E9: 309 ADL 18-11-2020

http://leap.ee.iisc.ac.in/sriram/teaching/ADL2020/

Housekeeping

- 1st mini-project
  - Deadlines
    - Presentation on Nov19 and Nov20
      - Your date allocation has been finalized
      - Presentation and report template will be sent out this week.
        - Report 1 page + references and tools
        - Slides 4 slides for individual project and 6 slides for 2-member.

Recap of previous class

Representation learning/data-visualization

- ✳ Restricted Boltzmann machine
  - ➡ Energy based model
    - ✓ Conditional independence
    - ✓ Sigmoidal function for conditional probability
  - ➡ Issues in RBM training

RBM - Training

- Model parameters $\Theta = \{\mathbf{W}, \mathbf{b}, \mathbf{c}\}$ $\mathbf{x} = [\mathbf{v}^T \ \mathbf{h}^T]^T$
  - ✓ Learnt by maximizing the log-likelihood $log(p(\mathbf{x}; \Theta))$
  - ✓ Non-convex optimization.
- Gradient descent based optimization

$$\frac{\partial log(p(\mathbf{x}; \Theta))}{\partial \Theta} = \frac{\partial log(\tilde{p}(\mathbf{x}; \Theta))}{\partial \Theta} - \frac{\partial log(Z(\Theta))}{\partial \Theta}$$

$$\frac{\partial log(Z(\Theta))}{\partial \Theta} = \frac{1}{Z} \frac{\partial Z(\Theta)}{\partial \Theta}$$

$$Z(\Theta) = \sum_{\mathbf{x}} \tilde{p}(\mathbf{x}; \Theta))$$

RBM - Training

* For exponential families

$$\tilde{p}(\mathbf{x};\Theta) > 0 \quad \forall\ \mathbf{x}$$

$$\frac{\partial log(Z(\Theta))}{\partial \Theta} = \frac{1}{Z}\sum_{\mathbf{x}} \frac{\partial \tilde{p}(\mathbf{x};\Theta)}{\partial \Theta}$$

$$\frac{\partial log(Z(\Theta))}{\partial \Theta} = \frac{1}{Z}\sum_{\mathbf{x}} \frac{\partial exp(log(\tilde{p}(\mathbf{x};\Theta)))}{\partial \Theta}$$

$$= \frac{1}{Z}\sum_{\mathbf{x}} exp(log(\tilde{p}(\mathbf{x};\Theta)))\frac{\partial (log(\tilde{p}(\mathbf{x};\Theta)))}{\partial \Theta}$$

$$= \sum_{\mathbf{x}} p(\mathbf{x};\Theta)\frac{\partial (log(\tilde{p}(\mathbf{x};\Theta)))}{\partial \Theta}$$

RBM - Training

$$\frac{\partial(log Z(\Theta))}{\partial \Theta} = \mathbb{E}_{\mathbf{x}\sim p}\left[\frac{\partial \tilde{p}(\mathbf{x}, \Theta)}{\partial \Theta}\right]$$

- ✳ Intractable to compute the exact gradient of the negative phase
  - ➡ Using approximations to gradients
    - ✓ Based on sampling methods.
      - ⋆ Monte-carlo Markov Chain (MCMC) based approximation
        - ⋆ Resorting to Gibbs sampling.

Approximating expectations

* Expectation is intractable in the gradient computation.

$$\mathbf{s} = \mathbb{E}_{x \sim p}[f(\mathbf{x})]$$

$$\hat{\mathbf{s}}_n = \frac{1}{n} \sum_{i=1}^{n} f(\mathbf{x}_i)$$

* Using law of large numbers and central limit theorem

$$\mathbb{E}[\hat{\mathbf{s}}_n] = \mathbf{s} \ ; \quad \lim_{n \to \infty} \hat{\mathbf{s}}_n = \mathbf{s} \ ; \ Var[\hat{\mathbf{s}}_n] = \frac{Var[f(\mathbf{x})]}{n}$$

Sampling

* The question of finding the right samples $\mathbf{x}_i$

* Should we sample based on the p or some other function q

$$p(\mathbf{x})f(\mathbf{x}) = q(\mathbf{x})\frac{p(\mathbf{x})f(\mathbf{x})}{q(\mathbf{x})}$$

* Using a suitable function q, the estimate of the intractable expectation is

$$\hat{\mathbf{s}}_q = \frac{1}{n}\sum_{i=1,\mathbf{x}_i\sim q}^{n}\frac{p(\mathbf{x}_i)f(\mathbf{x}_i)}{q(\mathbf{x}_i)}$$

Importance Sampling

Gibbs sampling with random initialization

- ✳ Using Markov chain Monte-Carlo (MCMC) sampling $p_{model}(\mathbf{x})$
  - ✓ Initialize random samples using uniform distribution.
  - ✓ Use the data samples to perform new updates of the samples
  - ✓ Perform k steps of this update.

Gibbs sampling with random initialization

Algorithm 18.1 A naive MCMC algorithm for maximizing the log-likelihood with an intractable partition function using gradient ascent.

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow burn in. Perhaps 100 to train an RBM on a small image patch.
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from the training set.
  $\mathbf{g} \leftarrow \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)}; \boldsymbol{\theta})$.
  Initialize a set of m samples $\{\tilde{\mathbf{x}}^{(1)}, \ldots, \tilde{\mathbf{x}}^{(m)}\}$ to random values (e.g., from a uniform or normal distribution, or possibly a distribution with marginals matched to the model's marginals).
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow \text{gibbs\_update}(\tilde{\mathbf{x}}^{(j)})$.
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m} \sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)}; \boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon \mathbf{g}$.
end while

Gibbs sampling with random initialization

- Using Markov chain Monte-Carlo (MCMC) sampling $p_{model}(\mathbf{x})$
  - ✓ Use the given data samples and the current model weights to perform Gibbs sampling
    - ★ Initially the model weights make for a poor estimation of the negative phase of the gradient.
      - ★ But the positive phase makes up for the lossy estimate.
    - ★ Once the model weights are updated for a few iterations
      - ★ The negative phase becomes more accurate.

Contrastive Divergence

Gibbs sampling with random initialization

Algorithm 18.2 The contrastive divergence algorithm, using gradient ascent as the optimization procedure.

Set ϵ, the step size, to a small positive number.
Set k, the number of Gibbs steps, high enough to allow a Markov chain sampling from $p(\mathbf{x};\boldsymbol{\theta})$ to mix when initialized from p_{data}. Perhaps 1-20 to train an RBM on a small image patch.
while not converged **do**
  Sample a minibatch of m examples $\{\mathbf{x}^{(1)}, \ldots, \mathbf{x}^{(m)}\}$ from the training set.
  $\mathbf{g} \leftarrow \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\mathbf{x}^{(i)};\boldsymbol{\theta})$.
  for $i = 1$ to m **do**
    $\tilde{\mathbf{x}}^{(i)} \leftarrow \mathbf{x}^{(i)}$.
  end for
  for $i = 1$ to k **do**
    for $j = 1$ to m **do**
      $\tilde{\mathbf{x}}^{(j)} \leftarrow$ gibbs_update($\tilde{\mathbf{x}}^{(j)}$).
    end for
  end for
  $\mathbf{g} \leftarrow \mathbf{g} - \frac{1}{m}\sum_{i=1}^{m} \nabla_{\boldsymbol{\theta}} \log \tilde{p}(\tilde{\mathbf{x}}^{(i)};\boldsymbol{\theta})$.
  $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \epsilon\mathbf{g}$.
end while

One step contrastive divergence - Training example

$$\mathbf{x} = [\mathbf{v}^T \ \mathbf{h}^T]^T$$

- Given a set of visible data $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N\} \in \mathcal{B}^D$
- Randomly initialize the model parameters $\Theta^0 = \{\mathbf{W}^0, \mathbf{b}^0, \mathbf{c}^0\}$ $k = 0$
  - ✓ Sampling $\{\mathbf{h}_1, \mathbf{h}_2, ..., \mathbf{h}_N\} \in \mathcal{B}^d$
    - ◉ Using conditional independence of hidden given visible
      - ⋆ Sample each $\{\tilde{\mathbf{h}}_1, \tilde{\mathbf{h}}_2, ..., \tilde{\mathbf{h}}_N\} \in \mathcal{B}^d$

$$p(h_{q,j} = 1|\mathbf{v}_q) = \sigma(\mathbf{v}_q^T \mathbf{W}_{:,j}^k + c_j) \quad q = \{1, .., N\}, j = \{1, .., d\}$$

One step contrastive divergence - Training example

$$\mathbf{x} = [\mathbf{v}^T \; \mathbf{h}^T]^T$$

- ➡ Given a set of visible data $\{\mathbf{v}_1, \mathbf{v}_2, ..., \mathbf{v}_N\} \in \mathcal{B}^D$
- ➡ Randomly initialize the model parameters $\Theta^0 = \{\mathbf{W}^0, \mathbf{b}^0, \mathbf{c}^0\}$ $k = 0$
  - ✓ Sampling visible nodes again
    - ◉ Using conditional independence of visible given hidden
      - ★ Sample visible nodes $\{\tilde{\mathbf{v}}_1, \tilde{\mathbf{v}}_2, ..., \tilde{\mathbf{v}}_N\} \in \mathcal{B}^D$

$$p(v_{q,i} = 1|\tilde{\mathbf{h}}_q) = \sigma(\mathbf{W}^k_{i,:}\tilde{\mathbf{h}}^T_q + b^k_i) \quad q = \{1, .., N\}, i = \{1, .., D\}$$

One-step contrastive divergence

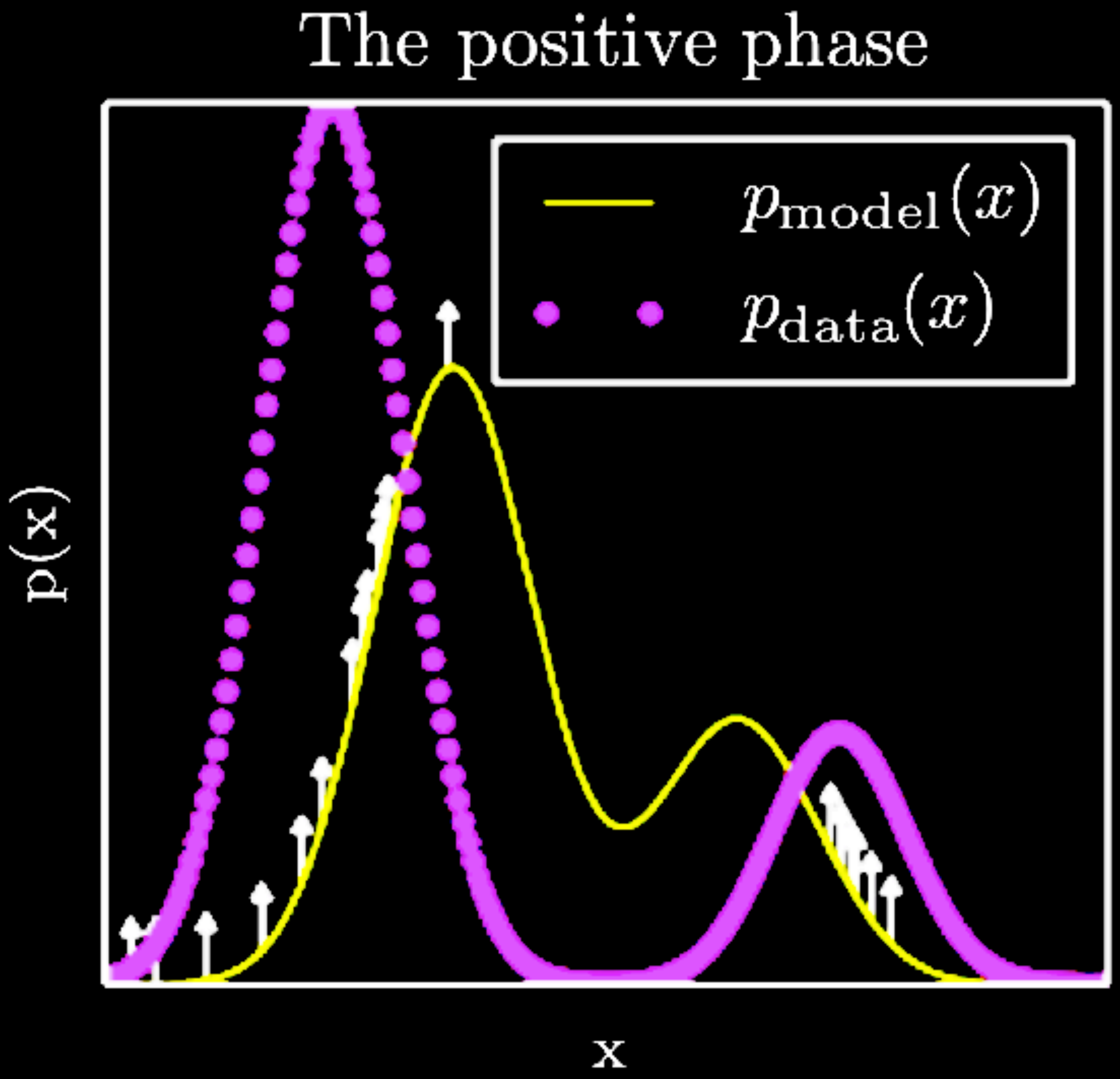
- Computing the gradient

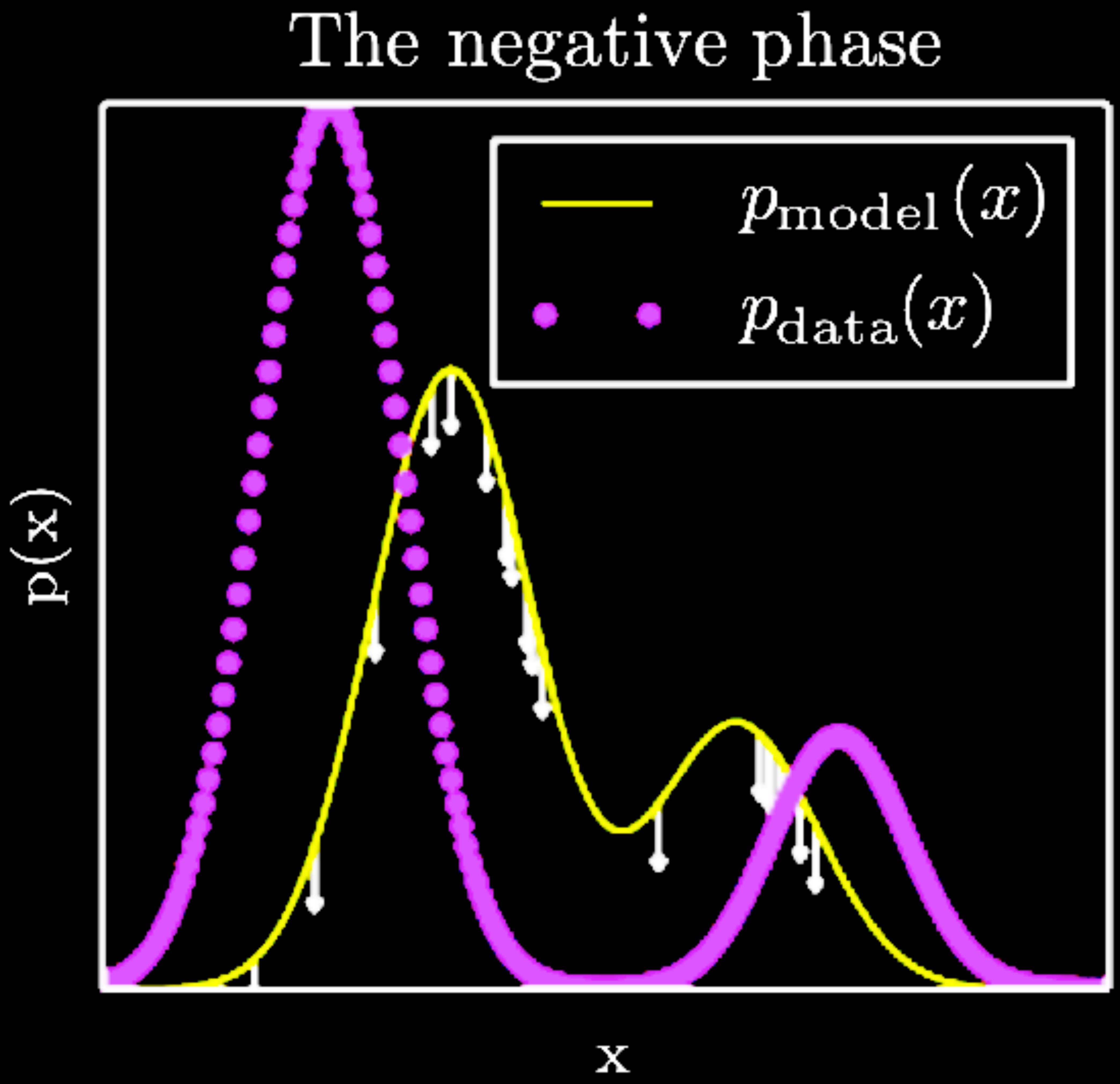
$$\frac{\partial(p([\mathbf{v}\ \mathbf{h}], \Theta)}{\partial \mathbf{W}} \approx \frac{1}{N}\sum_{q=1}^{N} \mathbf{v}_q \mathbf{h}_q^T - \frac{1}{N}\sum_{q=1}^{N} \tilde{\mathbf{v}}_q \tilde{\mathbf{h}}_q^T$$

- Performing gradient ascent using the approximate gradient

$$\boldsymbol{\Theta}^{k+1} = \boldsymbol{\Theta}^{k} + \eta \frac{\partial log(p(\mathbf{x}, \Theta))}{\partial \boldsymbol{\Theta}}\Bigg|_{\boldsymbol{\Theta}=\boldsymbol{\Theta}^k}$$

Positive phase and negative phase





"Motivation"

- Neuroscientific motivation of the learning in the brain
  - ➡ Real world samples in our day-to-day [Positive phase]
  - ➡ Hallucinations and dreams [Negative phase]

- Learning from the data.
  - ➡ Balancing the positive phase based learning with the negative phase.

Gaussian Bernoulli RBM

- For modeling real observations $\mathbf{v} \in \mathcal{R}^D$
- Define the energy function

$$E[\mathbf{v}, \mathbf{h}] = \frac{1}{2}(\mathbf{v} - \mathbf{a})^T(\mathbf{v} - \mathbf{a}) - \mathbf{v}^T\mathbf{W}\mathbf{h} - \mathbf{b}^T\mathbf{h}$$

$$p([\mathbf{v}, \mathbf{h}]) = \frac{e^{-E[\mathbf{v}, \mathbf{h}]}}{Z}$$

- The conditional distributions

$$p(\mathbf{v}|\mathbf{h}) = \mathcal{N}(\mathbf{W}\mathbf{h} + \mathbf{a}, \mathbf{I})$$

$$p(h_j = 1|\mathbf{v}) = \sigma(\mathbf{v}^T\mathbf{W}^k_{:,j} + b_j)$$

Properties of GRBM

- d = 0

$$E(\mathbf{v}) = \frac{1}{2}(\mathbf{v} - \mathbf{a})^T(\mathbf{v} - \mathbf{a})$$

$$p(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{Z}$$

- The marginal distribution is a Gaussian.

Properties of GRBM

* d = 1

$$E[\mathbf{v}, h] = \frac{1}{2}(\mathbf{v} - \mathbf{a})^T(\mathbf{v} - \mathbf{a}) - h\mathbf{v}^T\mathbf{w} - hb$$

$$p([\mathbf{v}, h]) = \frac{e^{-E[\mathbf{v}, h]}}{Z}$$

$$p([\mathbf{v}, h = 0]) = \alpha\mathcal{N}(\mathbf{a}, \mathbf{I})$$

$$p([\mathbf{v}, h = 1]) = (1 - \alpha)\mathcal{N}(\mathbf{a} + \mathbf{w}, \mathbf{I})$$

* The marginal distribution is then

$$p(\mathbf{v}) = p([\mathbf{v}, h = 0]) + p([\mathbf{v}, h = 1])$$

  ✓ 2-mixture Gaussian

Properties of GRBM

- For any general d dimensions

$$E[\mathbf{v}, \mathbf{h}_d] = E[\mathbf{v}, \mathbf{h}_{d-1}] + h_d \, \mathbf{v}^T \mathbf{W}_{:,d} + b_d h_d$$

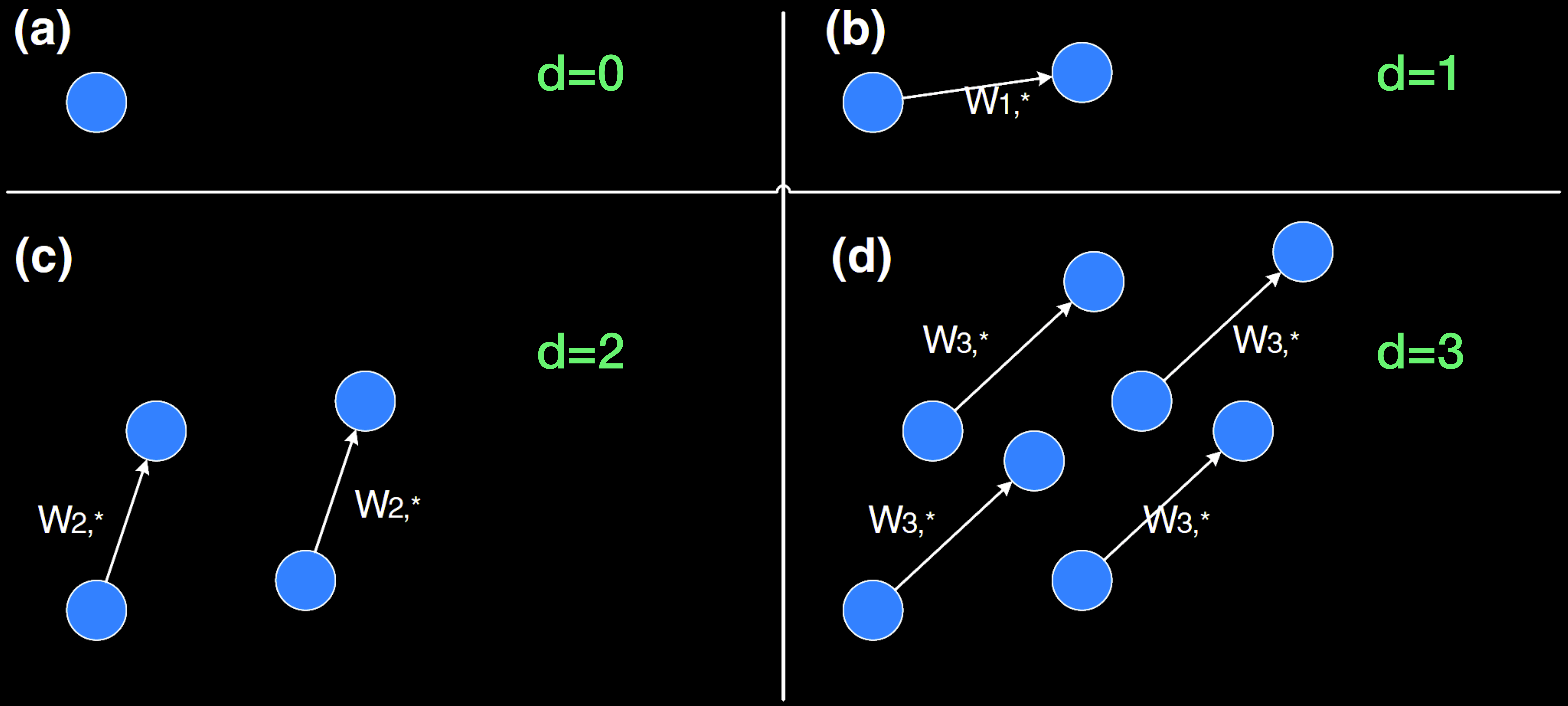
$$p([\mathbf{v}, [\mathbf{h}_{d-1}, h_d = 0]]) = \alpha \, p([\mathbf{v}, \mathbf{h}_{d-1}])$$

$$p([\mathbf{v}, [\mathbf{h}_{d-1}, h_d = 1]]) = (1 - \alpha) \, p([(\mathbf{v} + \mathbf{W}_{:,d}), \mathbf{h}_{d-1}])$$

- For d=0, 1 Gaussian, d=1, 2-mix Gaussian, …

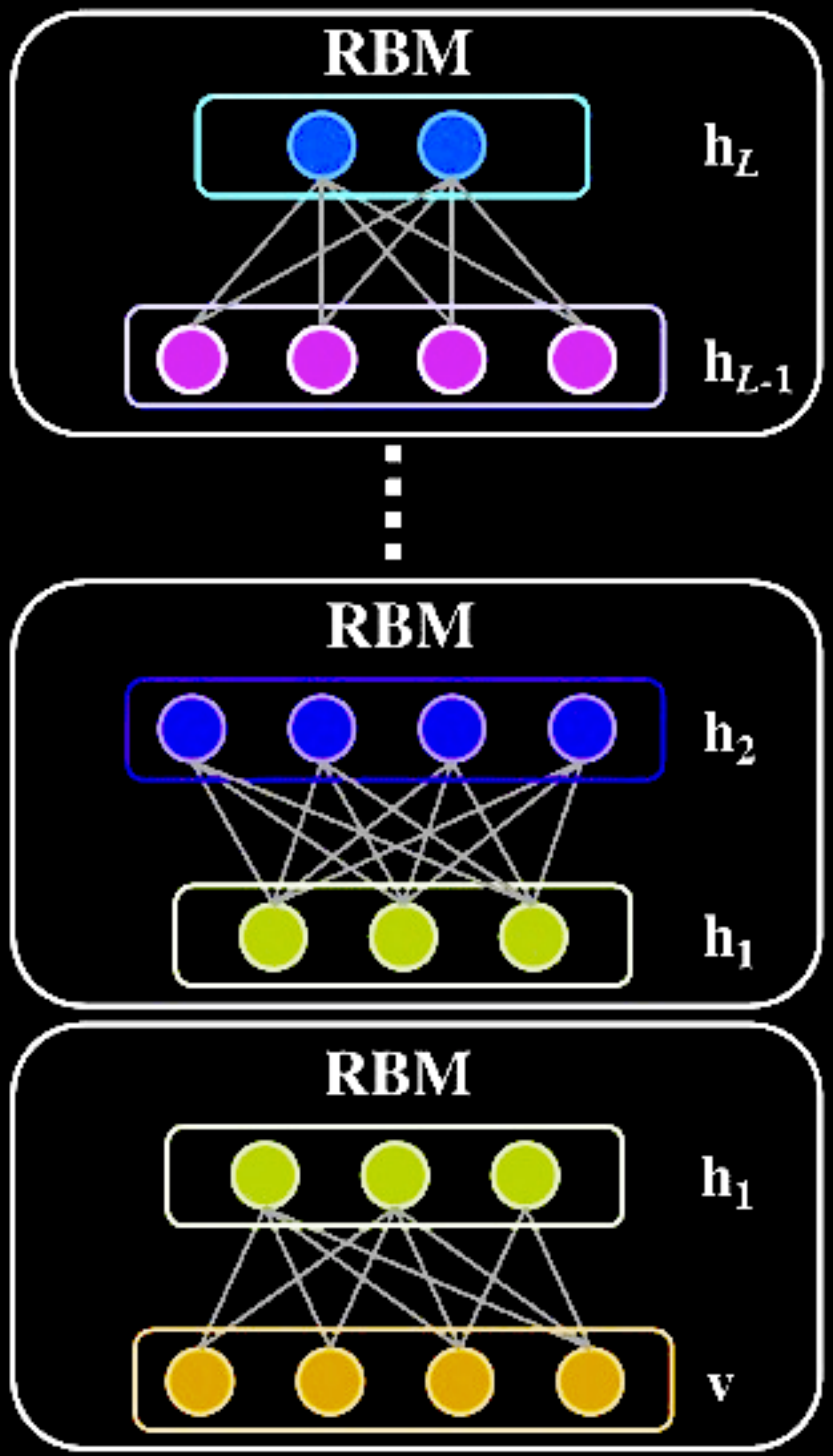
  ➡ 2^d mixture Gaussian for any arbitrary d.

GRBMs and GMMs

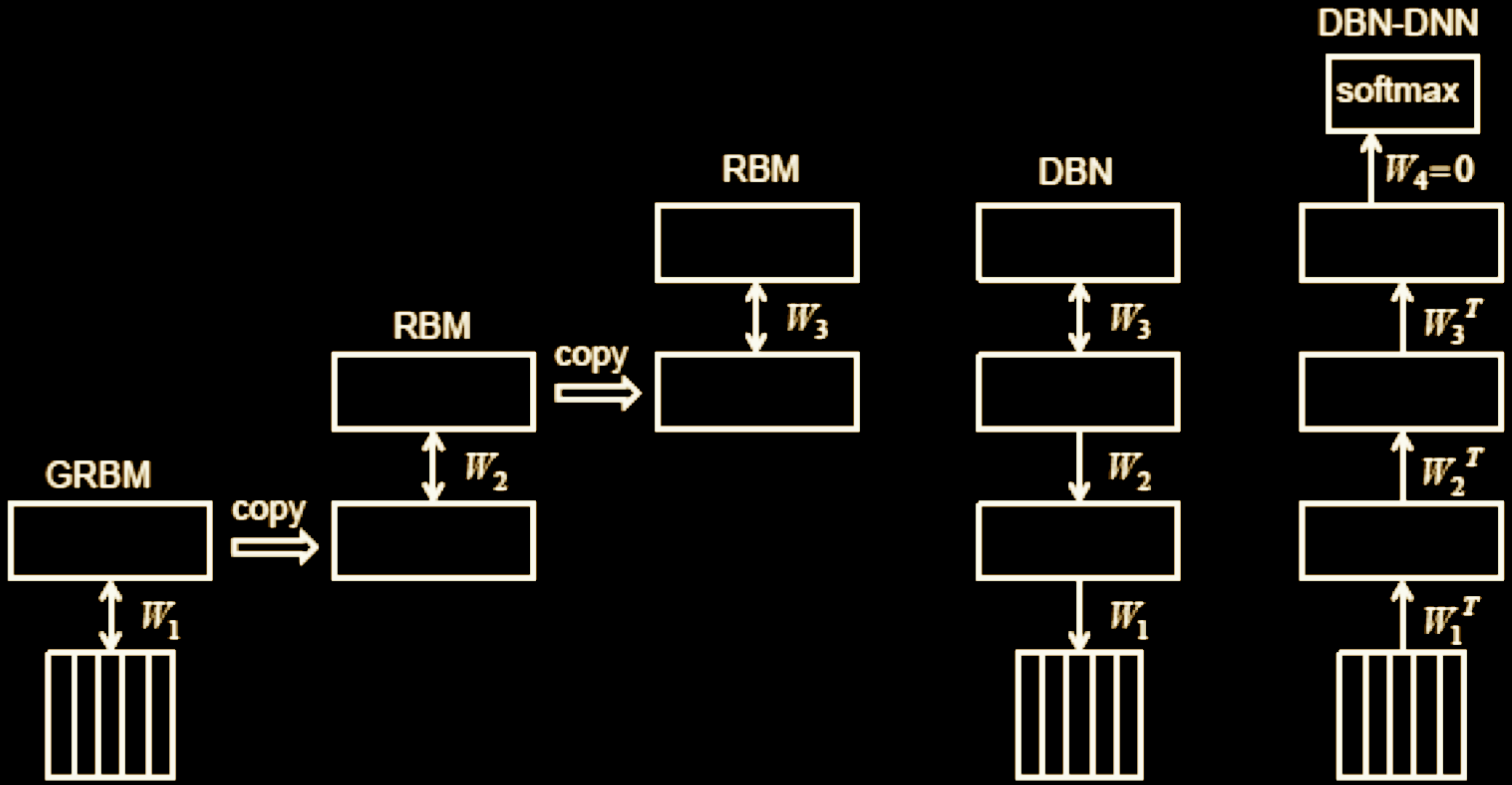


Deep Belief Networks (DBN)

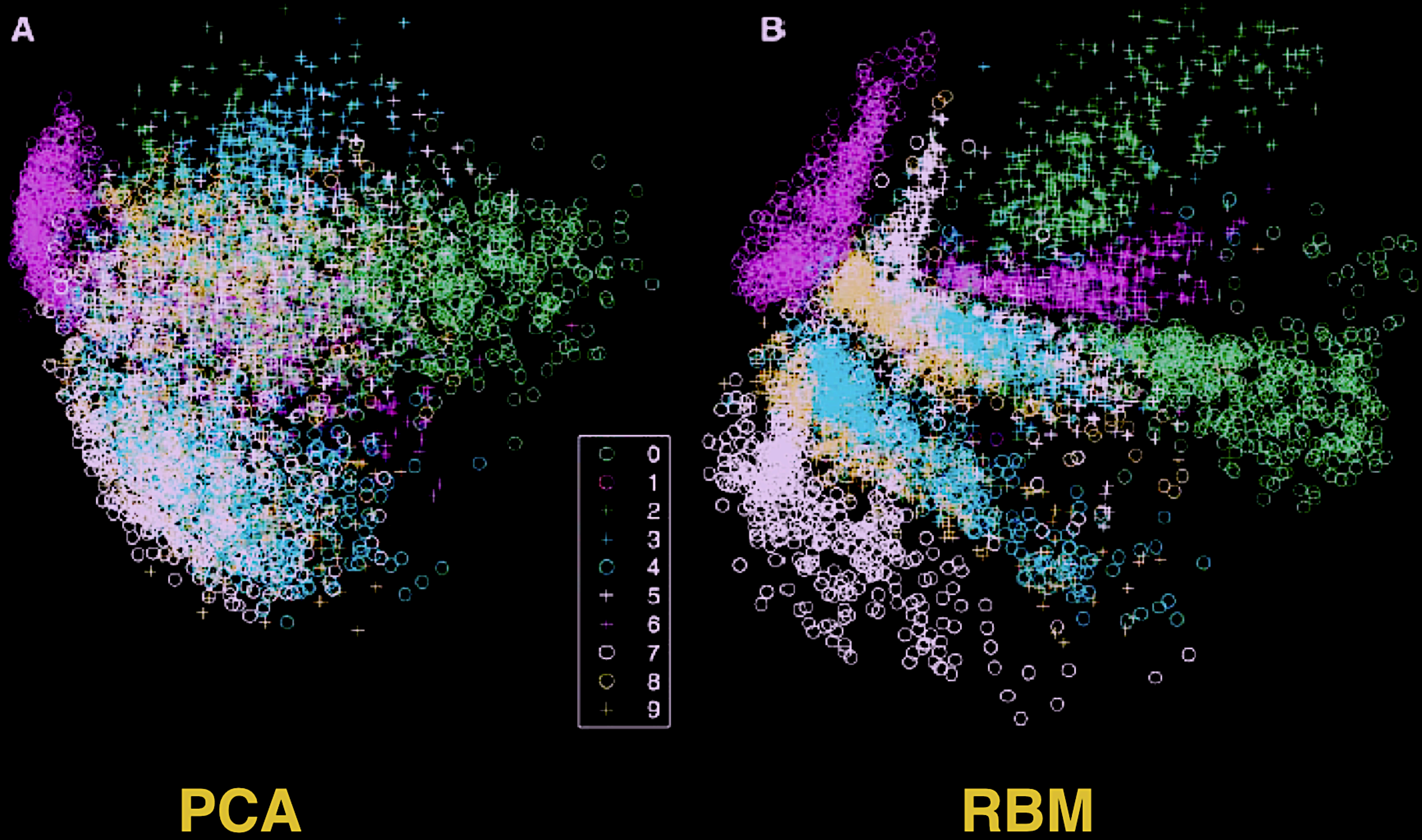
- Stacking RBMs in a disjoint fashion
  - ✓ Layer-wise training for each RBM.
    - ◉ Weights are frozen each layer before training the next layer.
  - ✓ Ancestral sampling can be performed for data generation
    - ★ Lossy sample generation due to accumulation of errors.
- Most common use - pre-training of DNNs.



DBNs for initialization



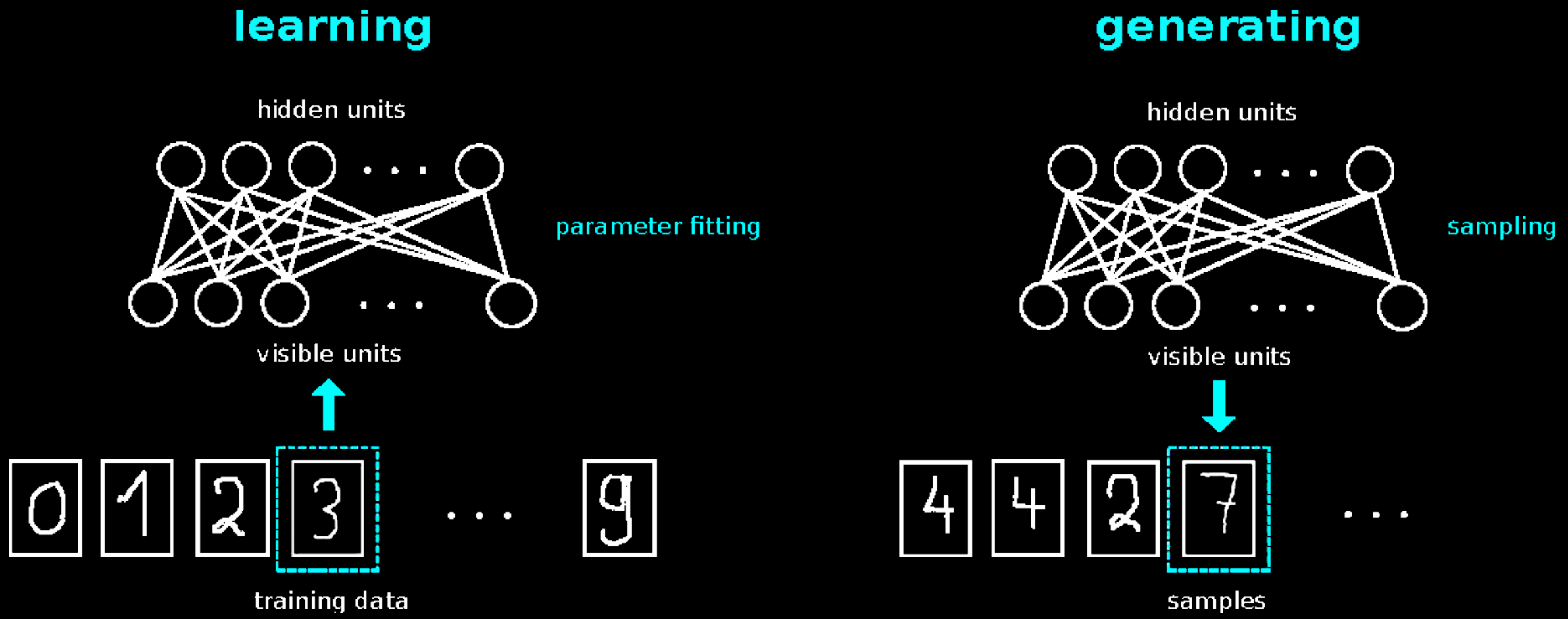
DBNs for visualization



More reading

Salakhutdinov, Ruslan, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering." *Proceedings of the 24th international conference on Machine learning*. 2007.

Data generation using RBMs



Data generation using RBMs

Source — https://lme.tf.fau.de/lecture-notes/lecture-notes-in-deep-learning-unsupervised-learning-part-1/

Deep Boltzmann machine

- Deep layers of connections with RBM structure.
  - ➡ Joint energy function.
  - ➡ Undirected graph