mathbf{x}) = \prod_j p(h_j|\mathbf{x})$$

$$p(h_j = 1|\mathbf{x}) = \frac{1}{1 + \exp(-(b_j + \mathbf{W}_j.\mathbf{x}))}$$

$$= \text{sigm}(b_j + \mathbf{W}_j.\mathbf{x})$$

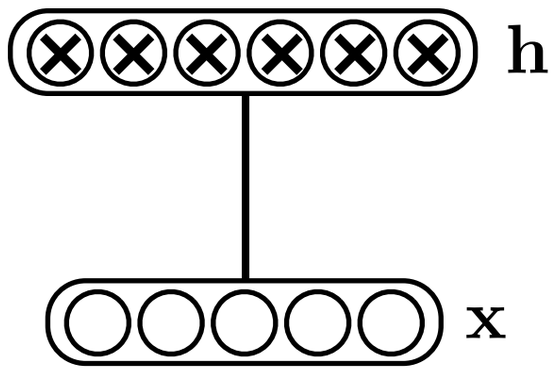


$$p(\mathbf{x}|\mathbf{h}) = \prod_k p(x_k|\mathbf{h})$$

$$p(x_k = 1|\mathbf{h}) = \frac{1}{1 + \exp(-(c_k + \mathbf{h}^\top \mathbf{W}_{.k}))}$$

$$= \text{sigm}(c_k + \mathbf{h}^\top \mathbf{W}_{.k})$$

Inference



$$p(\mathbf{x}) = \sum_{\mathbf{h} \in \{0,1\}^H} p(\mathbf{x}, \mathbf{h}) = \sum_{\mathbf{h} \in \{0,1\}^H} \exp(-E(\mathbf{x}, \mathbf{h}))/Z$$

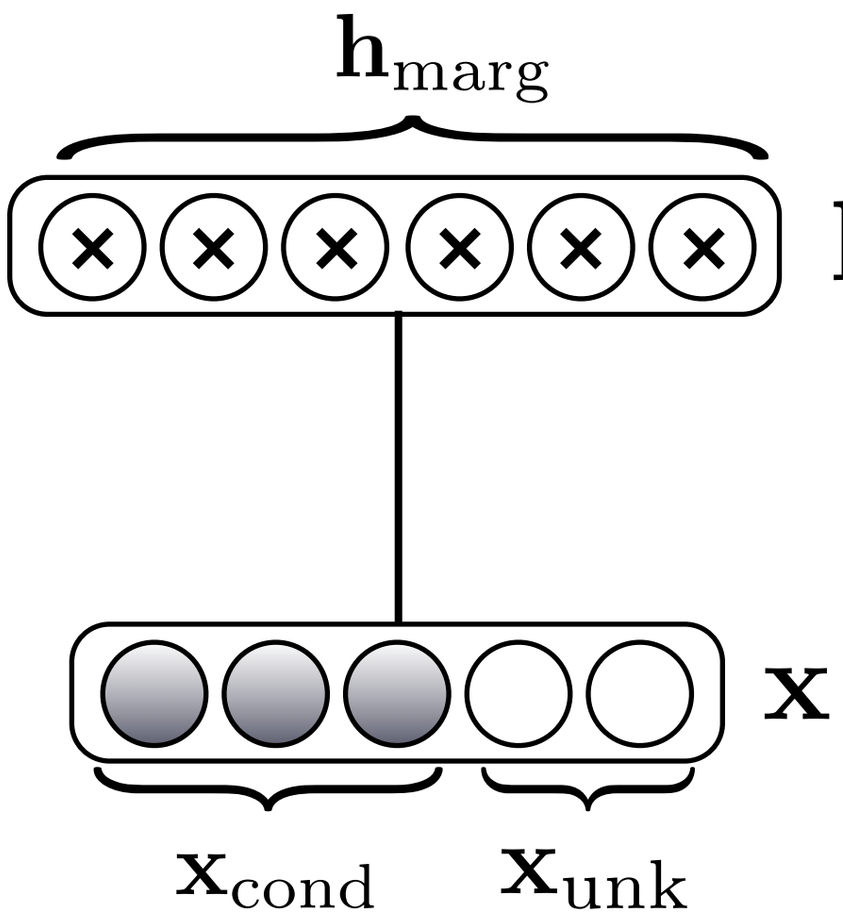
$$\sum_{\mathbf{h} \in \{0,1\}^H} \exp(-E(\mathbf{x}, \mathbf{h}))/Z = \sum_{h_1 \in \{0,1\}} \cdots \sum_{h_H \in \{0,1\}} \exp(-E(\mathbf{x}, \mathbf{h}))/Z$$

free energy

$$= \exp\left(\mathbf{c}^\top \mathbf{x} + \sum_{j=1}^{H} \log(1 + \exp(b_j + \mathbf{W}_j . \mathbf{x})) \right) /Z_F$$

$$= \exp(-F(\mathbf{x}))/Z_F$$

Inference: in general



$\mathbf{x}_{\mathrm{unk}}$: unknown variables, whose distribution we are interested in

$\mathbf{x}_{\mathrm{cond}}$: known variables, on which we condition

$\mathbf{h}_{\mathrm{marg}}$: unknown variables, over which we marginalize

$$p(\mathbf{x}_{\mathrm{unk}}|\mathbf{x}_{\mathrm{cond}}) = \frac{\sum_{\mathbf{h}_{\mathrm{marg}}} \exp(E(\mathbf{x}_{\mathrm{unk}}, \mathbf{x}_{\mathrm{cond}}, \mathbf{h}_{\mathrm{marg}}))}{\sum_{\mathbf{x}_{\mathrm{unk}}} \sum_{\mathbf{h}_{\mathrm{marg}}} \exp(E(\mathbf{x}_{\mathrm{unk}}, \mathbf{x}_{\mathrm{cond}}, \mathbf{h}_{\mathrm{marg}}))}$$

Learning

- To train an RBM, we minimize the negative log-likelihood (NLL) of the data

$$\mathcal{L}_{\text{gen}}(\mathcal{D}_{\text{train}}) = \frac{1}{n} \sum_{\mathbf{x}_t \in \mathcal{D}_{\text{train}}} -\log p(\mathbf{x}_t)$$

- We proceed by stochastic gradient descent

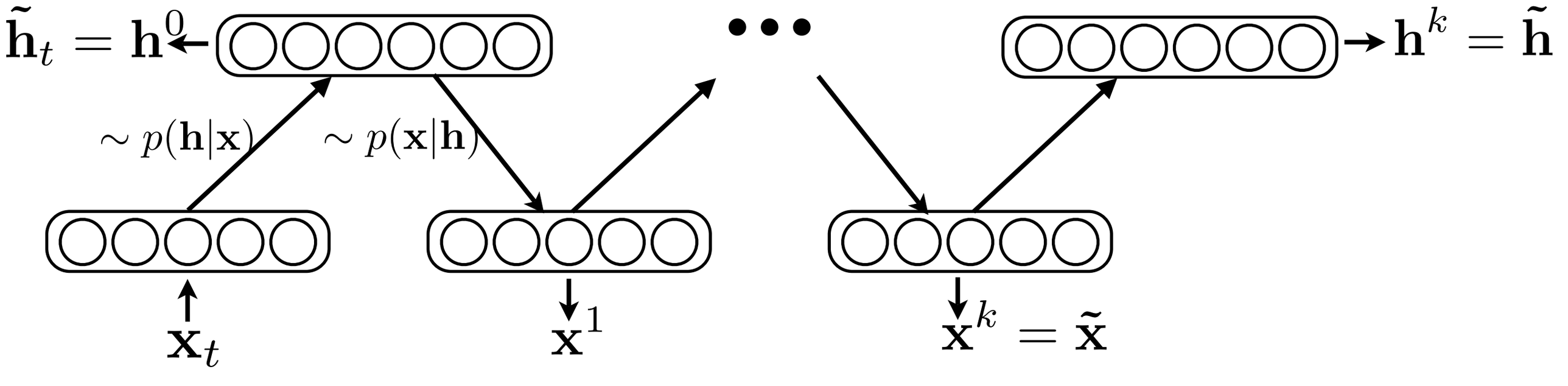
$$\frac{\partial -\log p(\mathbf{x}_t)}{\partial \theta} = \underbrace{\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x_t}, \mathbf{h})}{\partial \theta} \middle| \mathbf{x}_t\right]}_{\text{positive phase}} - \underbrace{\mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta}\right]}_{\text{negative phase}}$$

Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

- Idea:

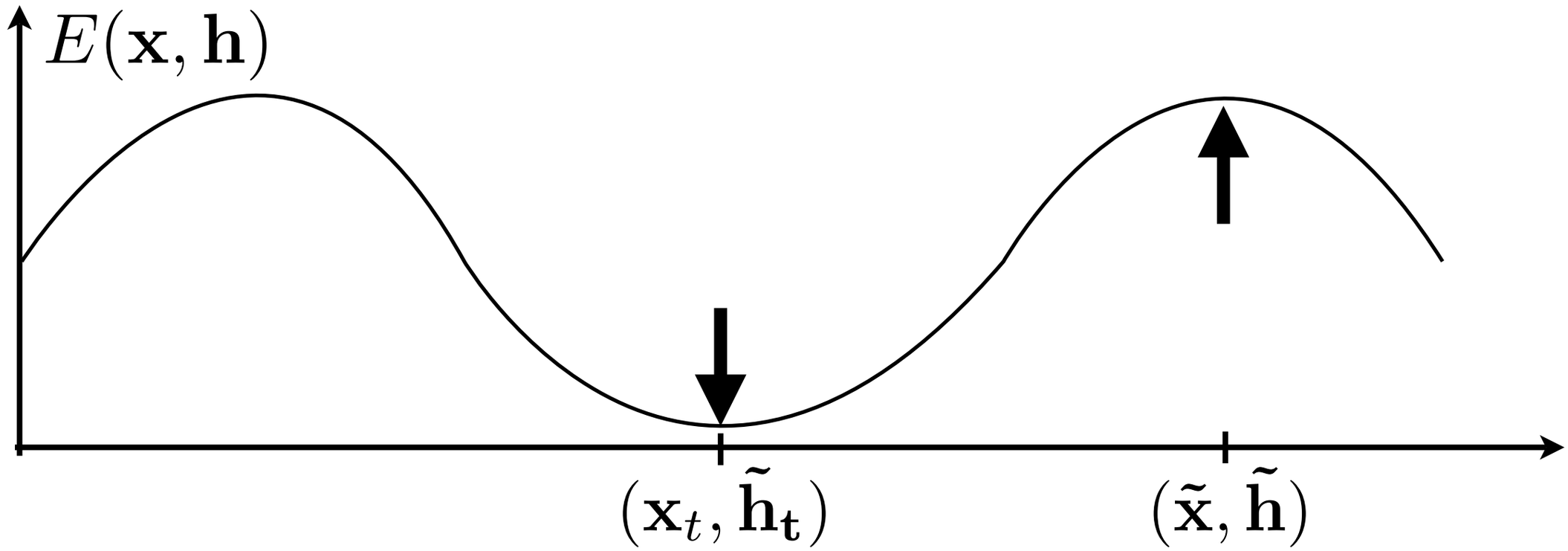
1. replace expectation by an evaluation at a single point $\tilde{\mathbf{x}}, \tilde{\mathbf{h}}$
2. obtain $\tilde{\mathbf{x}}, \mathbf{h}$ by MCMC sampling
3. start sampling chain at $\mathbf{x}_t$



Contrastive Divergence (CD)

(Hinton, Neural Computation, 2002)

$$\mathbb{E}_{\mathbf{h}}\left[\frac{\partial E(\mathbf{x}_t, \mathbf{h})}{\partial \theta}\middle|\mathbf{x}_t\right] \approx \frac{\partial E(\mathbf{x}_t, \tilde{\mathbf{h}}_\mathbf{t})}{\partial \theta} \qquad \mathbb{E}_{\mathbf{x},\mathbf{h}}\left[\frac{\partial E(\mathbf{x}, \mathbf{h})}{\partial \theta}\right] \approx \frac{\partial E(\tilde{\mathbf{x}}, \tilde{\mathbf{h}})}{\partial \theta}$$



Contrastive Divergence (CD)

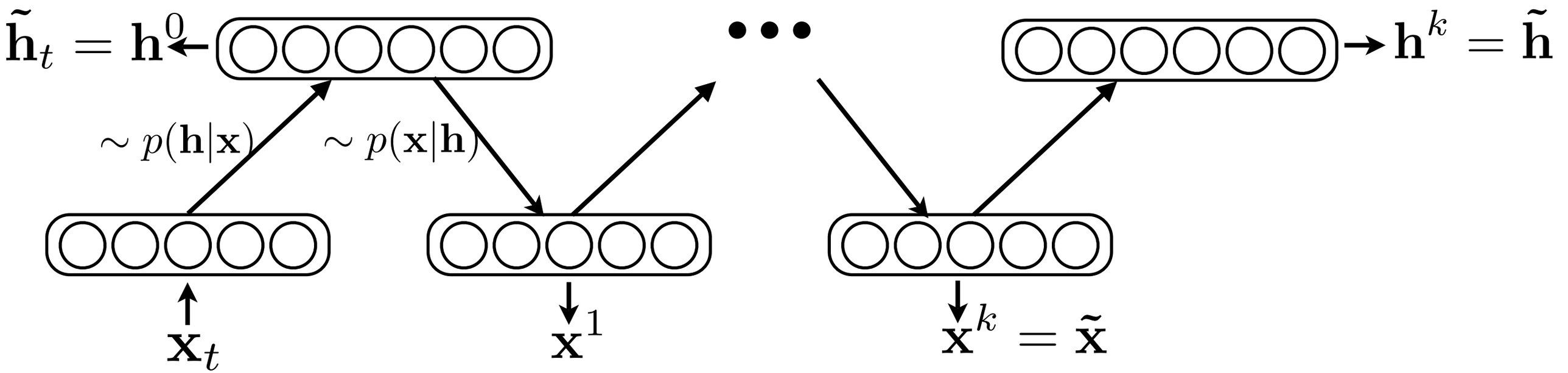
(Hinton, Neural Computation, 2002)

- CD-k: contrastive divergence with k Gibbs sampling steps
- In general, the bigger k is, the less biased is the estimate of the gradient
- In practice, k=1 works well
- For an RBM, the parameter updates won't actually need samples for $\mathbf{h}$

Other learning algorithm: Persistent CD (PCD)

(Tieleman, ICML2008)

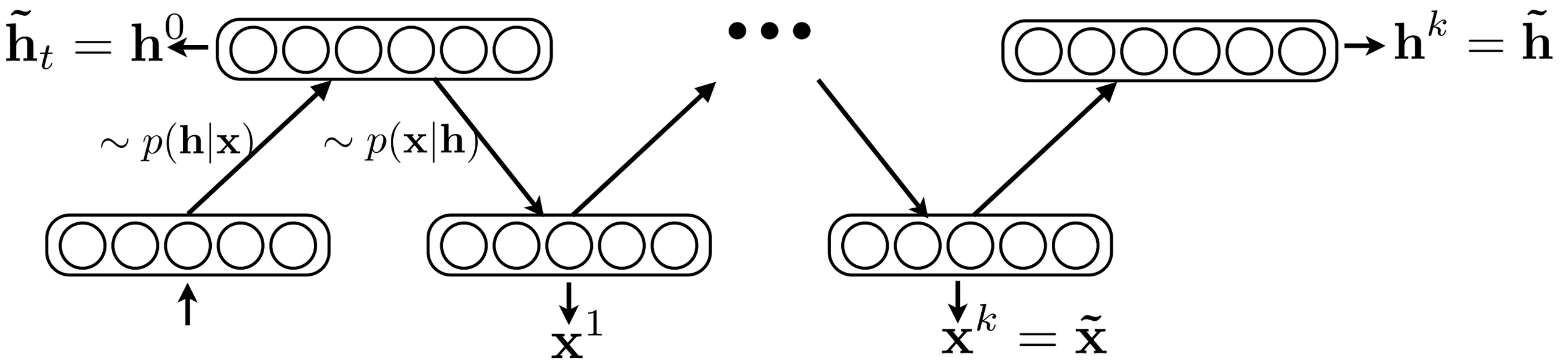
- Idea: instead of always reinitializing the Gibbs chain to $\mathbf{x}_t$, use $\tilde{\mathbf{x}}$ from the previous iteration instead



Other learning algorithm: Persistent CD (PCD)

(Tieleman, ICML2008)

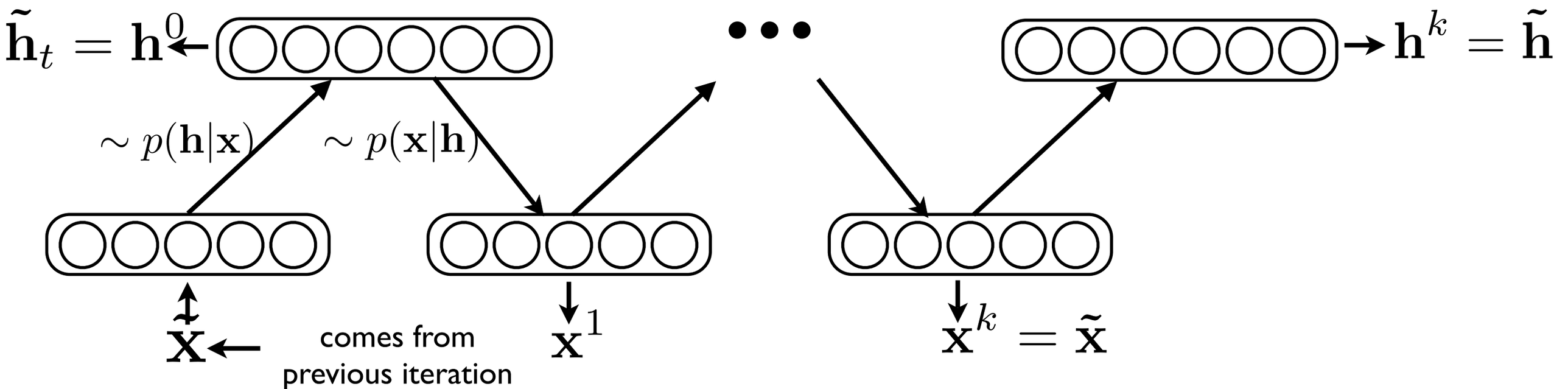
- Idea: instead of always reinitializing the Gibbs chain to $\mathbf{x}_t$, use $\tilde{\mathbf{x}}$ from the previous iteration instead



Other learning algorithm: Persistent CD (PCD)

(Tieleman, ICML2008)

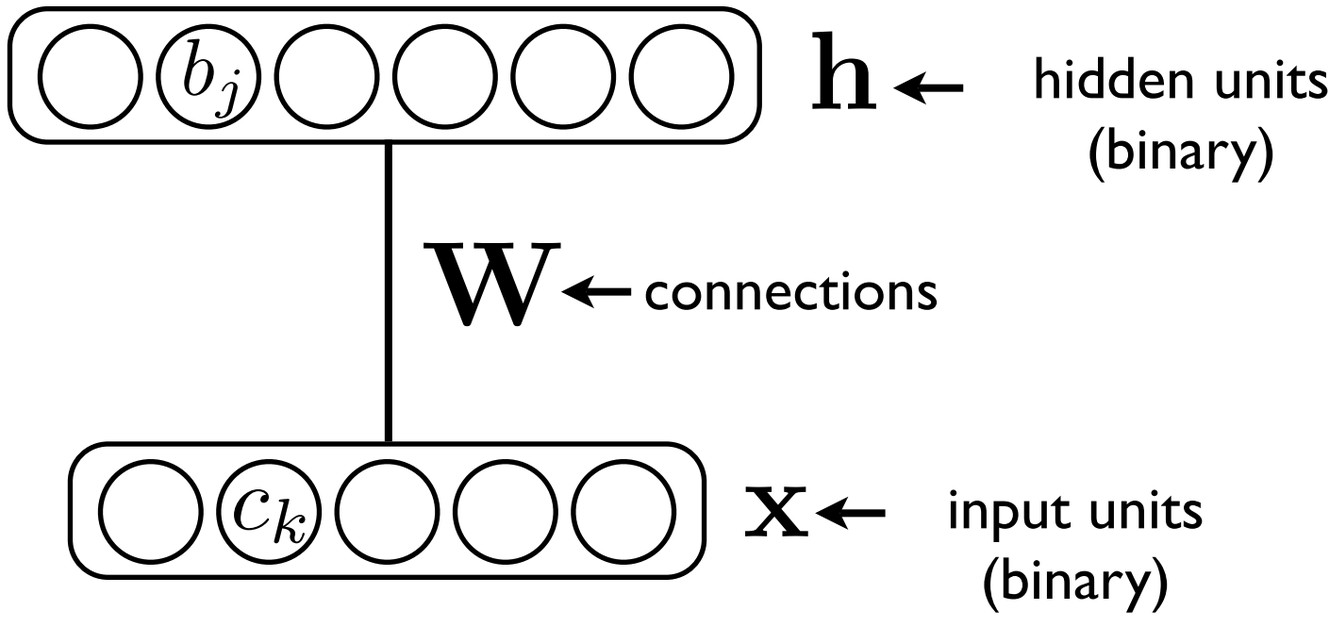
- Idea: instead of always reinitializing the Gibbs chain to $\mathbf{x}_t$, use $\tilde{\mathbf{x}}$ from the previous iteration instead



Plan

- RBMs can initialize deep neural networks, but not a Boltzmann machine anymore
- Three extensions to RBMs:
  - ★ Classification RBM
  - ★ Restricted Boltzmann Forest
  - ★ Deep Boltzmann Machine (DBM)

Classification Restricted Boltzmann Machine

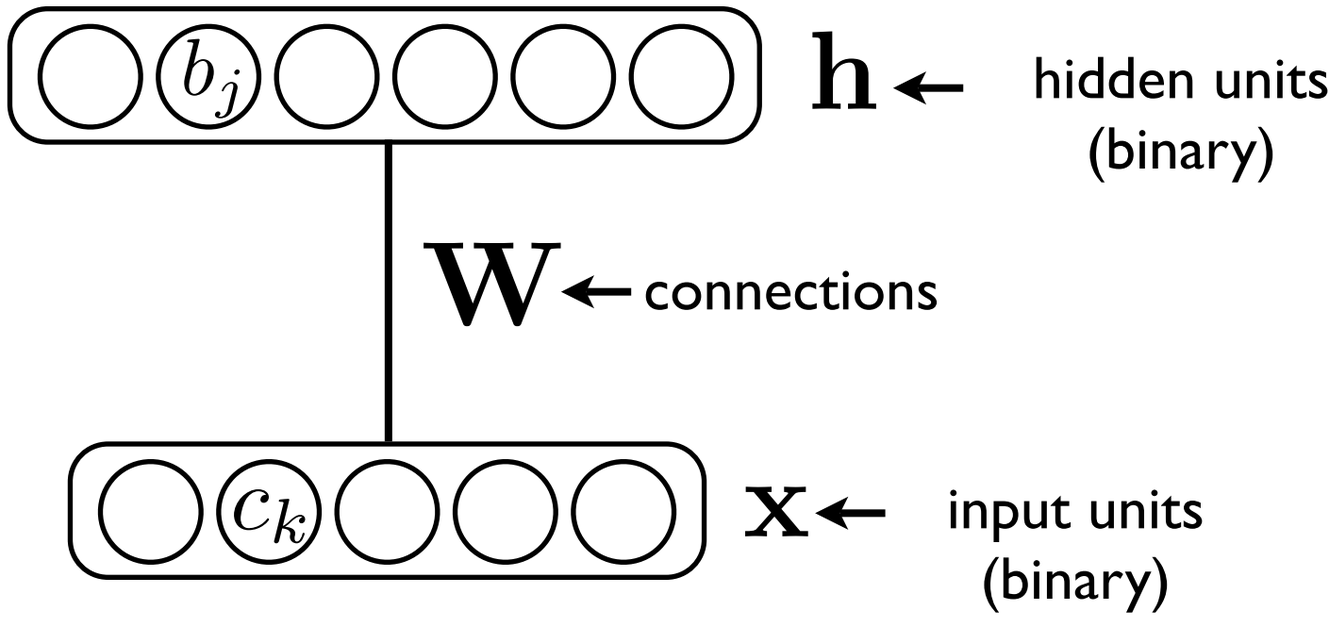


Energy function :

$$E(\mathbf{x}, \mathbf{h}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$
$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution: $p(\mathbf{x}, \mathbf{h}) = \exp(-E(\mathbf{x}, \mathbf{h}))/Z$

Classification Restricted Boltzmann Machine



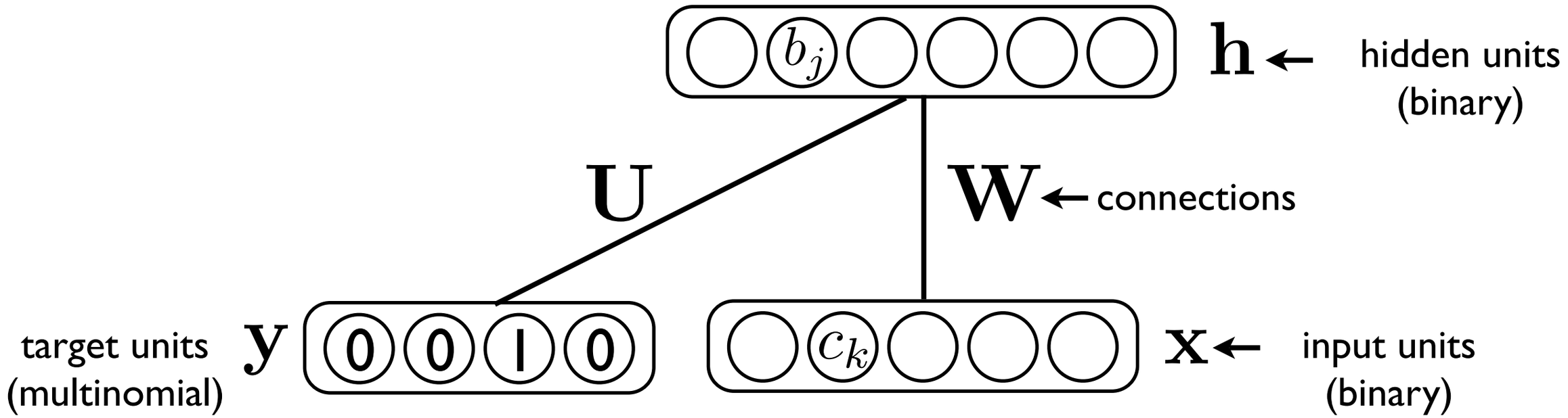
Energy function :

$$= -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h}$$

$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$

Distribution:

Classification Restricted Boltzmann Machine

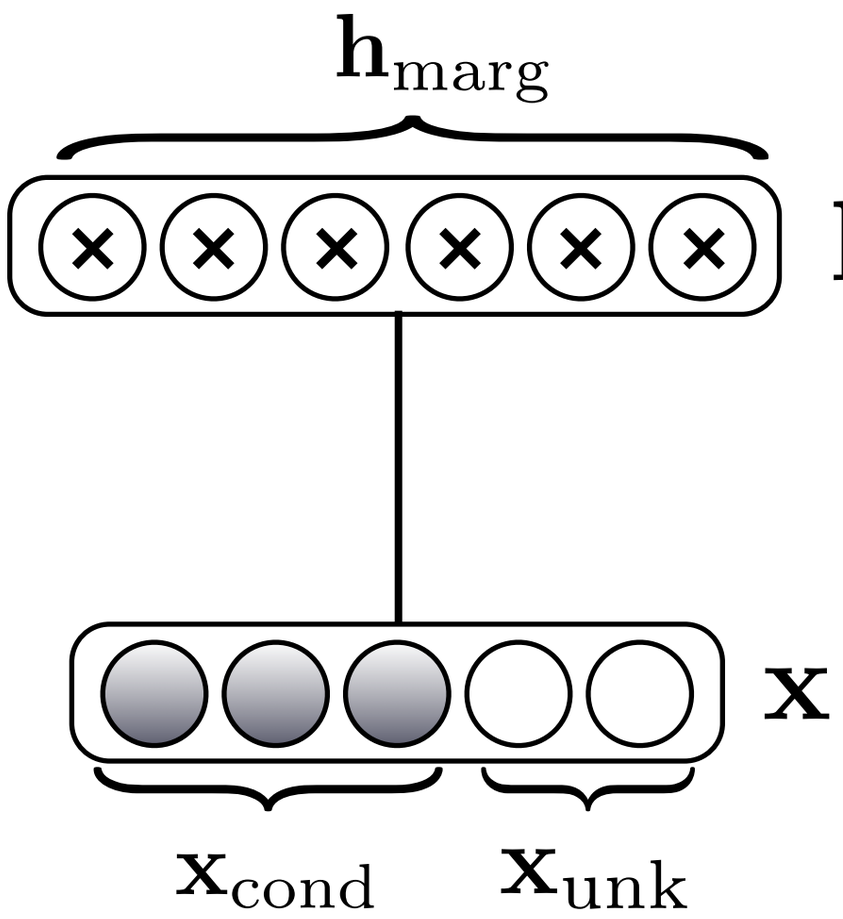


Energy function:

$$E(\mathbf{x}, \mathbf{h}, \mathbf{y}) = -\mathbf{h}^\top \mathbf{W}\mathbf{x} - \mathbf{c}^\top \mathbf{x} - \mathbf{b}^\top \mathbf{h} - \mathbf{h}^\top \mathbf{U}\mathbf{y} - \mathbf{a}^\top \mathbf{y}$$
$$= -\sum_{jk} W_{jk} h_j x_k - \sum_k c_k x_k - \sum_j b_j h_j$$
$$- \sum_{jl} U_{jl} h_j y_l - \sum_l a_l y_l$$

Distribution: $p(\mathbf{x}, \mathbf{h}, \mathbf{y}) = \exp(-E(\mathbf{x}, \mathbf{h}, \mathbf{y}))/Z$

Inference: in general



$\mathbf{x}_{\mathrm{unk}}$: unknown variables, whose distribution we are interested in

$\mathbf{x}_{\mathrm{cond}}$: known variables, on which we condition

$\mathbf{h}_{\mathrm{marg}}$: unknown variables, over which we marginalize

$$p(\mathbf{x}_{\mathrm{unk}}|\mathbf{x}_{\mathrm{cond}}) = \frac{\sum_{\mathbf{h}_{\mathrm{marg}}} \exp(E(\mathbf{x}_{\mathrm{unk}}, \mathbf{x}_{\mathrm{cond}}, \mathbf{h}_{\mathrm{marg}}))}{\sum_{\mathbf{x}_{\mathrm{unk}}} \sum_{\mathbf{h}_{\mathrm{marg}}} \exp(E(\mathbf{x}_{\mathrm{unk}}, \mathbf{x}_{\mathrm{cond}}, \mathbf{h}_{\mathrm{marg}}))}$$

Discriminative training (DRBM)

(Larochelle and Bengio, ICML2008)

- What if we are only interested in classification:

$$\mathcal{L}_{disc}(\mathcal{D}_{train}) = -\sum_{i=1}^{|\mathcal{D}_{train}|} \log p(y_i|\mathbf{x}_i)$$

- Since $p(y|\mathbf{x})$ can be computed exactly, we can also compute the discriminative gradient exactly:

$$-\frac{\partial \log p(y_i|\mathbf{x}_i)}{\partial \theta} = \frac{\partial}{\partial \theta} F(y_i, \mathbf{x}_i) - \mathbb{E}_{y|\mathbf{x}_i}\left[\frac{\partial}{\partial \theta} F(y, \mathbf{x}_i)\right]$$

where $F(y, \mathbf{x}) = -d_y - \sum_{j=1}^{H} \log\left(1 + \exp\left(c_j + U_{jy} + \sum_{i=1}^{d} W_{ji} x_i\right)\right)$

Hybrid generative/discriminative training (HDRBM)

(Larochelle and Bengio, ICML2008)

- We can explore the space between generative and discriminative training with α (Bouchard & Triggs, 2004)

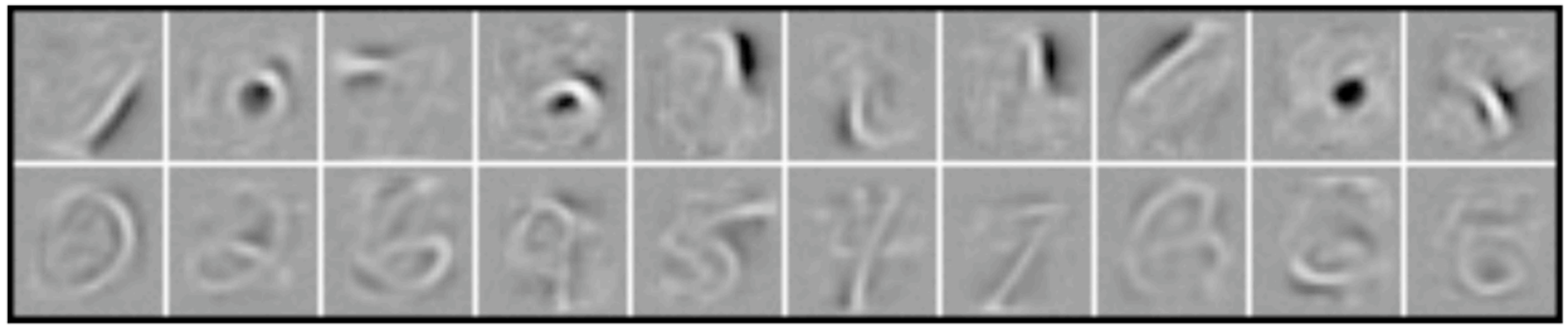
$$\mathcal{L}_{hybrid}(\mathcal{D}_{train}) = \mathcal{L}_{disc}(\mathcal{D}_{train}) + \alpha \mathcal{L}_{gen}(\mathcal{D}_{train})$$

- Perform two updates, one according to each criterion
- Generative criterion acts as a data-dependent regulariser

Experiment: MNIST

Model	Error
RBM ($\lambda = 0.005$, $n = 6000$)	3.39%
DRBM ($\lambda = 0.05$, $n = 500$)	1.81%
RBM+NNet	1.41%
HDRBM ($\alpha = 0.01$, $\lambda = 0.05$, $n = 1500$)	1.28%
Sparse HDRBM (idem + $n = 3000$, $\delta = 10^{-4}$)	**1.16%**
SVM	1.40%
NNet	1.93%

Subset of filters learned by the HDRBM on the MNIST dataset



Some filters act as edge detectors (first row), some other filters are more specific to a particular digit shape (second row)

Experiment: 20newsgroup

Model	Error
RBM ($\lambda = 0.0005$, $n = 1000$)	24.9%
DRBM ($\lambda = 0.0005$, $n = 50$)	27.6%
RBM+NNet	26.8%
HDRBM ($\alpha = 0.005$, $\lambda = 0.1$, $n = 1000$)	**23.8%**
SVM	32.8%
NNet	28.2%

Newsgroup	Most influencial words
comp.sys.ibm.pc.hardware	dos, ide, adaptec, pc, config, irq, vlb, bios, scsi, esdi,
comp.sys.mac.hardware	apple, mac, quadra, powerbook, lc, pds, centris
alt.atheism	bible, atheists, benedikt, atheism, religion
talk.politics.guns	firearms, handgun, firearm, gun, rkba, concealed
rec.sport.hockey	playoffs, penguins, didn, playoff, game, out, play

Restricted Boltzmann Forest

(Larochelle et al., Neural Computation, to appear)

- How to obtain a tractable density estimator from an RBM?
- Answer: constrain RBM such that we can compute Z!

$$Z = \sum_{\mathbf{h} \in \{0,1\}^H} \sum_{\mathbf{x} \in \{0,1\}^d} \exp(-E(\mathbf{x}, \mathbf{h}))$$

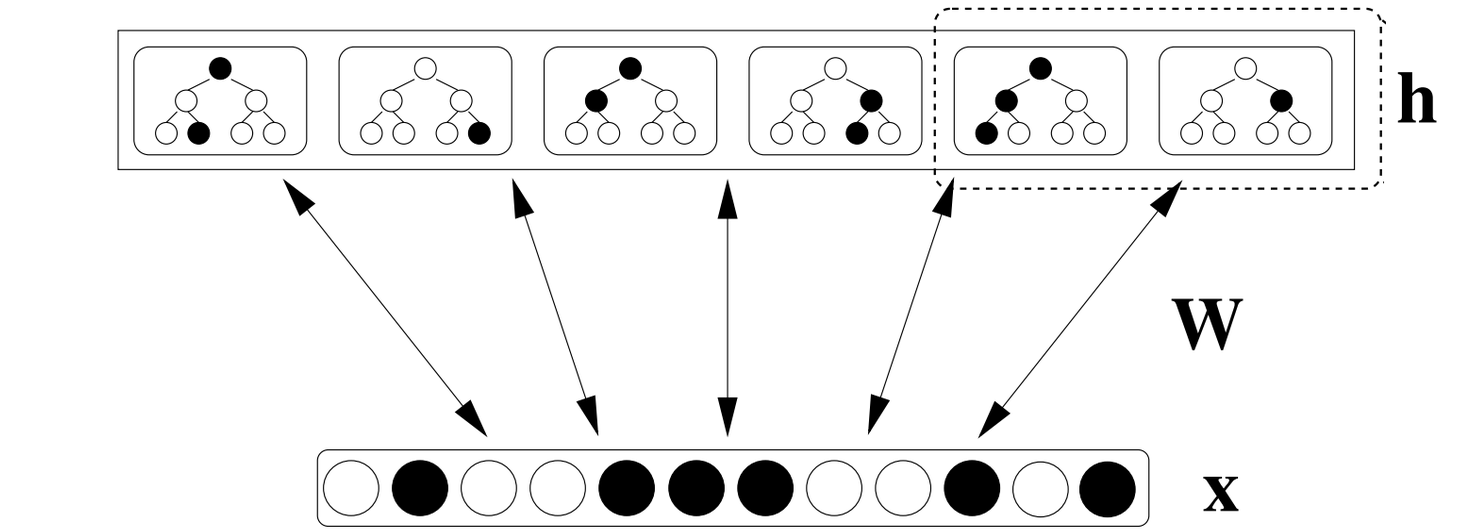
$$Z = \sum_{\mathbf{h} \in \{0,1\}^H} \exp(-F(\mathbf{h}))$$

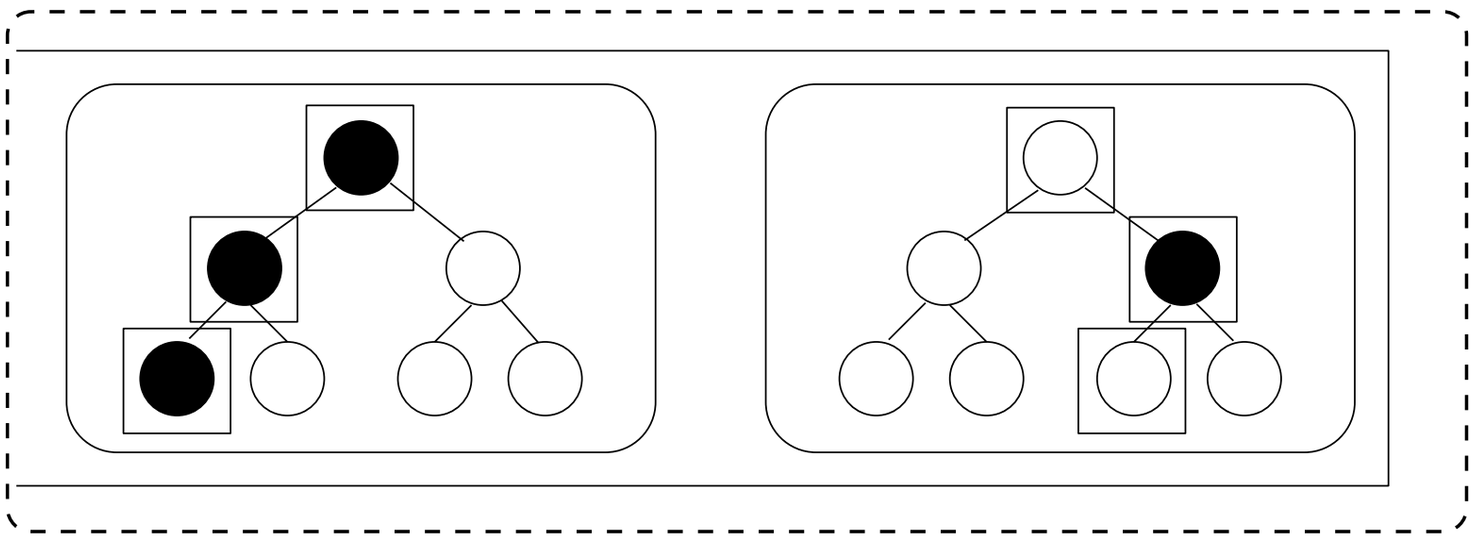
number of hidden units must be small

Restricted Boltzmann Forests

(Larochelle et al., Neural Computation, to appear)

- Alternative: constrain activations using a tree





Restricted Boltzmann Forests

(Larochelle et al., Neural Computation, to appear)

- Conditioned on $\mathbf{x}$, the distribution of each tree is independant
- Using belief propagation (sum-product) can:
  - ★ sample hidden units
  - ★ compute unit expectations (probabilities)
  - ★ sum out all hidden units

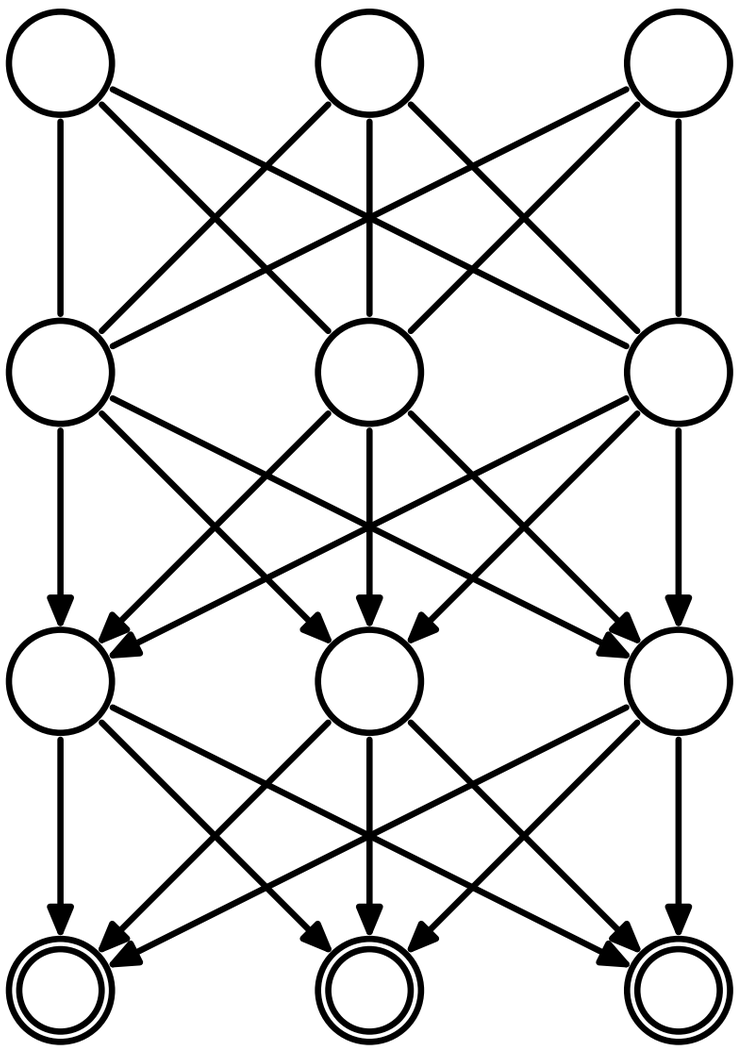
Experiments

Dataset	Nb. inputs	Set sizes (Train,Valid,Test)	Diff. NLL RBM	Diff. NLL RBForest
adult	123	5000,1414,26147	**4.18 ± 0.11**	**4.12 ± 0.11**
connect-4	126	16000,4000,47557	**0.75 ± 0.04**	**0.59 ± 0.04**
dna	180	1400,600,1186	**1.29 ± 0.72**	**1.39 ± 0.73**
mushrooms	112	2000,500,5624	*-0.69 ± 0.13*	0.042 ± 0.12
nips-0-12	500	400,100,1240	**12.65 ± 1.55**	**12.61 ± 1.55**
ocr-letter	128	32152,10000,10000	*-2.49 ± 0.44*	**3.78 ± 0.43**
rcv1	150	40000,10000,150000	*-1.29 ± 0.16*	**0.56 ± 0.16**
web	300	14000,3188,32561	**0.78 ± 0.30**	-0.15 ± 0.31

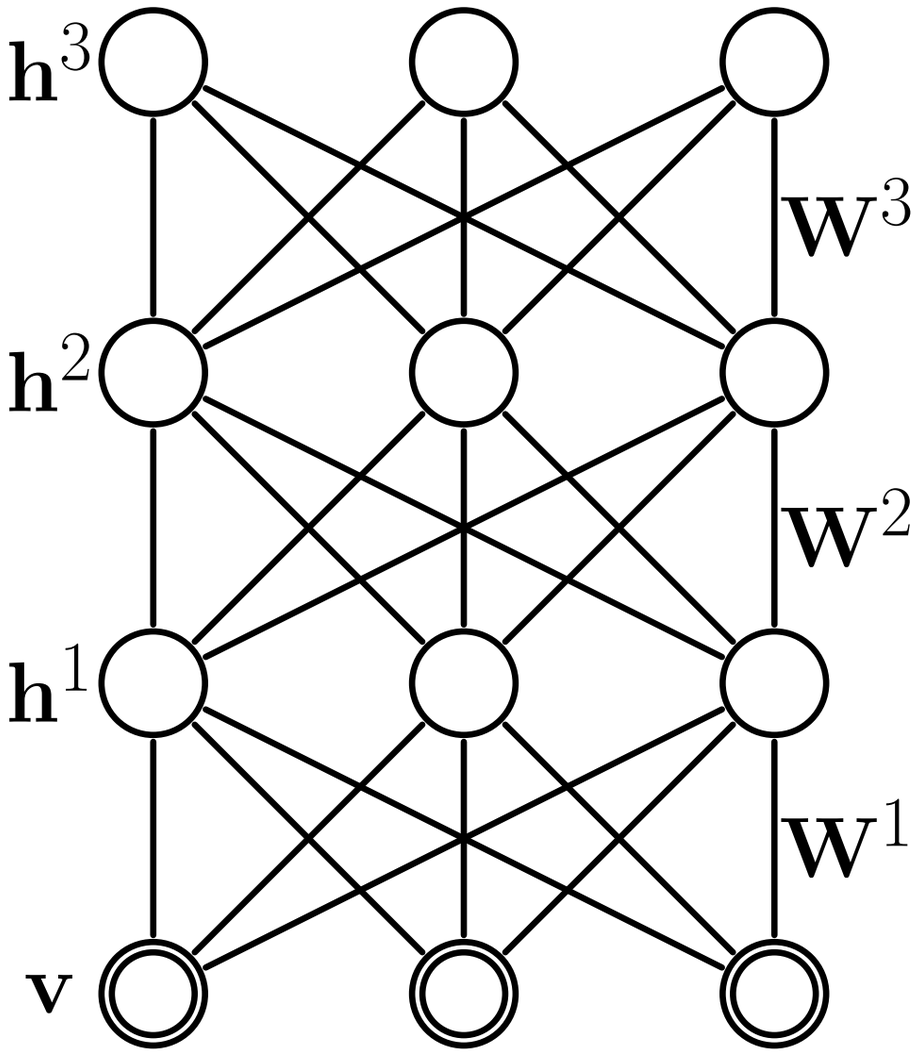
Deep Boltzmann Machines (DBM)

(Salakhutdinov and Hinton, AISTATS2009)

Deep Belief Network



Deep Boltzmann Machine



$$E(\mathbf{v}, \mathbf{h}; \theta) = -\mathbf{v}^\top \mathbf{W}^1 \mathbf{h}^1 - \mathbf{h}^{1\top} \mathbf{W}^2 \mathbf{h}^2 - \mathbf{h}^{2\top} \mathbf{W}^3 \mathbf{h}^3$$

Learning

- Learning under a DBM requires a similar gradient

$$\frac{\partial - \log P(\mathbf{v}_t)}{\partial \theta} = \underbrace{\mathbb{E}_{\mathbf{h}} \left[\frac{\partial E(\mathbf{v_t}, \mathbf{h}; \theta)}{\partial \theta} \middle| \mathbf{v}_t \right]}_{\text{positive phase}} - \underbrace{\mathbb{E}_{\mathbf{v},\mathbf{h}} \left[\frac{\partial E(\mathbf{v}, \mathbf{h}; \theta)}{\partial \theta} \right]}_{\text{negative phase}}$$

- We use Persistent CD for the negative phase
- The positive phase now must be approximated

Learning

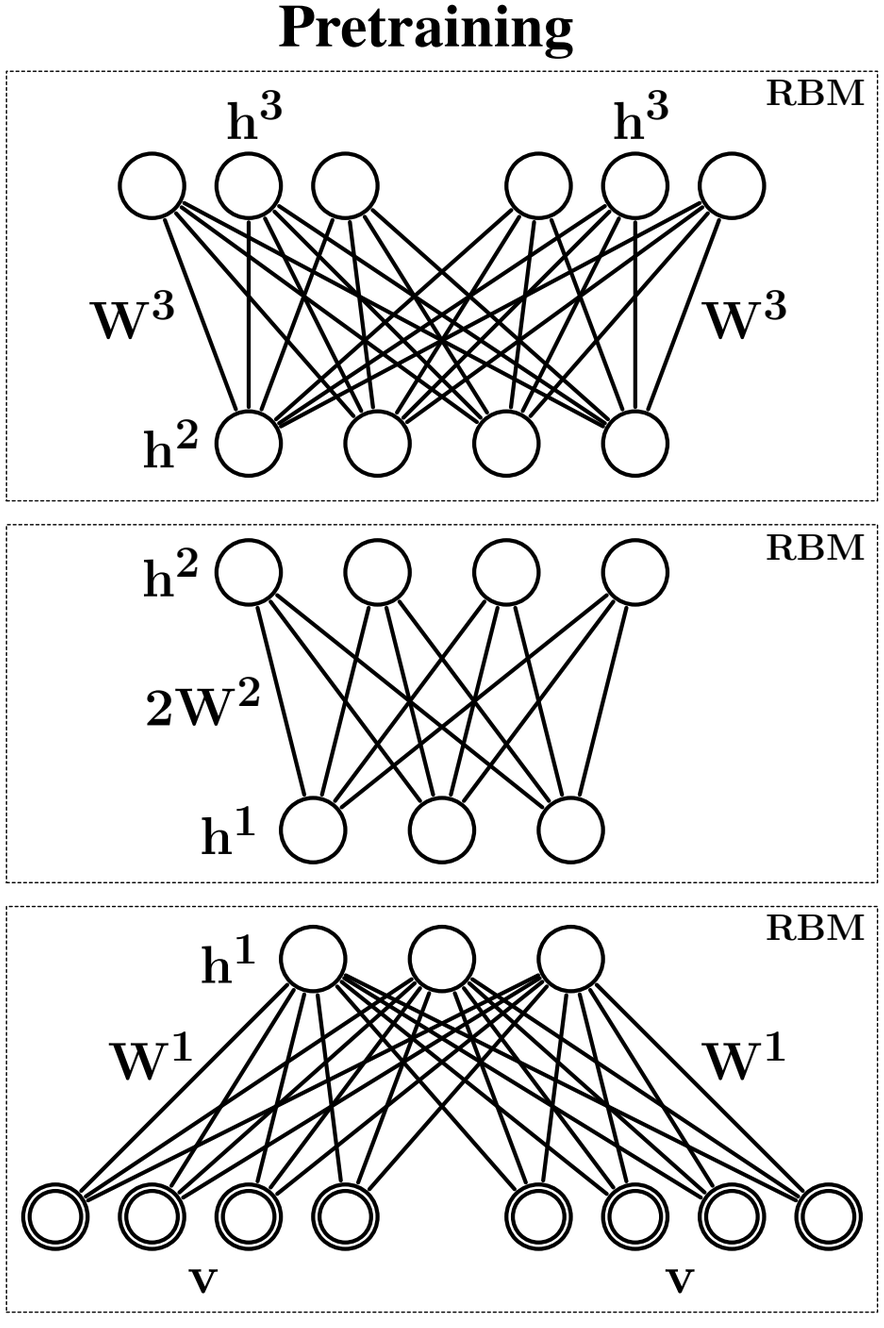
- In positive phase, a variational approach is used

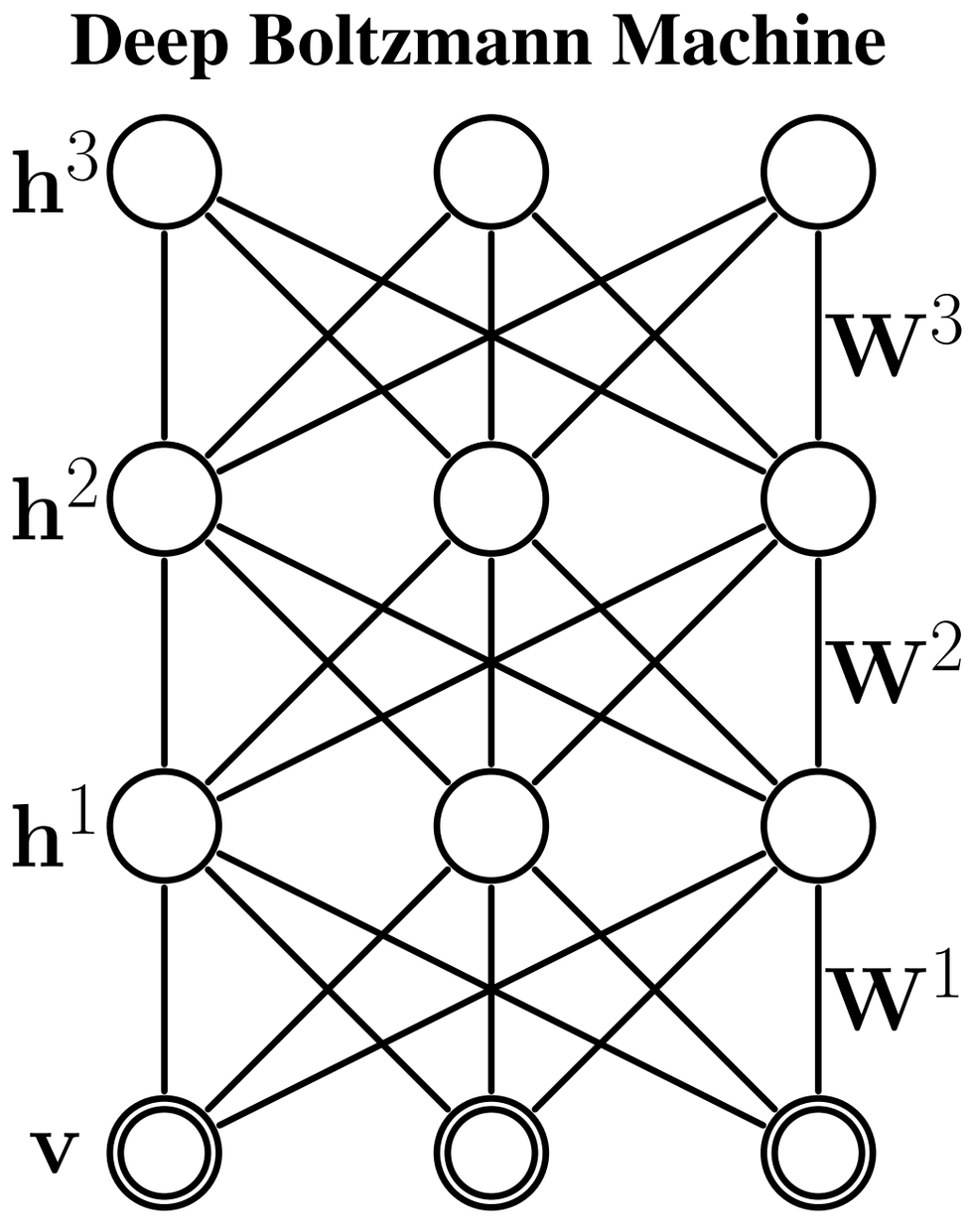
$$Q^{MF}(\mathbf{h}|\mathbf{v};\boldsymbol{\mu}) = \prod_{j=1}^{F_1}\prod_{k=1}^{F_2}\prod_{m=1}^{F_3} q(h_j^1)q(h_k^2)q(h_m^3) \quad \text{where } q(h_i^l = 1) = \mu_i^l \text{ for } l = 1,2,3$$

Procedure for minimizing $\mathrm{KL}(P(\mathbf{h}|\mathbf{v}), Q^{MF}(\mathbf{h}|\mathbf{v};\mu))$

$$\begin{cases} \mu_j^1 \leftarrow \sigma\left(\sum_{i=1}^{D} W_{ij}^1 v_i + \sum_{k=1}^{F_2} W_{jk}^2 \mu_k^2\right), \\ \mu_k^2 \leftarrow \sigma\left(\sum_{j=1}^{F_1} W_{jk}^2 \mu_j^1 + \sum_{m=1}^{F_3} W_{km}^3 \mu_m^3\right), \\ \mu_m^3 \leftarrow \sigma\left(\sum_{k=1}^{F_2} W_{km}^3 \mu_k^2\right). \end{cases}$$

Pretraining





Efficient Learning of Deep Boltzmann Machines

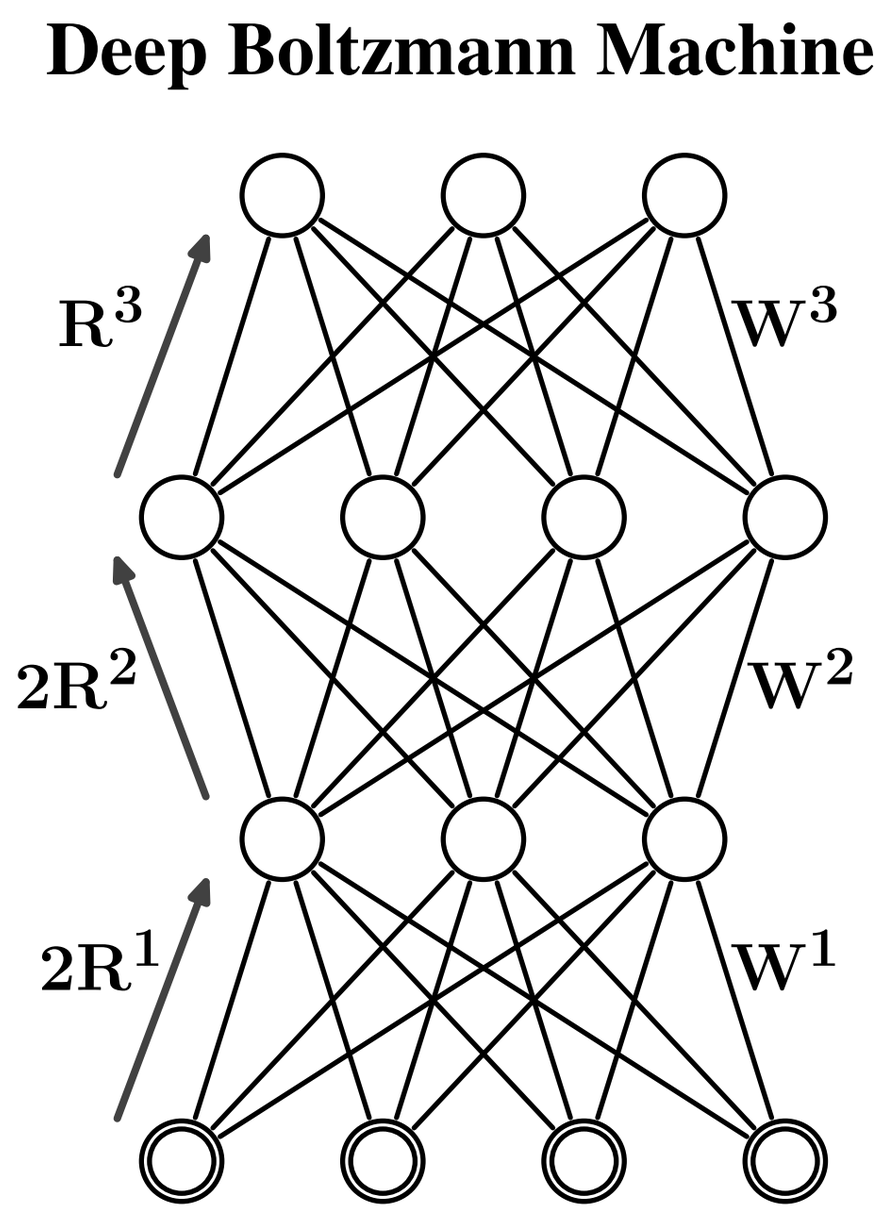
(Salakhutdinov and Larochelle, AISTATS2010)

- Unfortunately, the learning procedure described so far is quite slow
- The major bottleneck is variational inference, which requires around 25 passes up and down the network before converging

Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS2010)

- Idea: suppose that we have relatively good "mean-field" recognition net
- Could initialize mean-field with recognition net output



Efficient Learning of Deep Boltzmann Machines

(Salakhutdinov and Larochelle, AISTATS2010)

- Pseudocode

positive phase:

1. get prediction ν from recognition net
2. initialize mean-field to $\mu \leftarrow \nu$ and do only a few steps of mean-field

negative phase:

3. do Persistent CD
4. update DBM parameters
5. update recognition net parameters
6. go back to 1 until converged

Experiments

Classification

no top-down process

Dataset	DBM inference procedures						DBN	SVM	K-NN
	MF-0	MF-1	MFRec-1	MF-5	MFRec-5	MF-Full			
MNIST	1.38%	1.15%	**1.00%**	**1.01%**	**0.96%**	**0.95%**	1.17%	1.40%	3.09%
OCR letters	**8.68%**	**8.44%**	**8.40%**	**8.50%**	**8.48%**	**8.58%**	9.68%	9.70%	18.92%
NORB	9.32%	7.96%	**7.62%**	**7.67%**	**7.46%**	**7.23%**	8.31%	11.60%	18.40%

- Inference speed on NORB (per 100 examples)
  - ★ MF-Rec1: 0.69 sec
  - ★ MF-5: 3.20 sec
  - ★ MF-full: 16.00 sec
- Code is available online!

Conclusion

- Boltzmann machines provide a flexible framework for modeling data with latent variables
- Using tools from graphical models, it is possible to develop many extensions for the RBM
- Other variations:
  - ★ Conditional RBM (sequential data)
  - ★ Higher order RBM
  - ★ and more!

Merci!