Restricted Boltzmann Machines (RBM's)

Viktor Kuhn, Maximilian Lutz

August 8, 2019

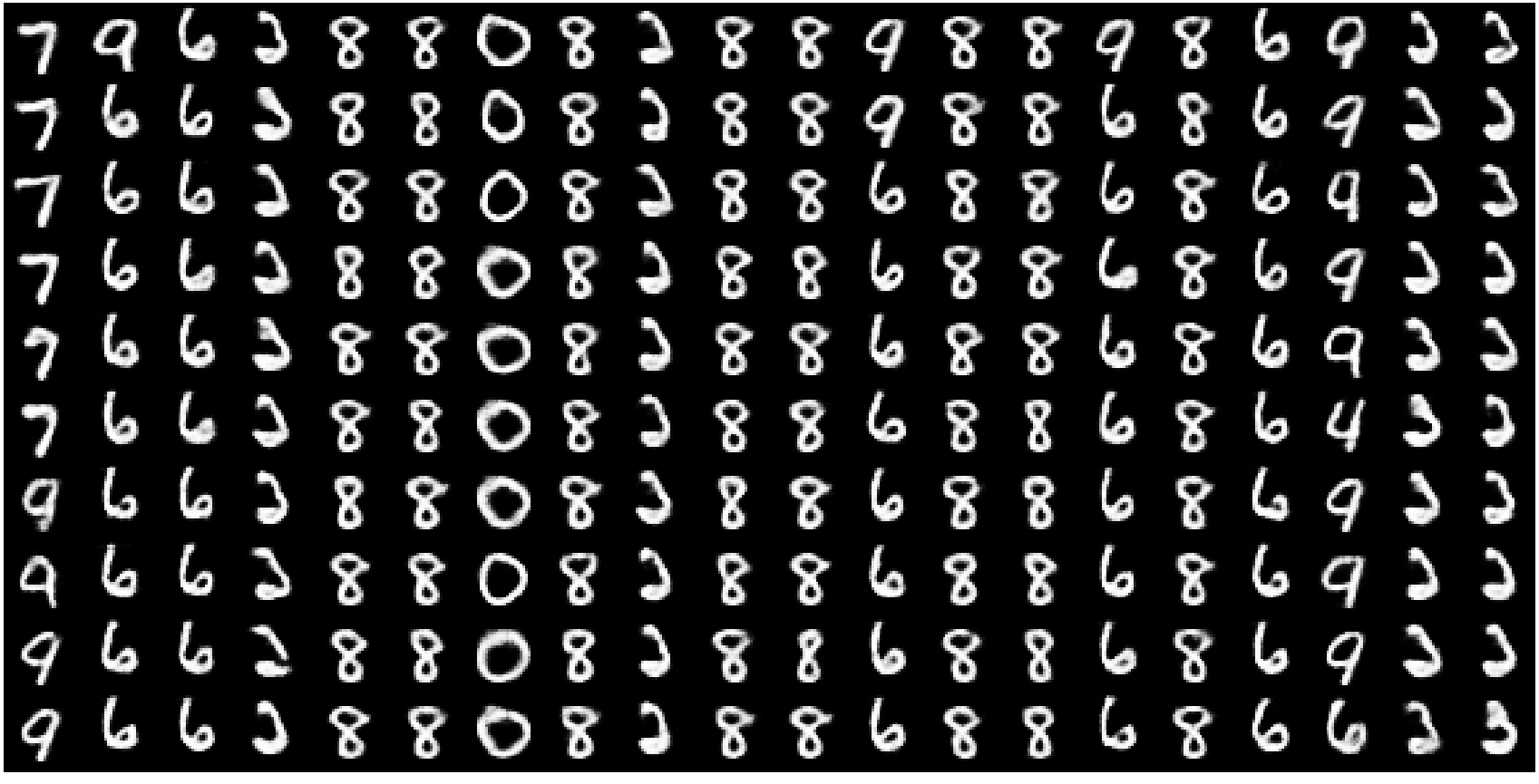
Why do we need this

We want to understand high-dimensional data with rich structure.
"Understanding" $\rightarrow$ Density estimation (true $p(\boldsymbol{x})$ under data)

- generative models
- Missing value imputation (given partial $\boldsymbol{x}$)
- Denoising (return original $\boldsymbol{x}$ given damaged $\tilde{\boldsymbol{x}}$)
- ...

$\rightarrow$ Density estimation (true $p(\boldsymbol{x})$ under data)

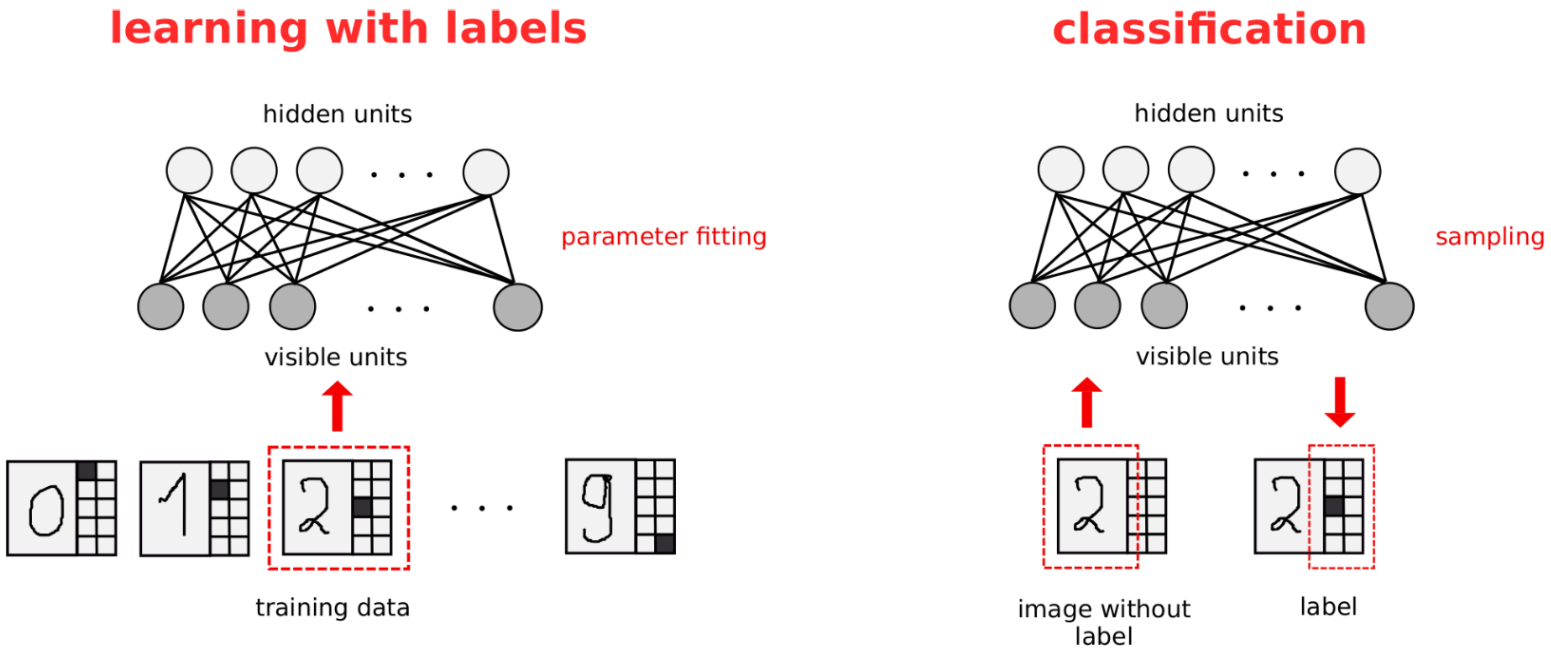
- generative models



[7]

$\rightarrow$ Density estimation (true $p(\boldsymbol{x})$ under data)

- Missing value imputation (given partial $\boldsymbol{x}$)
  $\rightarrow$ classification is actually special case!



$\rightarrow$ Density estimation (true $p(\boldsymbol{x})$ under data)

- Denoising (return original $\boldsymbol{x}$ given damaged $\tilde{\boldsymbol{x}}$)

[14] note: this result uses more advanced concepts based on RBMs

Why this is hard

This requires modelling the underlying probability distribution.

$\rightarrow$ fundamentally problematic

Naive approach to represent probabilitiy distribution has space complexity $\mathcal{O}(n^k)$ (where n is the cardinality of the state space of the probality distribution, k is the number of random variables)

Idea: use conditional structure of probability distribution to reduce complexity

$\rightarrow$ motivation for so called **graphical models**

Graphical models

$G = (V, E)$, V nodes, E edges, defines an **undirected Graph**.

clique is a subset of V in which all nodes are pairwise connected. Call a clique **maximal**, if no node can be added without violating the definition of a clique.

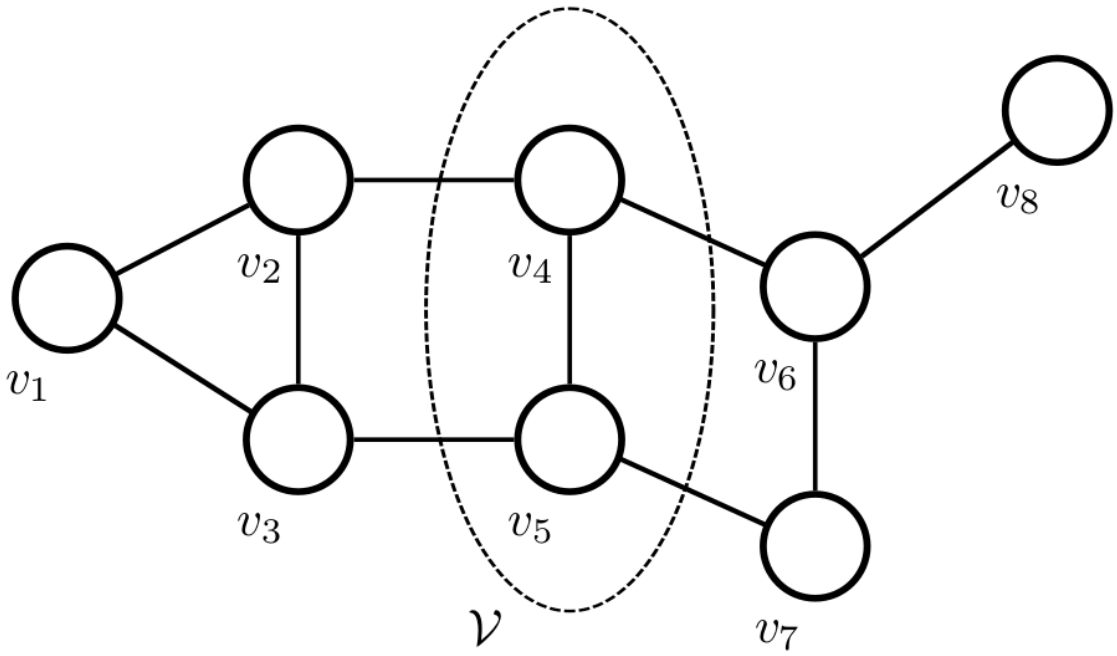


Figure 1: Undirected graph, $\{v_4, v_5\}$ clique, also maximal [6]

Graphical Models

$\rightarrow$ each node $v \in V$ corresponds to random variable X_v

$\rightarrow$ edges encode direct probabilistic interaction

Markov random fields

The set $\boldsymbol{X} = (X_v)_{v \in V}$ is called a **Markov random field (MRF)** iff

$$\forall \boldsymbol{x} : \forall v \in V : p\left(x_v | (x_w)_{w \in V \setminus \{v\}}\right) = p\left(x_v | (x_w)_{w \in \mathcal{N}_v}\right) \tag{1}$$

$$\mathcal{N}_v = \{w \in V : \{w, v\} \in E\} \text{ neighborhood of } v$$

In other words, in a Markov random field all X_v are conditionally independent of all other variables given their neighborhood.

Markov random fields

Theorem (Hammersley-Clifford) [15]

For a Markov random field the following equivalence holds

1) the distribution p is strictly positive

2) p factorizes over the Graph G, i.e.

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}) \tag{2}$$

where the product runs over all maximal cliques.

The $\psi_C(\boldsymbol{x})$ are often referred to as **clique potentials**.

Energy based models and Gibbs distribution

Rewrite (2):

$$p(\boldsymbol{x}) = \frac{1}{Z} \prod_{C \in \mathcal{C}} \psi_C(\boldsymbol{x}) = \frac{1}{Z} e^{\sum_{C \in \mathcal{C}} \ln \psi_C(\boldsymbol{x})} = \frac{1}{Z} e^{-E(\boldsymbol{x})} \tag{3}$$

$\rightarrow$ **Energy based model (EBM)** (alt.: harmony based model)

underlying distribution: **Boltzmann distribution / Gibbs distribution**

$\rightarrow$ + latent variables = **Boltzmann machine**

Problem

general MRFs virtually **untrainable**

Problem

general MRFs virtually **untrainable**

$\rightarrow$ in the most general case (fully connected) no benefit for graphical model!

$\rightarrow$ hard to do something clever without relying on some structure

Restricted Boltzmann machines (Harmonium)

Constraining structure $\rightarrow$ **bipartite graph**

one layer of visible & one layer of hidden/latent variables

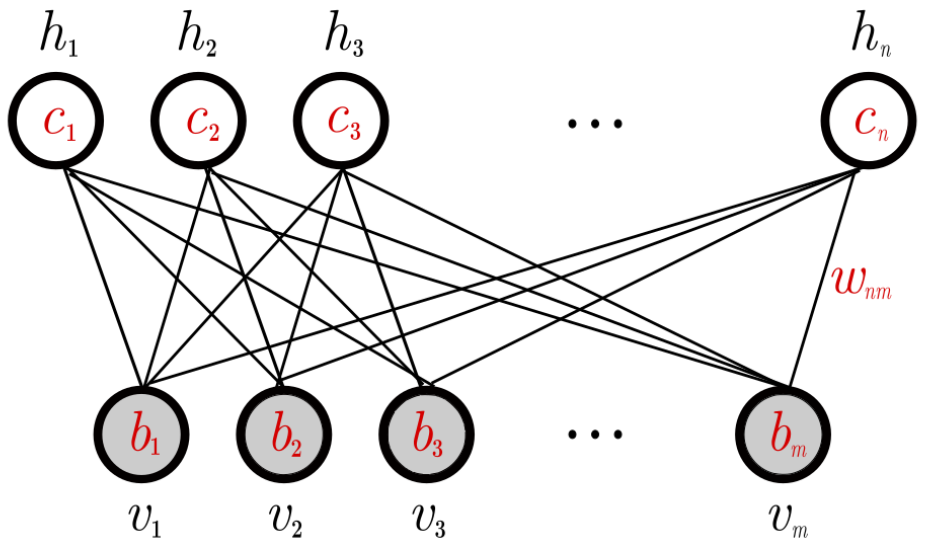
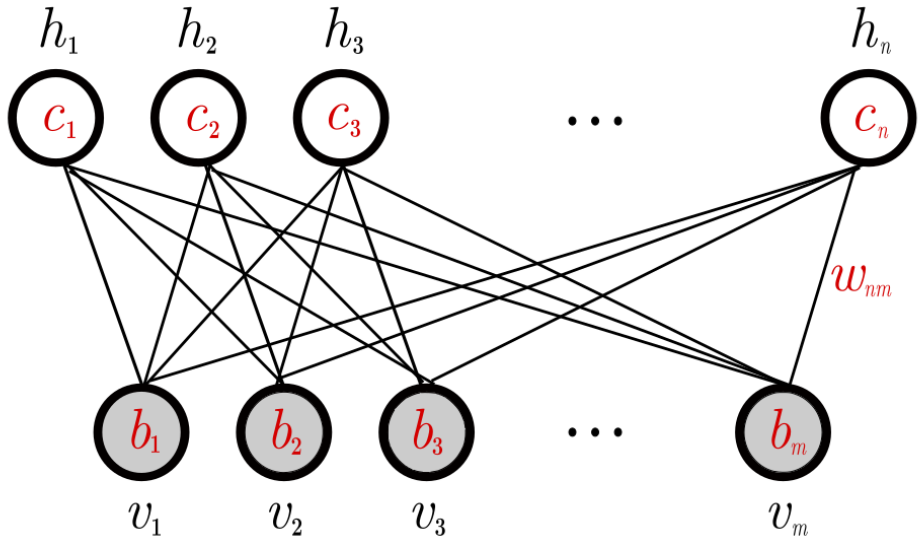
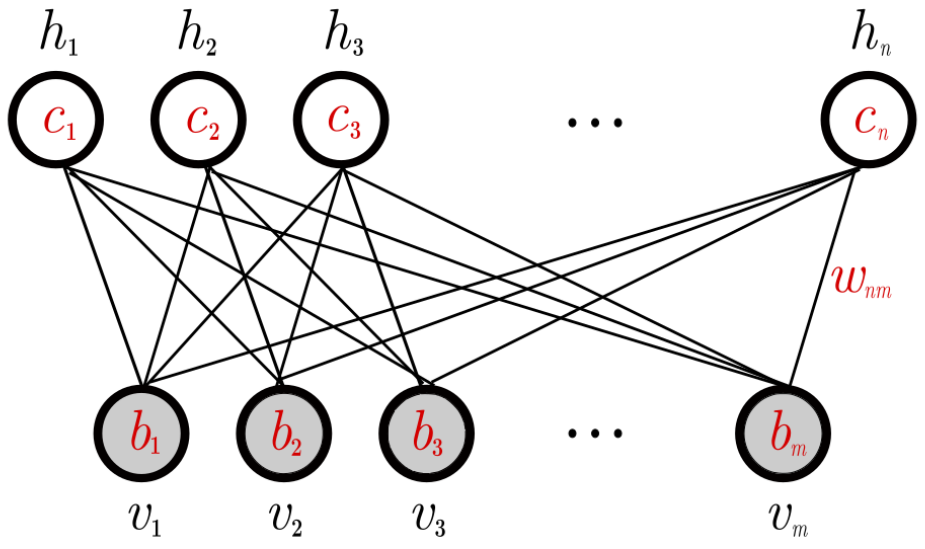


Figure 2: Restricted Boltzmann machine



binary random variables $(\boldsymbol{V}, \boldsymbol{H}) \rightarrow (\boldsymbol{v}, \boldsymbol{h}) \in \{0,1\}^{m+n}$

$$E(\boldsymbol{v}, \boldsymbol{h}) = -\sum_{i=1}^{n}\sum_{j=1}^{m} w_{ij} h_i v_j - \sum_{j=1}^{m} b_j v_j - \sum_{i=1}^{n} c_i h_i \tag{4}$$



takeaway: RBMs are Boltzmann machines with structure & energy constraint

Properties of RBMs

no connections inside each layer

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{i=1}^{n} p\left(h_i|\boldsymbol{v}\right)$$
$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{j=1}^{m} p\left(v_j|\boldsymbol{h}\right) \tag{5}$$

Properties of RBMs

universal approximators for distributions on $\{0,1\}^m$

$\rightarrow$ additionally quite a few other results about universal approximator properties [11] [13]

Sampling from RBMs

How to access what we learn

Calculating partition function of RBM's intractable

$\rightarrow$ **Markov chain Monte Carlo** (MCMC) methods provide ways to sample from $p(\boldsymbol{v}, \boldsymbol{h})$ without knowing Z

Markov Chains

A **Markov chain** is a family of random variables $X = \left\{X^k | k \in \mathbb{N}_0\right\}$ that, $\forall k \geq 0$ and $\forall j, i, i_0, \ldots, i_{k-1} \in \Omega$ satisfies

$$\Pr\left(X^{(k+1)} = j | X^{(k)} = i, \ldots, X^{(0)} = i_0\right) = \Pr\left(X^{(k+1)} = j | X^{(k)} = i\right) \tag{6}$$

Markov Chains

Transition matrix $\mathbf{P} = \left(\Pr\left(X^{(k+1)} = j | X^{(k)} = i\right)\right)_{i,j\in\Omega} = (p_{ij})_{i,j\in\Omega}$

stationary distribution π for which $\pi^{\mathrm{T}} = \pi^{\mathrm{T}}\mathbf{P}$

Detailed balance condition $\pi(i)p_{ij} = \pi(j)p_{ji}$ sufficient for stationarity of π

Markov Chains

irreducible (can get from any state in Ω to any other in finite number of transitions)

$$\forall s_i, s_j \in \Omega : \exists n \in \mathbb{N} : P\left(X_n = s_i | X_0 = s_j\right) > 0$$

aperiodic (every state can reoccur at irregular times)

$$\gcd\left\{n > 0 : \Pr\left(X_n = i | X_0 = i\right) > 0\right\} = 1$$

$\Rightarrow$ Markov chain over finite Ω converges to its stationary distribution

Gibbs sampling

Idea

Construct Markov chain by updating each variable based on its conditional distribution given state of the others

Gibbs sampling - Metropolis-Hastings

MRF $\boldsymbol{X} = (X_1, \ldots, X_N)$ w.r.t. $G = (V, E)$ and $p(\boldsymbol{x}) = \frac{1}{Z} e^{-E(\boldsymbol{x})}$, then produce new state as follows:

1. Pick $X_i, i \in V$ at random with strictly positive $q(i)$
2. Transition between $\boldsymbol{x}$ and $\boldsymbol{y}$, $\boldsymbol{x} \neq \boldsymbol{y}$ with $p_{\boldsymbol{xy}} =$
$$\begin{cases} q(i)p\left(y_i | \left(x_v\right)_{v \in \mathcal{N}_i}\right), & \text{if } \exists i \in V \text{ so that } \forall v \in V \text{ with } v \neq i : x_v = y_v \\ 0, & \text{else} \end{cases}$$
and stay in $\boldsymbol{x}$ with $p_{\boldsymbol{xx}} = \sum_{i \in V} q(i)p\left(x_i | \left(x_v\right)_{v \in \mathcal{N}_i}\right)$

Gibbs sampling

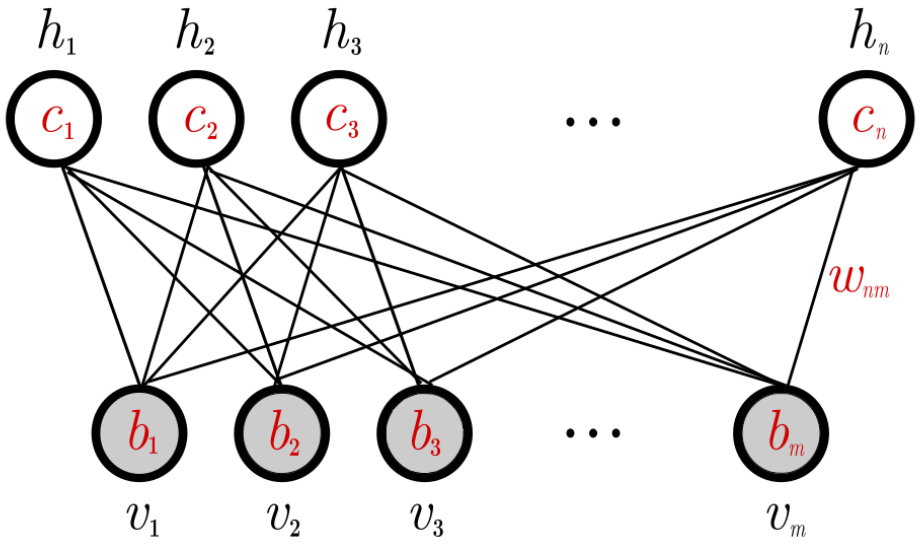
$p(\boldsymbol{x}) = \frac{1}{Z} e^{-E(\boldsymbol{x})}$ and conditional probabilities strictly positive

$\rightarrow$ Chain irreducible and aperiodic

$\rightarrow$ Chain converges to stationary distribution

$\rightarrow$ $p(\boldsymbol{x}) = \frac{1}{Z} e^{-E(\boldsymbol{x})}$ stationary distribution (detailed balance)

k-step Block Gibbs sampling in RBM's



no intralayer connections in RBM's

$\rightarrow$ sample states of all variables in one layer jointly

Algorithm 1 Block Gibbs sampling in RBM's

Input RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, Gibbs steps k
Output sample $(\boldsymbol{v}^{(k)}, \boldsymbol{h}^{(k)})$ after k Gibbs steps
$\boldsymbol{v}^{(0)} \leftarrow$ random_vector
for $t = 0, \ldots, k-1$ **do**
    for $i = 1..n$ **do**
        Sample $h_i^{(t)} \sim p\left(h_i | \boldsymbol{v}^{(t)}\right)$
    end for
    for $j = 1..m$ **do**
        Sample $v_j^{(t+1)} \sim p\left(v_j | \boldsymbol{h}^{(t)}\right)$
    end for
end for

Training RBMs

What to minimize

Gibbs sampling helps once we have parameters

But how to learn them?

$\rightarrow$ minimize distance between (unknown) data distribution q and RBM-distribution p in terms of **Kullback-Leibler divergence**

KL and log-likelihood

Minimizing

$$\mathrm{KL}(q\|p) = \sum_{\boldsymbol{x}\in\Omega} q(\boldsymbol{x}) \ln \frac{q(\boldsymbol{x})}{p(\boldsymbol{x})} = \sum_{\boldsymbol{x}\in\Omega} q(\boldsymbol{x}) \ln q(\boldsymbol{x}) - \sum_{\boldsymbol{x}\in\Omega} q(\boldsymbol{x}) \ln p(\boldsymbol{x}) \quad (7)$$

$\Longleftrightarrow$

Maximizing **log-likelihood**

$$\ln \mathcal{L}(\boldsymbol{\theta}|S) = \ln \prod_{i=1}^{\ell} p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right) = \sum_{i=1}^{\ell} \ln p\left(\boldsymbol{x}_i|\boldsymbol{\theta}\right) \quad (8)$$

over training data $S = \{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_\ell\}$ w.r.t. parameters $\boldsymbol{\theta}$

Unsupervised RBM learning

$\rightarrow$ Learning $=$ gradient ascent on log-likelihood with learning rate $\eta \in \mathbb{R}^+$,

$$\boldsymbol{\theta}^{(t+1)} = \boldsymbol{\theta}^{(t)} + \eta \frac{\partial}{\partial \boldsymbol{\theta}^{(t)}} \left(\ln \mathcal{L} \left(\boldsymbol{\theta}^{(t)} | S \right) \right) \tag{9}$$

Unsupervised RBM learning

For single training example $\boldsymbol{v}$,

$$\ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v}) = \ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} - \ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}$$

$$\Rightarrow \frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial \boldsymbol{\theta}} = \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right) - \frac{\partial}{\partial \boldsymbol{\theta}} \left(\ln \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \right)$$

$$= -\frac{1}{\sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}} + \frac{1}{\sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})}} \sum_{\boldsymbol{v},\boldsymbol{h}} e^{-E(\boldsymbol{v},\boldsymbol{h})} \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial \boldsymbol{\theta}} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} + \sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial \boldsymbol{\theta}} \qquad (10)$$

The first term in (10) is referred to as **positive phase** the second as **negative phase**.

Positive and negative phase

Positive phase

$-\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$

$\rightarrow$ finds hidden configurations that work well with $\boldsymbol{v}$ and lowers their energies

Negative Phase

$\sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial \boldsymbol{\theta}}$

$\rightarrow$ finds joint configuration that are the "best competitors" and raises their energies

Positive phase in RBM's

$\rightarrow$ generally very easy for RBM's exploiting independence property (5):

$$p(\boldsymbol{h}|\boldsymbol{v}) = \prod_{i=1}^{n} p\left(h_i|\boldsymbol{v}\right)$$

$$p(\boldsymbol{v}|\boldsymbol{h}) = \prod_{j=1}^{m} p\left(v_j|\boldsymbol{h}\right)$$

For example, positive Phase w.r.t. w_{ij} (and similarly for b_i, c_j):

Let $\boldsymbol{h}_{-i}$ denote state of all hidden variables but the i-th one, then

$$-\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v})\frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} = \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) h_i v_j = \sum_{\boldsymbol{h}} \prod_{k=1}^{n} p\left(h_k|\boldsymbol{v}\right) h_i v_j$$

$$= \sum_{h_i}\sum_{\boldsymbol{h_{-i}}} p\left(h_i|\boldsymbol{v}\right) p\left(\boldsymbol{h}_{-i}|\boldsymbol{v}\right) h_i v_j = \sum_{h_i} p\left(h_i|\boldsymbol{v}\right) h_i v_j \sum_{\boldsymbol{h_{-i}}} p\left(\boldsymbol{h}_{-i}|\boldsymbol{v}\right)$$

$$= p\left(H_i = 1|\boldsymbol{v}\right) v_j \tag{11}$$

$\rightarrow$ saves us from exponential sum in number of hidden variables

Negative phase in RBM's

We're not so lucky here!

$\rightarrow$ same trick gets rid only of either $\sum_{\boldsymbol{v}}$ or $\sum_{\boldsymbol{h}}$ in $\sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}$

For example, getting rid of $\sum_{\boldsymbol{h}}$ leaves:

$$\sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} = -\sum_{\boldsymbol{v}} p(\boldsymbol{v}) \sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) h_i v_j$$

$$= -\sum_{\boldsymbol{v}} p(\boldsymbol{v}) p\left(H_i = 1|\boldsymbol{v}\right) v_j \tag{12}$$

$\rightarrow$ sum remains exponential either in hidden or visible variables

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v}, \boldsymbol{h}} p(\boldsymbol{v}, \boldsymbol{h}) \frac{\partial E(\boldsymbol{v}, \boldsymbol{h})}{\partial w_{ij}}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}$$

$$= p\left(H_i = 1|\boldsymbol{v}\right) v_j + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}$$

$$\begin{aligned}
\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} \\
&= p\left(H_i = 1|\boldsymbol{v}\right) v_j + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} \\
&= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) h_i v_j
\end{aligned}$$

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}$$

$$= p\left(H_i = 1|\boldsymbol{v}\right) v_j + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}$$

$$= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) h_i v_j$$

$$= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p\left(H_i = 1|\boldsymbol{v}\right) v_j$$

$$\begin{aligned}
\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} &= -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} \\
&= p\left(H_i = 1|\boldsymbol{v}\right) v_j + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} \\
&= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) h_i v_j \\
&= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p\left(H_i = 1|\boldsymbol{v}\right) v_j \\
&= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) h_i v_j
\end{aligned}$$

Training algorithms

For gradient ascent, need a way to compute second term:

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} = p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) h_i v_j$$

$\rightarrow$ approximate this sum by MCMC methods

$$\sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) h_i v_j$$

$\rightarrow$ think of sum as expectation value

$\rightarrow$ Gibbs sampling $\rightarrow$ approximate as average over a few samples

Algorithm 2 Naive MCMC gradient estimation

Input RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S
Output Δw_{ij} for $i = 1, \ldots, n, j = 1, \ldots, m$
init: $\Delta w_{ij} = 0$ for $i = 1, \ldots, n, j = 1, \ldots, m$
for all $\boldsymbol{v} \in S$ **do**
    for l=1..ξ **do**
        $\boldsymbol{v}^{(0)} \leftarrow$ random_vector
        for $t = 0, \ldots, k-1$ **do**
            for $i = 1..n$ **do**
                Sample $h_i^{(t)} \sim p\left(h_i | \boldsymbol{v}^{(t)}\right)$
            end for
            for $j = 1..m$ **do**
                Sample $v_j^{(t+1)} \sim p\left(v_j | \boldsymbol{h}^{(t)}\right)$
            end for
        end for
    end for ▷ here ξ $(\boldsymbol{v}, \boldsymbol{h})$ pairs (approximately from $p(\boldsymbol{v}, \boldsymbol{h})$) $\rightarrow$ call these $(\boldsymbol{v}^l, \boldsymbol{h}^l)$
    for $i = 1, \ldots, n, j = 1, \ldots, m$ **do**
        $\Delta w_{ij} \leftarrow \Delta w_{ij} + p\left(H_i = 1 | \boldsymbol{v}\right) \cdot v_j - \frac{1}{\xi} \sum_{l=1..\xi} h_i^l v_j^l$
    end for
end for

Problem

requires running a lot of Gibbs chains (computationally expensive)

$\rightarrow$ take a step back, tackle the exponential sum again

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} = -\sum_{\boldsymbol{h}} p(\boldsymbol{h}|\boldsymbol{v}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}} + \sum_{\boldsymbol{v},\boldsymbol{h}} p(\boldsymbol{v},\boldsymbol{h}) \frac{\partial E(\boldsymbol{v},\boldsymbol{h})}{\partial w_{ij}}$$

$$= p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p\left(H_i = 1|\boldsymbol{v}\right) v_j$$

Contrastive divergence $CD - k$

- initialize Gibbs chain by data distribution $\rightarrow$ less time to burn in
- approximate the expectation value using a single sampled $\boldsymbol{v}^{(k)}$ (that is after k Gibbs steps)

Contrastive divergence $CD - k$

- initialize Gibbs chain by data distribution $\rightarrow$ less time to burn in
- approximate the expectation value using a single sampled $\boldsymbol{v}^{(k)}$ (that is after k Gibbs steps)

$$\frac{\partial \ln \mathcal{L}(\boldsymbol{\theta}|\boldsymbol{v})}{\partial w_{ij}} = p\left(H_i = 1|\boldsymbol{v}\right) v_j - \sum_{\boldsymbol{v}} p(\boldsymbol{v}) p\left(H_i = 1|\boldsymbol{v}\right) v_j$$

$$\approx p\left(H_i = 1|\boldsymbol{v}\right) v_j - p(H_i = 1|\boldsymbol{v}^{(k)}) v_j^{(k)}$$

Algorithm 3 k-step contrastive divergence [1]

Input RBM $(V_1, \ldots, V_m, H_1, \ldots, H_n)$, training batch S
Output $\Delta w_{ij}, \Delta b_j$ and Δc_i for $i = 1, \ldots, n, j = 1, \ldots, m$
init: $\Delta w_{ij} = \Delta b_j = \Delta c_i = 0$ for $i = 1, \ldots, n, j = 1, \ldots, m$
for all $\boldsymbol{v} \in S$ **do**
    $\boldsymbol{v}^{(0)} \leftarrow \boldsymbol{v}$
    for $t = 0, \ldots, k-1$ **do**
        for $i = 1..n$ **do**
            Sample $h_i^{(t)} \sim p\left(h_i | \boldsymbol{v}^{(t)}\right)$
        end for
        for $j = 1..m$ **do**
            Sample $v_j^{(t+1)} \sim p\left(v_j | \boldsymbol{h}^{(t)}\right)$
        end for
    end for
    for $i = 1, \ldots, n, j = 1, \ldots, m$ **do**
        $\Delta w_{ij} \leftarrow \Delta w_{ij} + p\left(H_i = 1 | \boldsymbol{v}^{(0)}\right) \cdot v_j^{(0)} - p\left(H_i = 1 | \boldsymbol{v}^{(k)}\right) \cdot v_j^{(k)}$
    end for
    for $j = 1..m$ **do**
        $\Delta b_j \leftarrow \Delta b_j + v_j^{(0)} - v_j^{(k)}$
    end for
    for $j = 1..n$ **do**
        $\Delta c_i \leftarrow \Delta c_i + p\left(H_i = 1 | \boldsymbol{v}^{(0)}\right) - p\left(H_i = 1 | \boldsymbol{v}^{(k)}\right)$
    end for
end for

Why does this even work?

- non-trivial and surprising when first introduced
- does not follow gradient of log-likelihood
  $\rightarrow$ important results give conditions for convergence and bounds on the expectation error, empirical observations support this further ([3, 4, 5, 17])

Why does this even work?

- non-trivial and surprising when first introduced
- does not follow gradient of log-likelihood
  $\rightarrow$ important results give conditions for convergence and bounds on the expectation error, empirical observations support this further ([3, 4, 5, 17])
- even more surprising: in practice, very often $k = 1$ [9]

Persistent contrastive divergence (PCD)

do not reinitialize Gibbs chains, but keep them permanently

$\rightarrow$ update step is small enough so distribution keeps close to previous step

Where to go from here?

- better training
  - fast persistent contrastive divergence (FPCD)
  - parallel tempering
- other models
  - Deep Belief Networks
  - Conditional Boltzmann machines
  - ...

References i


D. H. Ackley, G. E. Hinton, and T. J. Sejnowski.
A learning algorithm for boltzmann machines.
Cognitive science, 9(1):147–169, 1985.


Y. Bengio.
Learning deep architectures for ai.
Foundations and Trends in Machine Learning, 2(1):1127, 2009.


Y. Bengio and O. Delalleau.
Justifying and generalizing contrastive divergence.
Neural computation, 21(6):1601–1621, 2009.


M. A. Carreira-Perpinan and G. E. Hinton.
On contrastive divergence learning.
In *Aistats*, volume 10, pages 33–40. Citeseer, 2005.

References ii

A. Fischer and C. Igel.
Bounding the bias of contrastive divergence learning.
Neural computation, 23(3):664–673, 2011.

A. Fischer and C. Igel.
Training restricted boltzmann machines: An introduction.
Pattern Recognition, 47(1):25–39, Jan. 2014.

I. Goodfellow, Y. Bengio, and A. Courville.
Deep learning.
MIT press, 2016.

G. E. Hinton.
Training products of experts by minimizing contrastive divergence.
Neural computation, 14(8):1771–1800, 2002.

References iii

G. E. Hinton.

A practical guide to training restricted boltzmann machines.

In *Neural networks: Tricks of the trade*, pages 599–619. Springer, 2012.

G. E. Hinton, S. Osindero, and Y.-W. Teh.

A fast learning algorithm for deep belief nets.

Neural computation, 18(7):1527–1554, 2006.

D. MacKay.

Failures of the one-step learning algorithm.

In *Available electronically at http://www. inference. phy. cam. ac. uk/mackay/abstracts/gbm. html*, 2001.

References iv


P. Mehta, M. Bukov, C.-H. Wang, A. G. Day, C. Richardson, C. K. Fisher, and D. J. Schwab.
A high-bias, low-variance introduction to machine learning for physicists.
Physics Reports, 2019.


G. Montufar and N. Ay.
Refinements of universal approximation results for deep belief networks and restricted boltzmann machines.
Neural Computation, 23(5):1306–1319, 2011.


V. Parmar and D. M. Suri.
Design exploration of hybrid cmos-oxram deep generative architectures.
01 2018.

References v

B. Pierre.

Markov chains: Gibbs fields, Monte Carlo simulation, and queues.

Springer, 2001.

P. Smolensky.

Information processing in dynamical systems: Foundations of harmony theory.

Technical report, Colorado Univ at Boulder Dept of Computer Science, 1986.

A. L. Yuille.

The convergence of contrastive divergences.

In *Advances in neural information processing systems*, pages 1593–1600, 2005.

Error Bound CD-k

Let p denote the marginal distribution of the visible units of an RBM and let q be the empirical distribution defined by a set of samples $v_1, \ldots, v_l$. Then an upper bound on the expectation of the error of the CD-k approximation of the log-likelihood derivative w.r.t some RBM parameter Θ_a is given by

$$\left| E_{q\left(\boldsymbol{v}^{(0)}\right)} \left[E_{p\left(\boldsymbol{v}^{(k)} | \boldsymbol{v}^{(0)}\right)} \left[\frac{\partial \ln p\left(\boldsymbol{v}^{(k)}\right)}{\partial \theta_a} \right] \right| \leq \frac{1}{2} \left| q - p \right| \left(1 - e^{-(m+n)\Delta}\right)^k$$

with

$$\Delta = \max \left\{ \max_{l \in \{1,\ldots,m\}} \vartheta_l, \max_{l \in \{1,\ldots,n\}} \xi_l \right\}$$
$$\vartheta_l = \max \left\{ \left| \sum_{i=1}^{n} I_{\{w_{il} > 0\}} w_{il} + b_l \right|, \left| \sum_{i=1}^{n} I_{\{w_{il} < 0\}} w_{il} + b_l \right| \right\}$$
$$\xi_l = \max \left\{ \left| \sum_{j=1}^{m} I_{\{w_{lj} > 0\}} w_{lj} + c_l \right|, \left| \sum_{j=1}^{m} I_{\{w_{lj} < 0\}} w_{lj} + c_l \right| \right\}$$

RBMs and neural networks

Why is $p(H_i = 1|\boldsymbol{v})$ in positive phase computationally feasible?

$\rightarrow$ Z cancels out; explicit calculation shows:

$$p(H_i = 1|\boldsymbol{v}) = \text{sig}\left(\sum_{j=1}^{m} w_{ij} v_j + c_i\right)$$
$$p(V_j = 1|\boldsymbol{h}) = \text{sig}\left(\sum_{i=1}^{n} w_{ij} h_i + b_j\right)$$

Identify:

single variable $\rightarrow$ neuron

conditional probability $\rightarrow$ sigmoid activation function

RBM $\rightarrow$ **stochastic neural network**