Charu C. Aggarwal
IBM T J Watson Research Center
Yorktown Heights, NY

Restricted Boltzmann Machines

Neural Networks and Deep Learning, Springer, 2018
Chapter 6

Restricted Boltzmann Machines

- Most of the neural architectures map inputs to outputs.
  - Ideal for supervised models.
  - Autoencoders can be used for unsupervised models by replicating the output.
- Restricted Boltzmann machines are borrowed from probabilistic graphical models.
  - Graph of probabilistic dependencies between *binary* states that are *outcomes* of distributions.
  - Binary training data provides some examples of states.
  - Ideal for unsupervised models.

Key Differences from Conventional Neural Networks

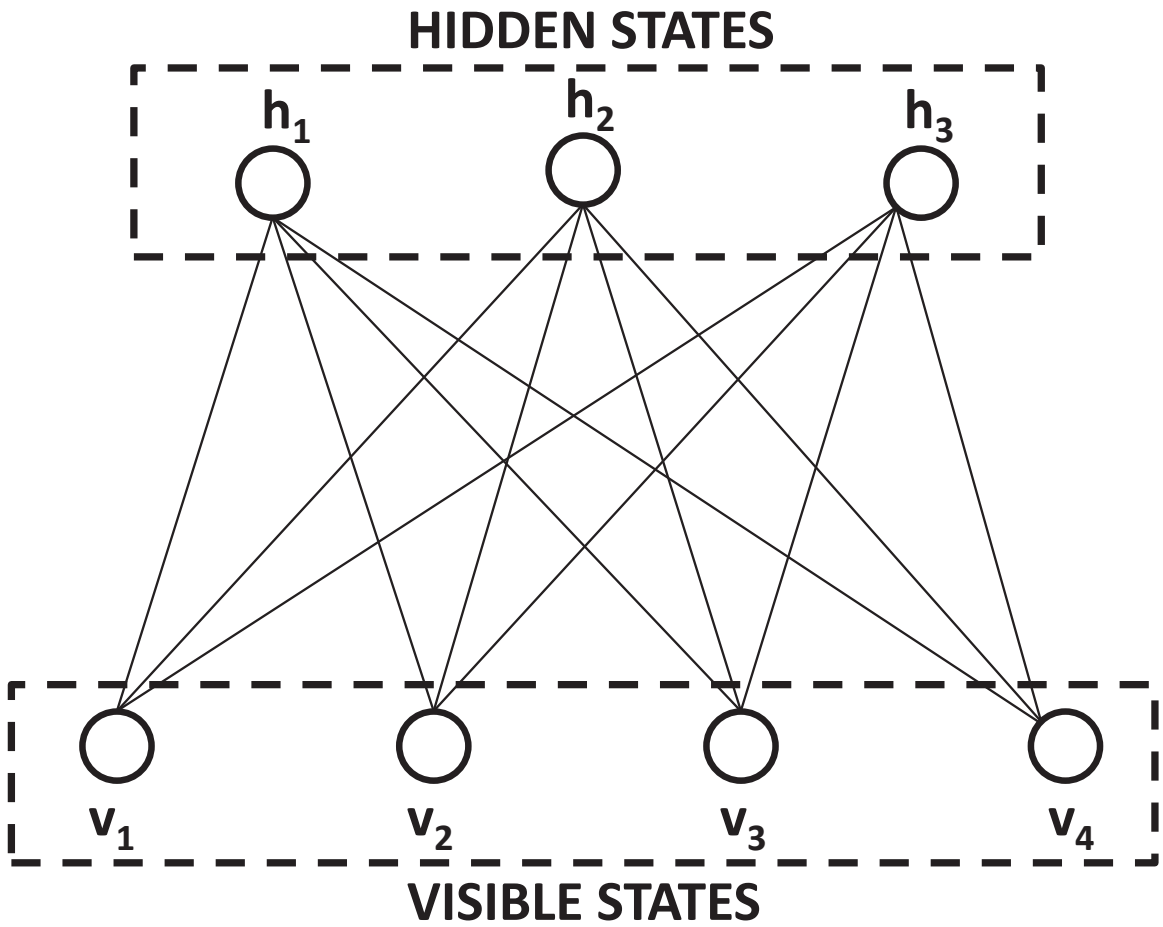
- No input to output mapping
- States are *discrete samples* of probability distributions with interdependencies among samples.
- Training points provide examples of *some* (visible) states.
- **Computational graph abstraction:** The parameterized edges define dependencies among states.
  - The computational graph abstraction is main commonality (can be exploited for pre-training)
  - Can approximately convert a sampling-based dependency to real-valued operation for initialization of a related (conventional) neural network

Historical Significance

- Most of the practical applications in neural networks use supervised learning.

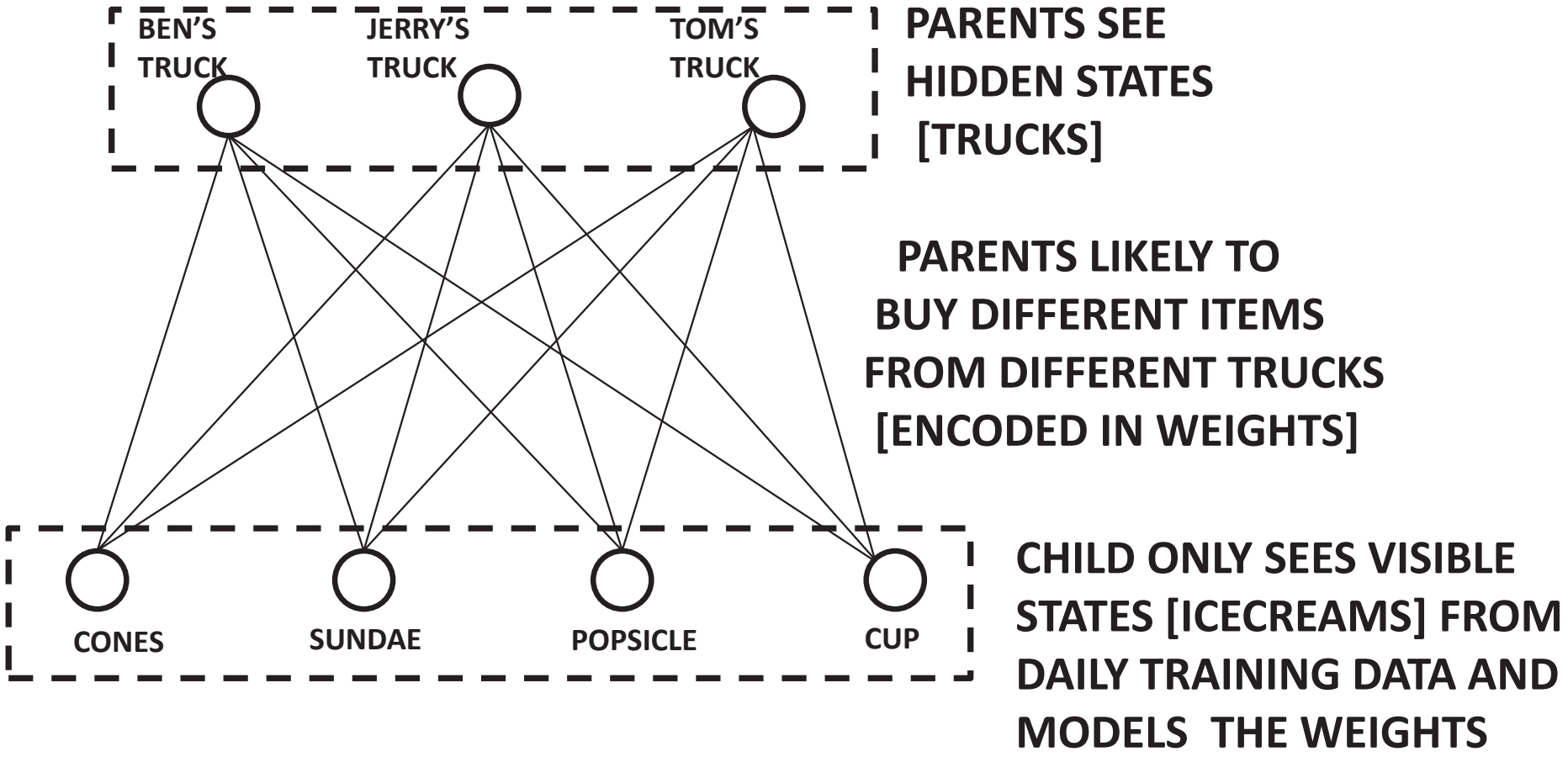
- RBMs can still be used for unsupervised pre-training of conventional neural networks and also extended to supervised learning.
  - Replace binary state outcomes with fractional probabilities
  - Treat fractional values as the activations of a conventional neural network
  - Pretraining owes its historical origins to RBMs

Defining a Restricted Boltzmann Machine



- *Bipartite* graph of *binary* hidden states and visible states connected by undirected edges signifying probabilistic dependencies $\Rightarrow$ Origin of word "*restricted*" from bipartite model

An Interpretable Boltzmann Machine



- Undirected model $\Rightarrow$ Probability to buy icecreams and pick trucks depend on one another (using weights)

What Kind of Model does a Restricted Boltzmann Machine Build?

- Probability distributions of the *binary* hidden and visible states depend on one another.
  - Weights on edges control probabilistic dependencies.
  - Training data assumed to be samples of visible states.
- We want to learn weights that are "consistent" with training samples.
- Use energy function to force "consistency" $\Rightarrow$ Unsupervised model.
- The model can use learned weights to output samples that are consistent with training data $\Rightarrow$ Generative model.

Notations

- We assume that the *binary* hidden units are $h_1 \dots h_m$ and the visible units are $v_1 \dots v_d$.
- The bias associated with the visible node v_i be denoted by $b_i^{(v)}$.
- The bias associated with hidden node h_j is denoted by $b_j^{(h)}$.
- The weight of the edge between visible node v_i and hidden node h_j is denoted by w_{ij}.
- Can be generalized to non-binary data with some work.

Probabilistic Relationships

- Want to learn weights w_{ij} so that samples of the training data are most "consistent" with the following relationships:

$$P(h_j = 1|\overline{v}) = \frac{1}{1 + \exp(-b_j^{(h)} - \sum_{i=1}^{d} v_i w_{ij})} \tag{1}$$

$$P(v_i = 1|\overline{h}) = \frac{1}{1 + \exp(-b_i^{(v)} - \sum_{j=1}^{m} h_j w_{ij})} \tag{2}$$

- Use energy function to force consistency by minimizing expected value of $E = -\sum_i b_i^{(v)} v_i - \sum_j b_j^{(h)} h_j - \sum_{i,j:i<j} w_{ij} v_i h_j$

How Data is Generated from a Boltzmann Machine

- Data is generated by using *Gibb's sampling*.
- Randomly initialize visible states and then sample hidden states using Equation 1 (previous slide).
- Alternately sample hidden states and visible states using Equations 1 and 2 until thermal equilibrium is reached.
- A particular set of visible states at thermal equilibrium provides a sample of a binary training vector.
- The weights implicitly encode the distribution by defining probabilistic dependencies.

Intuition for Weights

- Consider weights like affinities $\Rightarrow$ Large positive values of w_{ij} imply that states will be "on " together.

- We already have samples showing which visible states are "on" together.

- Weights will be learned in such a way that hidden states will be connected to correlated visible states with large weights.

  - **Biological motivation:** In Hebbian learning, a synapse between two neurons is strengthened when the neurons on either side of the synapse have highly correlated outputs.

  - Contrastive divergence algorithm learns weights.

Overview of Contrastive Divergence

- **Positive phase:** Draw b instances of hidden states based on visible states fixed to each of a mini-batch of b training points $\Rightarrow$ Yields $\langle v_i, h_j \rangle_{pos}$

- **Negative phase:** For each of the b instances in positive phase *continue to* alternately sample visible states and hidden states from one another for r iterations $\Rightarrow$ Yields $\langle v_i, h_j \rangle_{neg}$

$$
\begin{aligned}
w_{ij} &\Leftarrow w_{ij} + \alpha \left(\langle v_i, h_j \rangle_{pos} - \langle v_i, h_j \rangle_{neg} \right) \\
b_i^{(v)} &\Leftarrow b_i^{(v)} + \alpha \left(\langle v_i, 1 \rangle_{pos} - \langle v_i, 1 \rangle_{neg} \right) \\
b_j^{(h)} &\Leftarrow b_j^{(h)} + \alpha \left(\langle 1, h_j \rangle_{pos} - \langle 1, h_j \rangle_{neg} \right)
\end{aligned}
$$

Remarks on Contrastive Divergence

- Strictly speaking, the negative phase needs a very large number of iterations to reach *thermal equilibrium* in negative phase.

- Positive phase requires only one iteration because visible states are fixed to training points.

- Contrastive divergence says that only a small number of iterations of negative phase are sufficient for "good" update of weight vector (even without thermal equilibrium).

- In the early phases of training, one iteration is enough for "good" update.

- Can increase number of iterations in later phases.

Utility of Unsupervised Learning

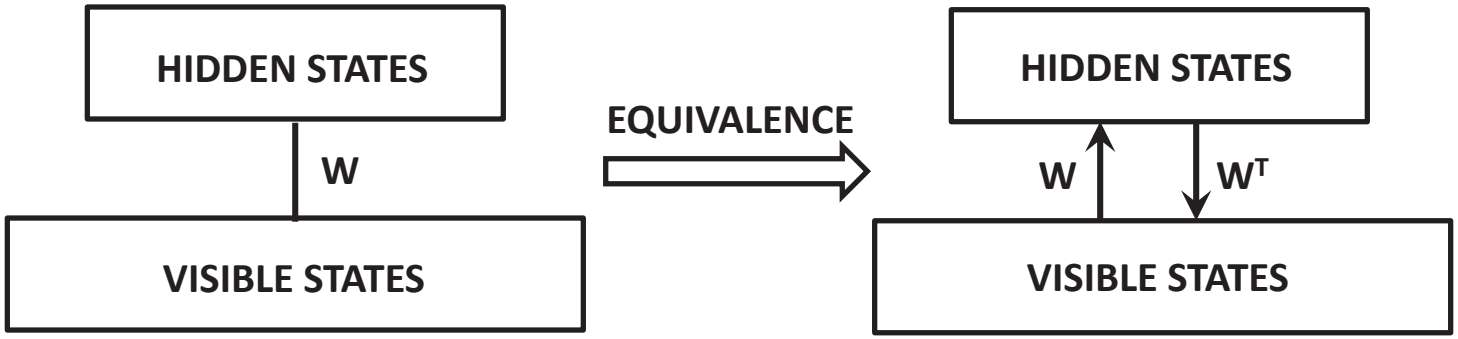
- One can use an RBM to initialize an autoencoder for binary data (later slides).

- Treat the sigmoid-based sampling as an a sigmoid activation.

- Basic idea can be extended to multilayer neural networks by using *stacked RBMs*.

  – One of the earliest methods for pretraining.

Equivalence of Directed and Undirected Models

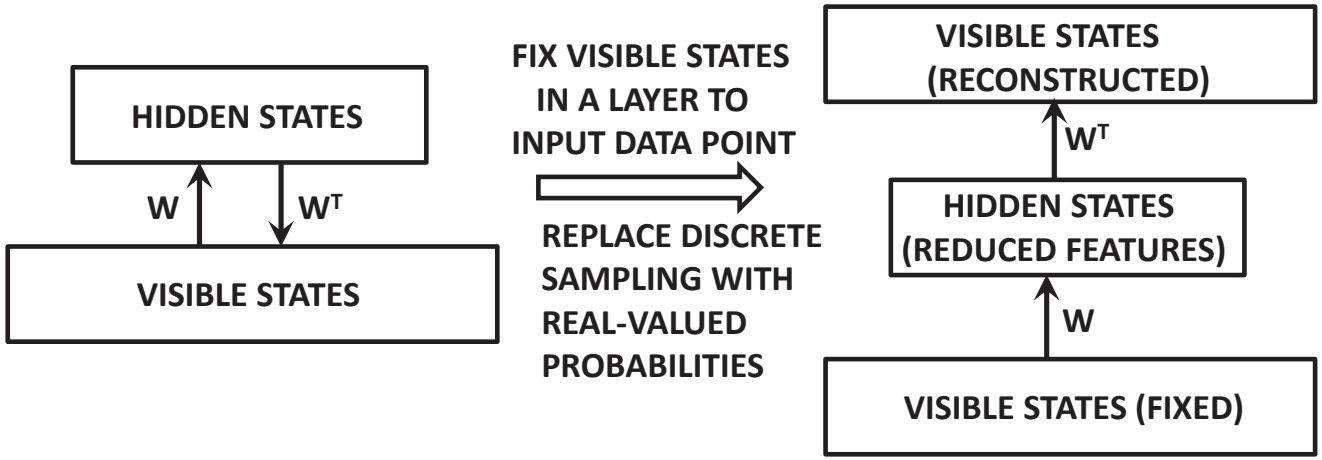


- Replace undirected edges with directed edges

$$\overline{h} \sim \text{Sigmoid}(\overline{v}, \overline{b}^{(h)}, W)$$
$$\overline{v} \sim \text{Sigmoid}(\overline{h}, \overline{b}^{(v)}, W^T)$$

- Replace sampling with real-valued operations

Using a Trained RBM to Initialize a Conventional Autoencoder



- Architecture on right uses *real-valued* sigmoid operations rather than discrete operations $\Rightarrow$ Conventional autoencoder!.

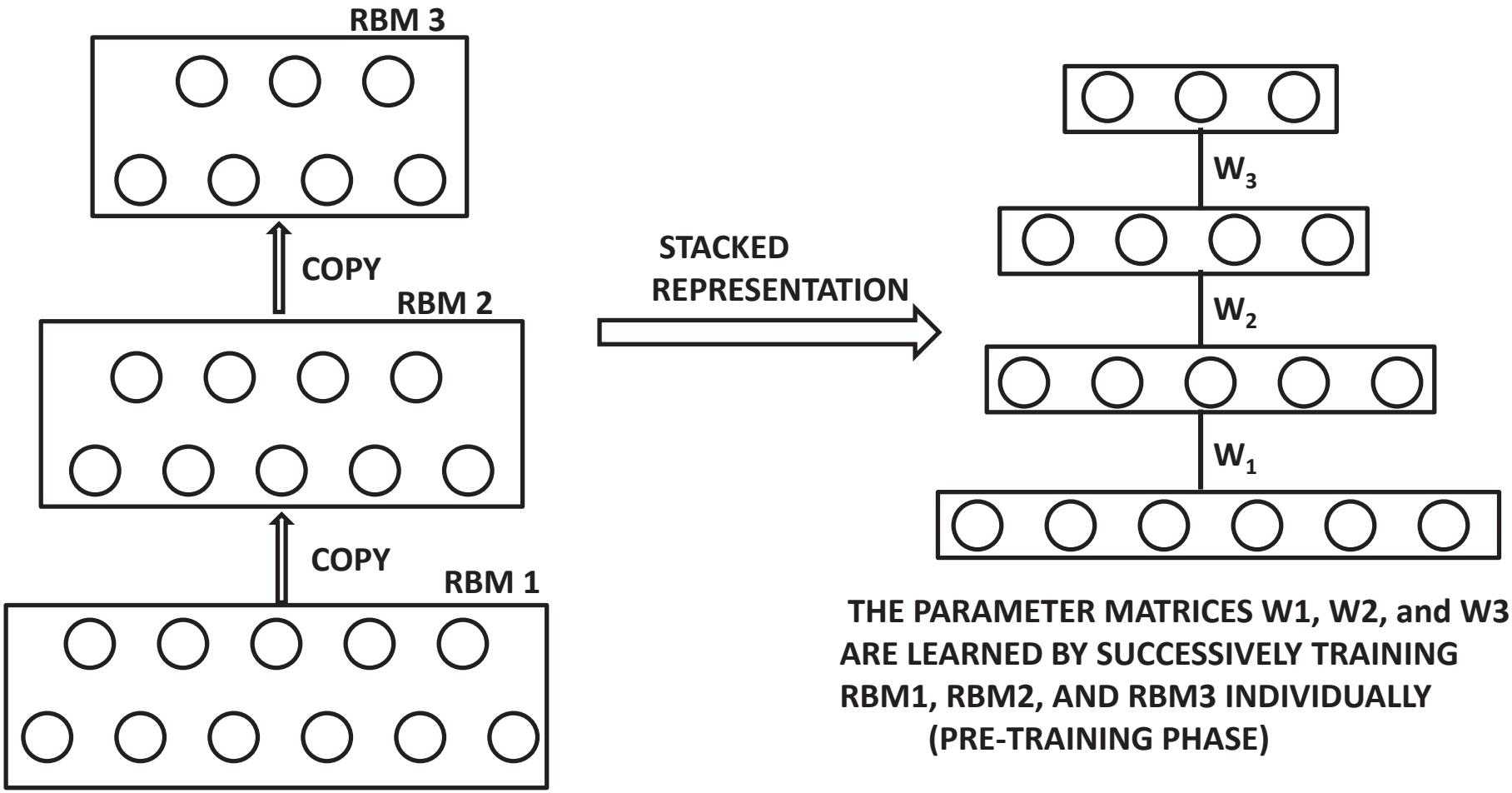
$$\widehat{h}_j = \frac{1}{1 + \exp(-b_j^{(h)} - \sum_{i=1}^{d} v_i w_{ij})} \tag{3}$$

$$\widehat{v}_i = \frac{1}{1 + \exp(-b_i^{(v)} - \sum_{j=1}^{m} \widehat{h}_j w_{ij})} \tag{4}$$

Why Use an RBM to Initialize a Conventional Neural Network?

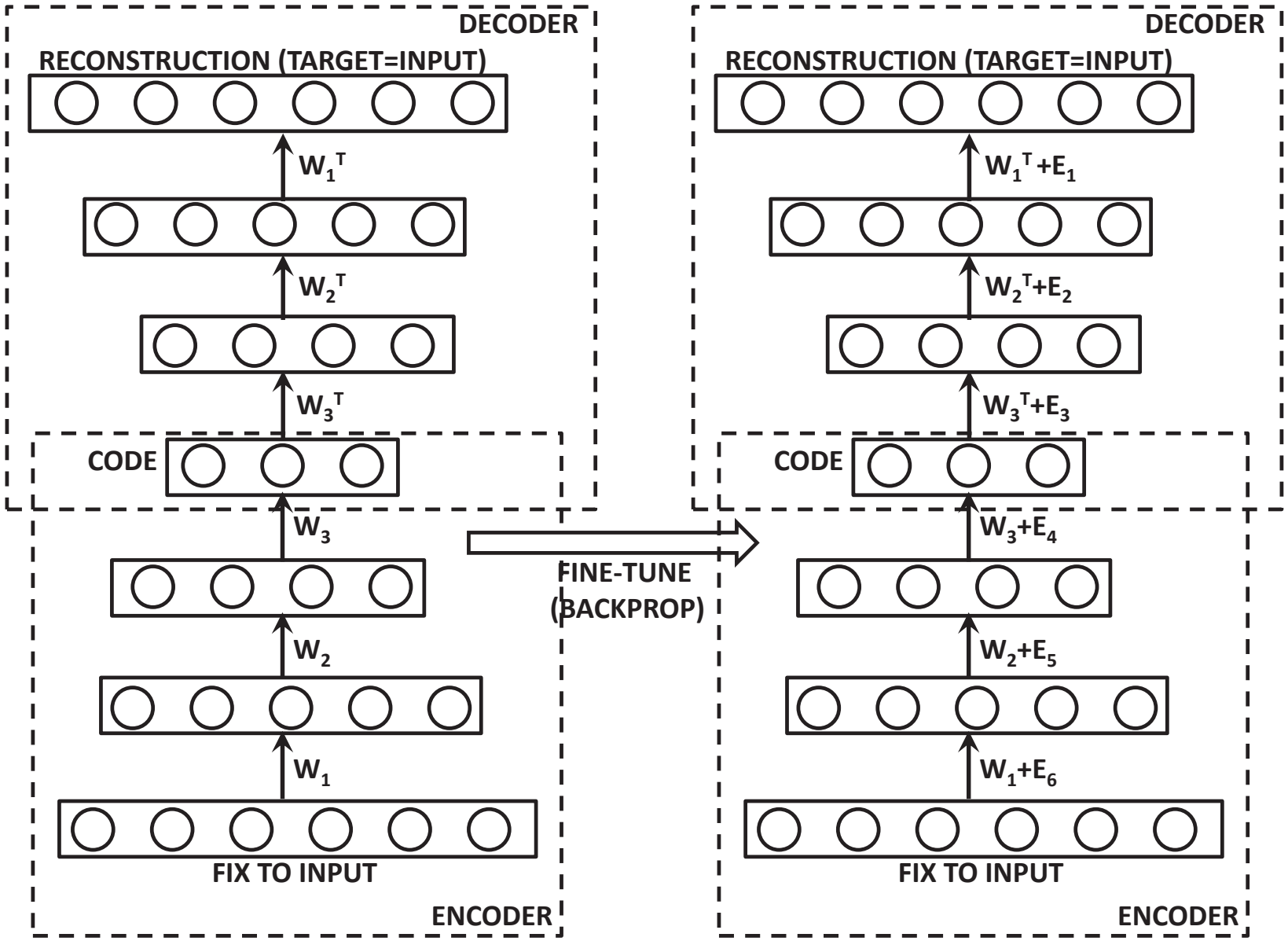
- In the early years, conventional neural networks did not train well (especially with increased depth).
  - Vanishing and exploding gradient problems
  - RBM trains with contrastive divergence (no vanishing and exploding gradient)
- Real-valued approximation was used with stacked RBMs to initialize deep networks
- Approach was generalized to conventional autoencoders later

Stacked RBM



- Train different layers sequentially

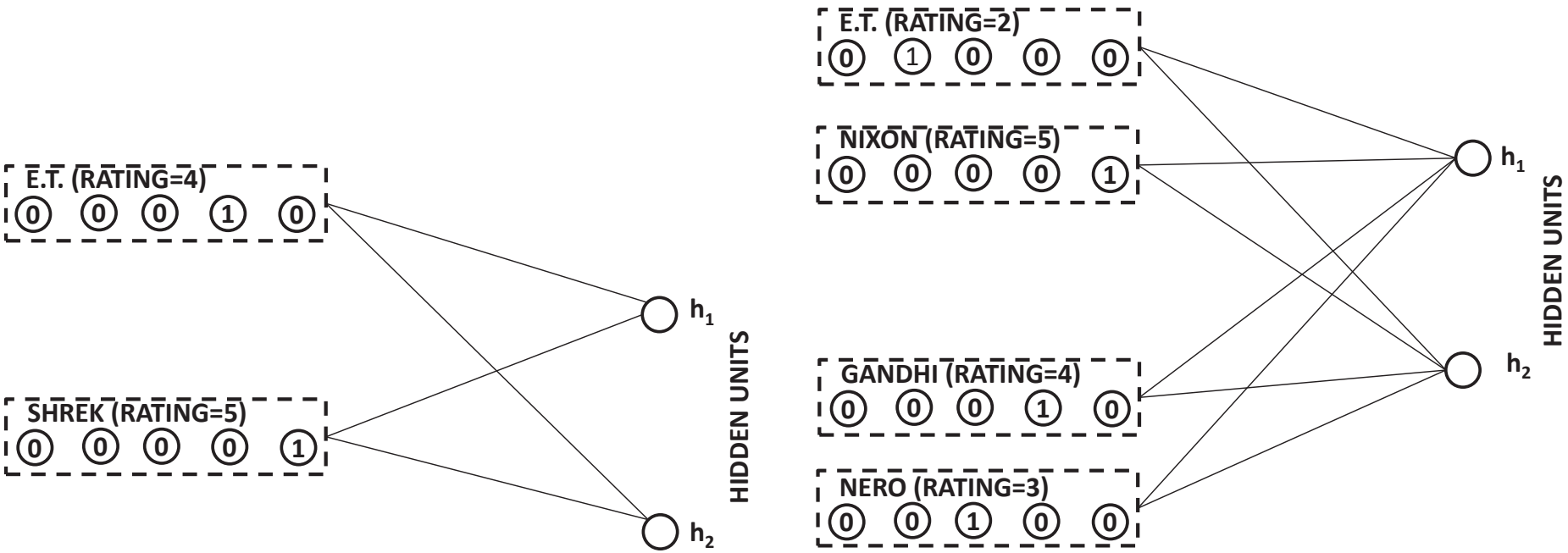
Stacked RBM to Conventional Neural Network



Applications

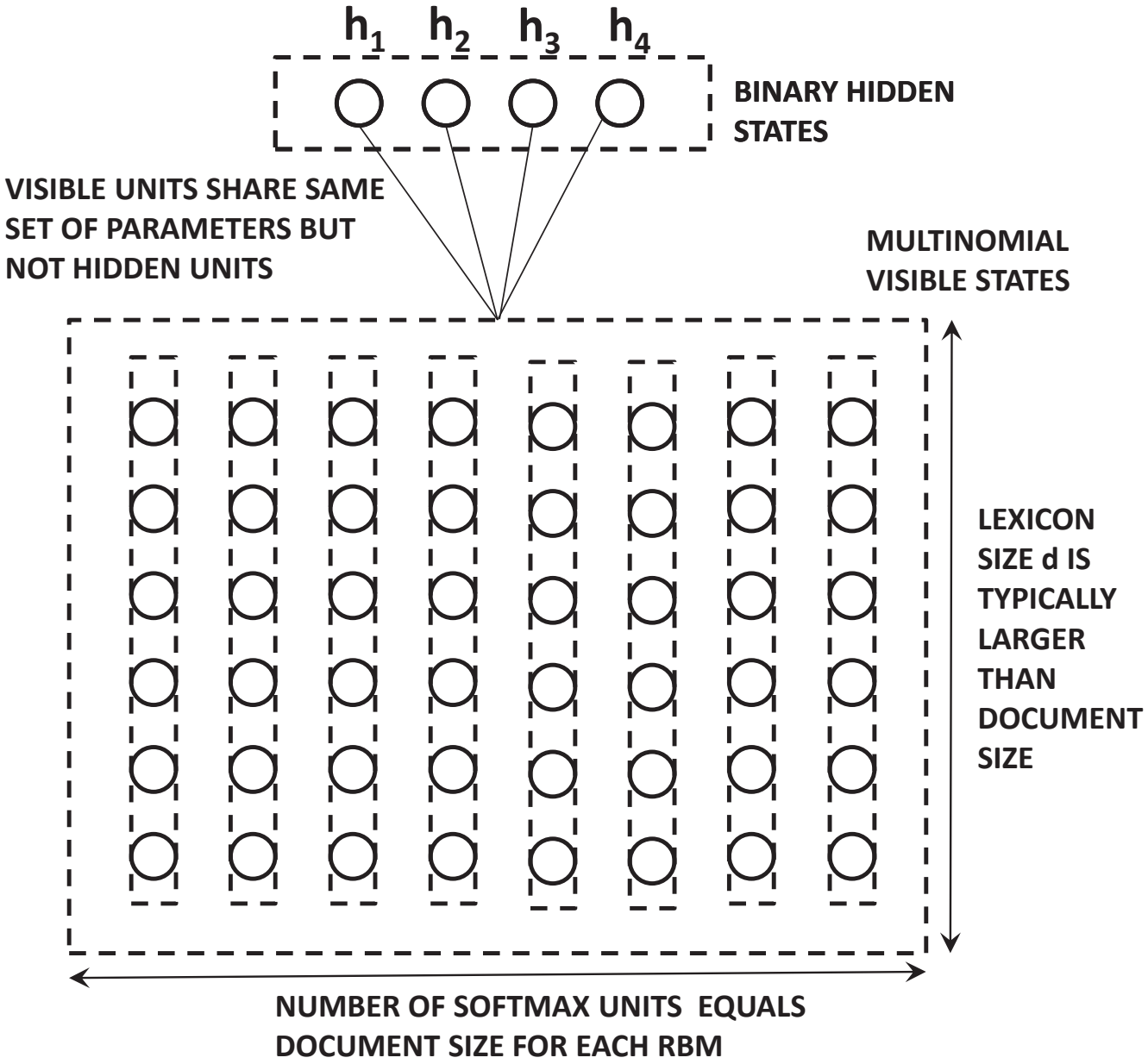
- Pretraining can be used for supervised and unsupervised applications
  - Collaborative filtering: Was a component of Netflix prize contest
    - Gives different results from an autoencoder-like architecture in an earlier lecture
  - Topic models
  - Classification

Collaborative Filtering



- Changes for softmax activations and shared weights across RBMs

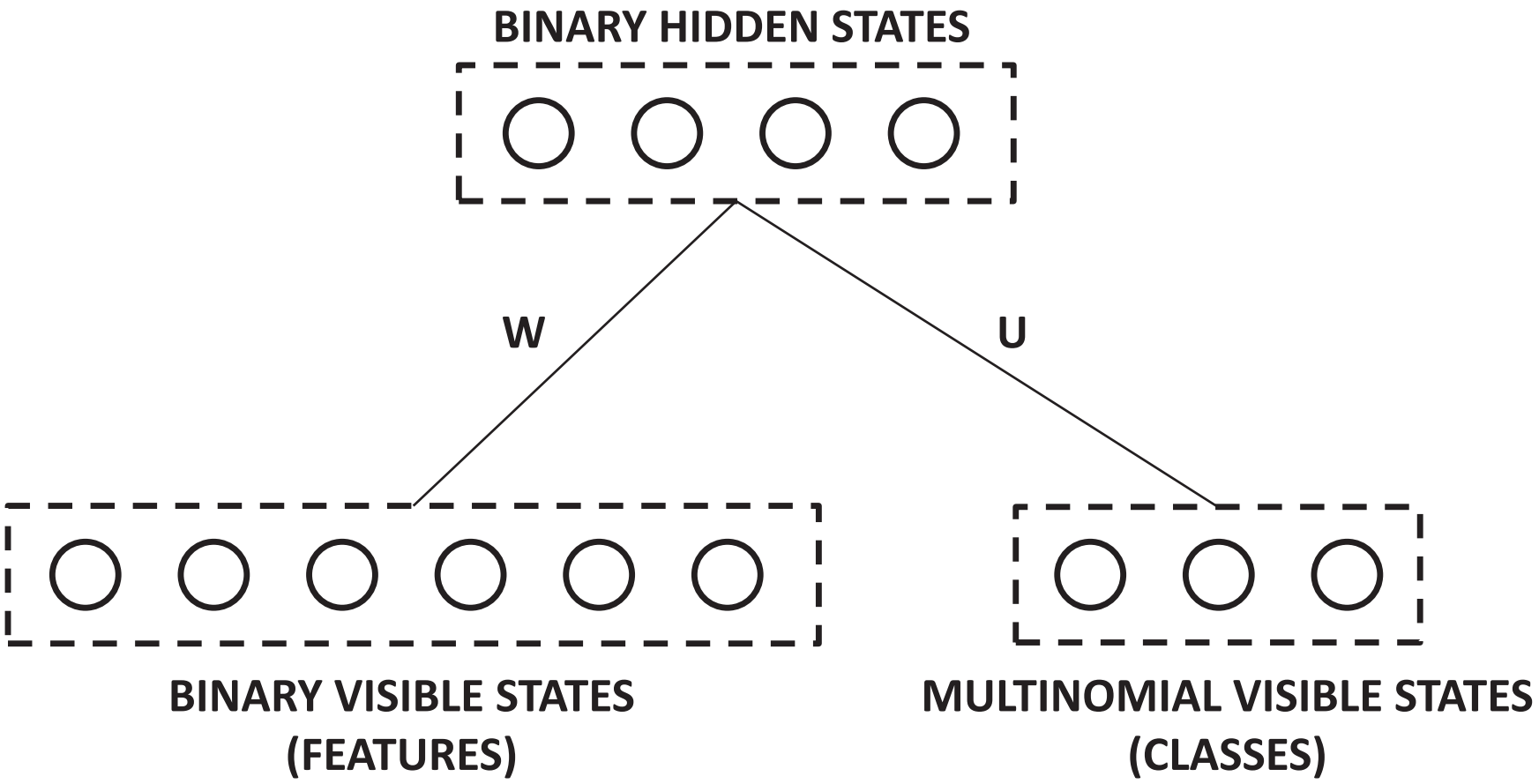
Topic Models



Classification

- Can be used for unsupervised pretraining for classification
  - Goal of RBM is only to learn features in unsupervised way
  - Class label does not get a state in RBM
- Can also be used to train by treating a class label as a state.
  - Hidden features are connected to both class variables and feature variables.
  - *Generative* approach of RBMs does not fully optimize for classification accuracy $\Rightarrow$ Need *discriminative* Boltzmann Machines (Larochelle *et al*).

Classification Architecture



Comments

- RBMs represent a special case of probabilistic graphical models.
- Provides an alternative to the autoencoder.
- Can be extended to non-binary data.
- These models are not quite as popular anymore.
- Significant historical significance in starting the idea of pre-training for deep models.