# Boltzmann Machine

## Overview and Prospects

Hooman Zolfaghari
Ramtin Moslemi
Borna (lastname)
Houman (lastname)

# Machine Learning & Physics

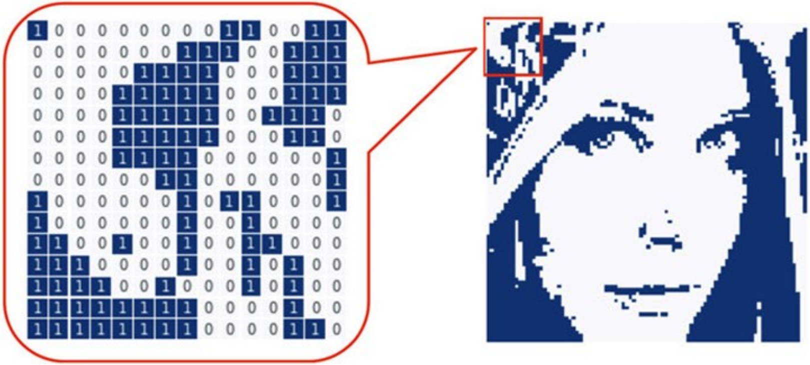
# How are they related

- Bridge between physics and machine learning: **information**
- Amount of information = Extent of surprise
- Amount of information of event $A = -\log \mathbb{P}(\text{event } A)$.

Little “information” $\Leftrightarrow$ difficult to predict $\Leftrightarrow$ large information entrop

A lot of “information” $\Leftrightarrow$ easy to predict $\Leftrightarrow$ small information entropy

# Maxwell’s demon & Drawing Decisions



# Machine Learning Overview

> "*Intelligence is not just about pattern recognition and function approximation. It's about modeling the world*". — Josh Tenenbaum, NeurIPS 2021.

- We have an observable space $\mathcal{Z}$, a probability measure $\mathbb{P}$, a sample $S = \{Z_i\}_{i=1}^{n}$.
- Formulate the problem with a Hypothesis set $\mathcal{H}$ and a "Loss function".
- (Ultimate goal) Statistical Risk: $\arg\min_{h \in \mathcal{H}} R(h) = \mathbb{E}[\ell(h)]$
- Empirical Risk: $\hat{R}_S(h) = \frac{1}{n} \sum_{Z \in S} \ell(h(Z))$

# Artificial Neural Network

- A feed-forward neural network defines a function $f : \mathbb{R}^d \rightarrow \mathbb{R}^k$ recursively via layers .

$$f(X) = f^{(L)} \circ f^{(L-1)} \circ \cdots \circ f^{(1)}(X) \quad f^{(i)}(X^{(i)}) = \sigma(WX^{(i)} + b)$$

**Universal Approximation Theorem:**

For any continuous function $g : \mathbb{R}^d \rightarrow \mathbb{R}^k$ on a compact subset $K \subset \mathbb{R}^d$ and for any $\varepsilon > 0$, there exists a neural network $f \in \mathcal{H}$ such that:

$$\sup_{x \in K} \left| g(x) - f(x) \right| < \varepsilon.$$

# Training ANN

- **Gradient Descent:**

$$\theta \to f, \quad \theta_{t+1} = \theta_t - \eta \nabla_\theta \hat{R}_n(f)$$

- In infinite-width limit, the distribution over outputs of a randomly initialized neural network converges to a Gaussian Process
- In gradient descent, the evolution of the network's predictions can be approximated by a linear model characterized by the NTK:

$$\Theta(x, x') = \langle \nabla_\theta f(x; \theta), \nabla_\theta f(x'; \theta) \rangle.$$

- In infinite-width limit, the NTK can guarantee convergence under some assumptions.

# Energy Based Models

# Outsource

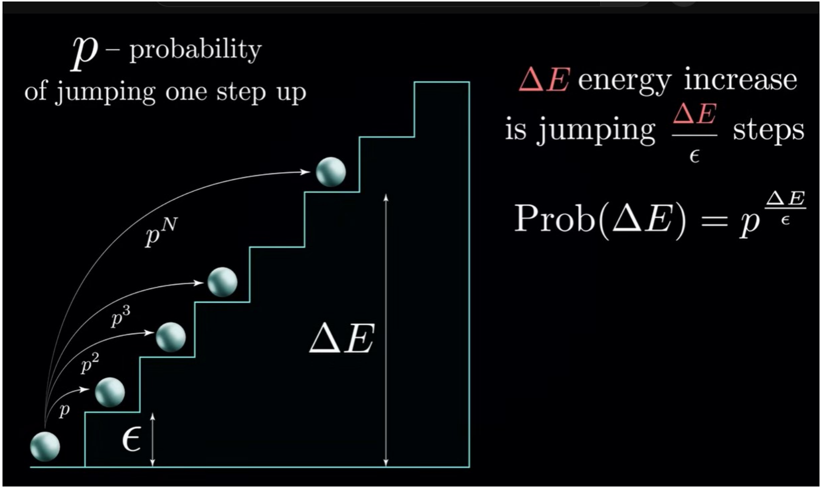
Outsourcing

# Boltzman Machine

# Introduction

- Idea was taken from statistical physics and formulated by cognitive scientists.
- The Hopfield network can reproduce specific patterns it has memorized
- Goal: "Understanding" the data instead of only memorizing
- Boltzmann machine can generate new patterns it has never seen before.
- Multiple potential outputs (non-deterministic)
- BM = Hopfield network + Stochasticity + Hidden Units

# Introduction



# Introduction

- We can formulate this by denoting $T = \frac{-\ln(p)}{\epsilon}$, as:

$$\mathbb{P}(\Delta E) = e^{\frac{-\Delta E}{kT}} \tag{1}$$

- This gives us relative probability of changing energy.
- Knowing all probabilities have volume one, gives us absolute probability of each state.
- Giving us the partition function as the total volume.

$$\mathbb{P}(E) = \frac{1}{Z} e^{-E/T} \tag{2}$$

# Inference

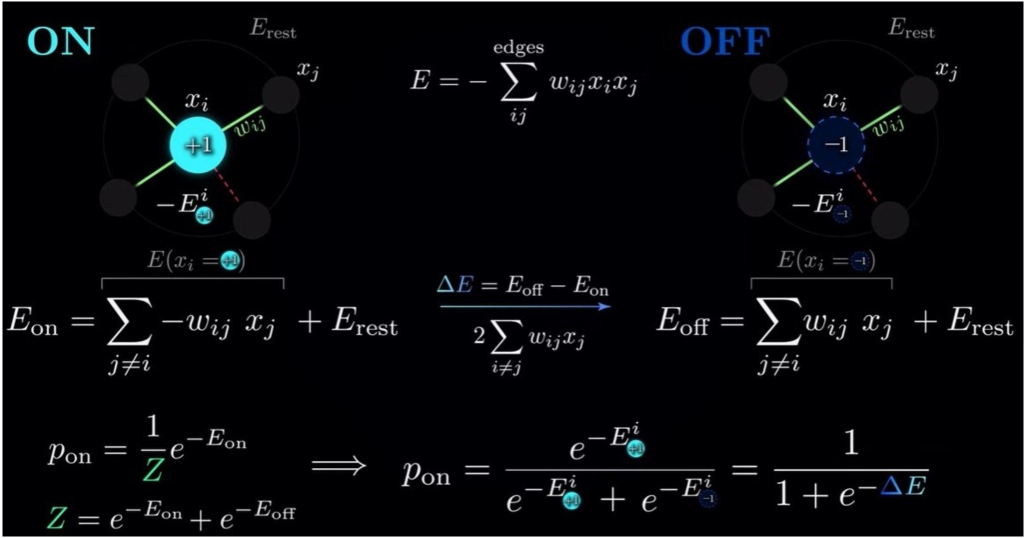
- Hopfield Network's **deterministic** update rule:

$$h_i = \sum_{j \neq i} w_{ij} x_j \to \text{input to neuron } i, \quad x_i = \begin{cases} +1 & h_i > 0 \\ -1 & h_i < 0 \end{cases} \to \text{update rule}$$

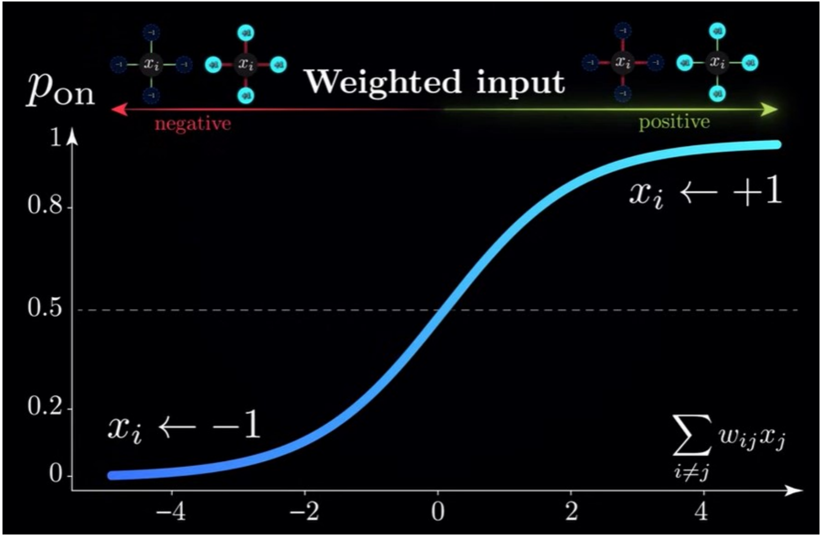
- Always Greedy moving to the lowest energy state possible
- Boltzmann Machine update rule:

$$x_i = \begin{cases} +1 & \text{with probability } \mathbb{P}(E_{\text{on}}) \\ -1 & \text{with probability } 1 - \mathbb{P}(E_{\text{off}}) \end{cases}$$

# Inference

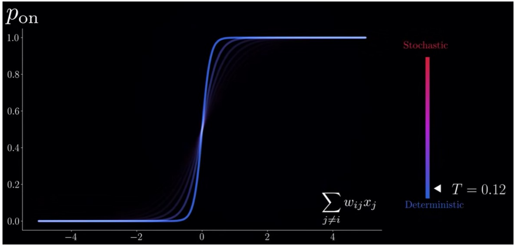

ON
OFF
$E_{\text{rest}}$
$x_j$
$x_i$
$w_{ij}$
$+1$
$-1$
$-E^i_{+1}$
$-E^i_{-1}$
$E = -\sum_{ij}^{\text{edges}} w_{ij} x_i x_j$
$E_{\text{on}} = \overbrace{\sum_{j \neq i} -w_{ij}\, x_j}^{E(x_i = +1)} + E_{\text{rest}}$
$\Delta E = E_{\text{off}} - E_{\text{on}}$
$2 \sum_{i \neq j} w_{ij} x_j$
$E_{\text{off}} = \overbrace{\sum_{j \neq i} w_{ij}\, x_j}^{E(x_i = -1)} + E_{\text{rest}}$
$p_{\text{on}} = \frac{1}{Z} e^{-E_{\text{on}}}$
$Z = e^{-E_{\text{on}}} + e^{-E_{\text{off}}}$
$\Longrightarrow$
$p_{\text{on}} = \frac{e^{-E^i_{+1}}}{e^{-E^i_{+1}} + e^{-E^i_{-1}}} = \frac{1}{1 + e^{-\Delta E}}$

# Inference

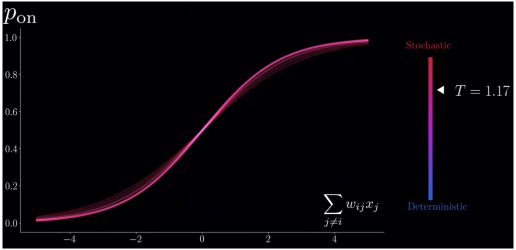


$p_{\text{on}}$
$x_i$
$x_i$
Weighted input
$x_i$
$x_i$
negative
positive
1
0.8
0.5
0.2
0
$x_i \leftarrow +1$
$x_i \leftarrow -1$
$\sum_{i \neq j} w_{ij} x_j$
−4
−2
0
2
4

# Inference



$p_{\text{on}}$
Stochastic
$T = 0.12$
Deterministic
$\sum_{j \neq i} w_{ij} x_j$

$p_{\text{on}}$
Stochastic
$T = 1.17$
Deterministic
$\sum_{j \neq i} w_{ij} x_j$

# Learning

- Instead of memorizing patterns we want to learn probability distribution of data.
- We want the training data to have high probability
- Notice that increasing a states probability affects other states probability through partition function

$$\log \mathbb{P}(\text{data states}) = \sum_{i=1}^{N} \log(\frac{1}{Z} e^{-E(x^{(i)})/T})$$

# Learning

$$\underbrace{\log \mathbb{P}(\text{data states})}_{\text{Maximaize}} = -\frac{1}{T}[\sum_{i=1}^{N} \underbrace{E(x^{(i)})}_{\text{Minimize}}] - N \underbrace{\log Z}_{\text{Minimize}}$$

- Taking derivative with each $w_{ij}$:
  - First term:( similar to Hopfield) $\frac{\partial E(x^{(i)})}{\partial w_{ij}} = -(x_i x_j)$
  - Second term: $\frac{\partial \log Z}{\partial w_{ij}} = \sum_{s \in \text{all states}} \mathbb{P}(s) x_i^s x_j^s$
- Giving us the **Contrastive Hebbian Rule**:

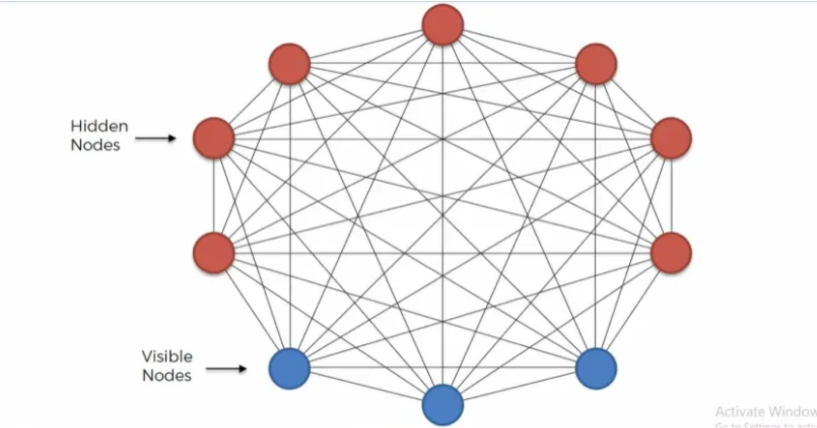
$$\Delta w_{ij} = \frac{1}{N} \sum_{n=1}^{N} x_i^{(n)} x_j^{(n)} - \mathbb{E}[x_i^s x_j^s]$$

- second term computed by iterative sampling until equilibrium.

# Hidden Units

- Visible Units: States directly encoding the data
- Hidden Units: Internal representation of latent variables
- We use the same update rule in inference
- In Learning, input the data and sample the hidden state

# Hidden Units



# Energy Based Model

- Energy function:

$$E(x;\theta) = -(\mathbf{x}^{\mathbf{T}}\mathbf{W}\mathbf{x} + \mathbf{b}^{\mathbf{T}}\mathbf{x})$$

- Parameters same as in *Hopfield* networks and *Ising* models
- Problem: Hard to train (due to the partition function)
- Solution:
  - Introducing latent variables
  - Restricting connections among observables.

# Restricting BMs

- Consider: binary observable variables $\mathbf{x} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{D}}$ and binary latent (hidden) variables $\mathbf{z} \in \{\mathbf{0}, \mathbf{1}\}^{\mathbf{M}}$
- The relationships among variables specified through this *energy function*:

$$E(\mathbf{x}, \mathbf{z}; \theta) = -\mathbf{x}^{\mathbf{T}}\mathbf{W}\mathbf{z} - \mathbf{b}^{\mathbf{T}}\mathbf{x} - \mathbf{c}^{\mathbf{T}}\mathbf{x} \tag{3}$$

- For this EF, the RBM is defined by the *Gibbs* distribution:

$$p(\mathbf{x}, \mathbf{z}; \theta) = \frac{\mathbf{1}}{\mathbf{Z}_{\theta}} \exp(-\mathbf{E}(\mathbf{x}, \mathbf{z}; \theta)) \tag{4}$$

# Restricting BMs

- the *partition function*:

$$Z_\theta = \sum_{\mathbf{x}} \sum_{\mathbf{z}} \exp\big( - E(\mathbf{x}, \mathbf{z}; \theta)\big) \tag{5}$$

- The marginal probability over observables (the likelihood of observation):

$$p(\mathbf{x}|\theta) = \frac{1}{Z_\theta} \exp\big( - F(\mathbf{x}; \theta)\big) \tag{6}$$

- where $F(\cdot)$ is the *free energy*:

$$F(\mathbf{x}; \theta) = -\mathbf{b}^\top \mathbf{x} - \sum_j \log\big(1 + \exp(c_j + (\mathbf{W}_{\cdot j})^\top \mathbf{x})\big). \tag{7}$$

# RBM

- The presented model is called a restricted Boltzmann machine (RBM).
- Useful property: the conditional distribution over the hidden variables factorizes given the observable variables and vice versa:

$$p(z_m = 1|\mathbf{x}, \theta) = \mathrm{sigm}\big(c_m + (\mathbf{W}_{\cdot m})^\top \mathbf{x}\big), \tag{8}$$

$$p(x_d = 1|\mathbf{z}, \theta) = \mathrm{sigm}\big(b_d + \mathbf{W}_{d\cdot}\mathbf{z}\big). \tag{9}$$

# Learning RBMs

- For given data $\mathcal{D} = \{\mathbf{x}_n\}_{n=1}^{N}$
- Train an RBM using the maximum likelihood

$$\ell(\theta) = \frac{1}{N} \sum_{\mathbf{x}_n \in \mathcal{X}} \log p(\mathbf{x}_n \mid \theta) \tag{10}$$

- The gradient with respect to $\theta$:

$$\nabla_\theta \ell(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \left( \nabla_\theta F(\mathbf{x}_n; \theta) - \sum_{\hat{\mathbf{x}}} p(\hat{\mathbf{x}}|\theta) \nabla_\theta F(\hat{\mathbf{x}}; \theta) \right) \tag{11}$$

- Cannot be computed analytically because the second term requires summing over all configurations of observables.

# Learning RBMs

- One way to sidestep this: Stochastic approximation
- Replacing the expectation under $p(\mathbf{x}|\theta)$ by a sum over $S$ samples $\{\hat{\mathbf{x}}_1, \ldots, \hat{\mathbf{x}}_S\}$ drawn according to $p(\mathbf{x}|\theta)$:

$$\nabla_\theta \ell(\theta) \approx -\frac{1}{N} \sum_{n=1}^{N} \nabla_\theta F(\mathbf{x}_n; \theta) - \frac{1}{S} \sum_{s=1}^{S} \nabla_\theta F(\hat{\mathbf{x}}_s; \theta). \tag{12}$$

# Learning RBMs

- A different approach: *contrastive divergence*
- Approximates the expectation under $p(\mathbf{x}|\theta)$ by a sum over samples $\tilde{\mathbf{x}}_n$ drawn from a distribution obtained by applying $K$ steps of the block Gibbs sampling procedure:

$$\nabla_\theta \ell(\theta) \approx -\frac{1}{N} \sum_{n=1}^{N} \big(\nabla_\theta F(\mathbf{x}_n; \theta) - \nabla_\theta F(\tilde{\mathbf{x}}_n; \theta)\big). \tag{13}$$

# Learning RBMs

- The original CD used K steps of the Gibbs chain, starting and is restarted after every parameter update.
- An alternative approach, Persistent Contrastive Divergence (PCD) does not restart the chain after each update typically resulting in a slower convergence rate but eventually better performance

# Higher-Order Relationships

- The energy function allows the modeling of higher-order dependencies among variables.
- For instance: Third-order multiplicative interactions by introducing two kinds of hidden variables
  - **Subspace Units:** Reflect feature variations, robust to invariances:
  - **Gate Units:** Activate subspace units, pool subspace features.

# Higher-Order Relationships

## Random Variables for SubspaceRBM

- Observables: $\mathbf{x} \in \{0, 1\}^D$.
- Gate Units: $\mathbf{h} \in \{0, 1\}^M$.
- Subspace Units: $\mathbf{S} \in \{0, 1\}^{M \times K}$.
- **Connections:** $x_i$, $h_j$, and $s_{jk}$.

# Higher-Order Relationships

## Energy Function for SubspaceRBM

$$E(\mathbf{x}, \mathbf{h}, \mathbf{S}; \theta) = -\sum_{i=1}^{D} \sum_{j=1}^{M} \sum_{k=1}^{K} W_{ijk} x_i h_j s_{jk} - \sum_{i=1}^{D} b_i x_i - \sum_{j=1}^{M} c_j h_j - \sum_{j=1}^{M} h_j \sum_{k=1}^{K} D_{jk} s_{jk}, \tag{14}$$

$$\theta = \{W, \mathbf{b}, \mathbf{c}, \mathbf{D}\}, \quad W \in \mathbb{R}^{D \times M \times K}, \quad \mathbf{b} \in \mathbb{R}^{D}, \quad \mathbf{c} \in \mathbb{R}^{M}, \quad \mathbf{D} \in \mathbb{R}^{M \times K}.$$

# Higher-Order Relationships

## Conditional Distributions in SubspaceRBM

$$p(x_i = 1|\mathbf{h}, \mathbf{S}) = \mathrm{sigm}\left(\sum_j \sum_k W_{ijk} h_j s_{jk} + b_i\right) \tag{15}$$

$$p(s_{jk} = 1|\mathbf{x}, h_j) = \mathrm{sigm}\left(\sum_i W_{ijk} x_i h_j + h_j D_{jk}\right) \tag{16}$$

$$p(h_j = 1|\mathbf{x}) = \mathrm{sigm}\left(-K \log 2 + c_j + \sum_{k=1}^{K} \mathrm{softplus}\left(\sum_i W_{ijk} x_i + D_{jk}\right)\right) \tag{17}$$

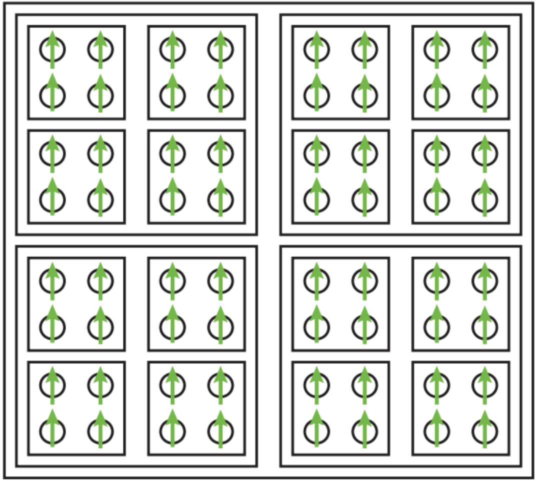
# Renormalization Group

# Introduction to Renormalization Group (RG)

**The Renormalization Group (RG)** is a mathematical framework used to study lengths or energy scales, particularly in statistical physics and quantum field theory.

$$e^{-H_{\mathsf{RG},\theta}[\{h_j\}]} \equiv \sum_{v_i} e^{T_\theta(\{v_i\},\{h_j\})-H(\{v_i\})} \tag{18}$$

# Introduction to Renormalization Group (RG)



# RG in the 1D Ising Model

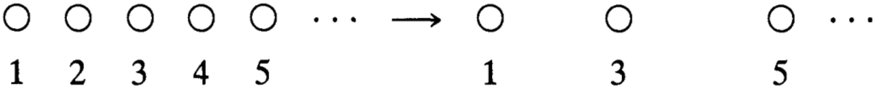
The RG approach can be illustrated with the one-dimensional Ising model. Without a magnetic field, the partition function $Q$ is given by ($K = J/(k_B T)$):

$$Z(K, N) = \sum_{s_1, s_2, \ldots, s_N = \pm 1} \exp\Big[K(s_1 s_2 + s_2 s_3 + s_3 s_4 + \ldots)\Big], \qquad (19)$$

**Decimation:** Summing over even-numbered spins reduces the problem size.

$$Z(K', N) = \sum_{s_1, s_3, \ldots} \Big[\exp(K' s_1 s_3) + \exp(-K' s_1 s_3)\Big] \cdots . \qquad (20)$$

# RG in the 1D Ising Model



# Hidden Units in RBMs

**Restricted Boltzmann Machines (RBMs)** consist of visible ($v$) and hidden ($h$) units:

$$P(v,h) = \frac{1}{Z}\exp\Big(\sum_i a_i v_i + \sum_j b_j h_j + \sum_{i,j} v_i W_{ij} h_j\Big). \tag{21}$$

Hidden units simplify data by marginalizing over $h$:

$$P(v) = \sum_h P(v,h). \tag{22}$$

RBM Hamiltonian for the hidden units:

$$p_\theta(\{h_j\}) \equiv \frac{e^{-H_{\text{RBM},\theta}[\{h_j\}]}}{Z}. \tag{23}$$

hidden units condition over visible units:

$$T(\{v_i\},\{h_j\}) = -E(\{v_i\},\{h_j\}) + H[\{v_i\}]. \tag{24}$$

# Mapping RG to Hidden Units in RBMs

We divide both sides of Eq. 18 by Z to get:

$$\frac{e^{-H_{\text{RG},\theta}[\{h_j\}]}}{Z} = \frac{\sum_{v_i} e^{T_\theta(\{v_i\},\{h_j\})-H(\{v_i\})}}{Z} \tag{25}$$

Substituting Eq. 24 into this equation yields

$$\frac{e^{-H_{\text{RG},\theta}[\{h_j\}]}}{Z} = \frac{\sum_{v_i} e^{-E(\{v_i\},\{h_j\})}}{Z} = p_\theta(\{h_j\}) \tag{26}$$

Substituting Eq. 23 into the right-hand side yields the desired result

$$H_{\text{RG},\theta}[\{h_j\}] = H_{\text{RBM},\theta}[\{h_j\}] \tag{27}$$

**Analogy:**

- RG reduces degrees of freedom (e.g., decimation of spins).
- Hidden units in RBMs marginalize over latent variables.

# Deep Architectures and RG

$$e^{T(\{v_i\},\{h_j\})} = e^{-E(\{v_i\},\{h_j\})+H[\{v_i\}]}$$

$$= \frac{p_\theta(\{v_i\},\{h_j\})}{p_\theta(\{v_i\})} e^{H[\{v_i\}]-H_{\mathsf{RBM},\theta}[\{v_i\}]}$$

$$= p_\theta(\{h_j\}|\{v_i\}) e^{H[\{v_i\}]-H_{\mathsf{RBM},\theta}[\{v_i\}]}$$

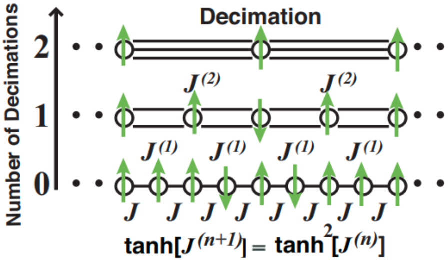
only when RG satisfies

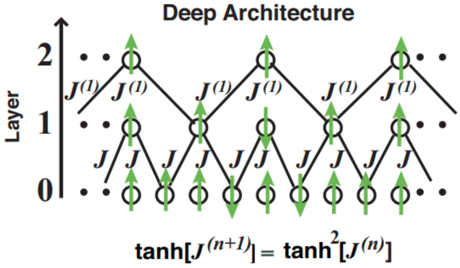
$$\sum_{h_i} e^{T_\theta(\{v_i\},\{h_j\})} = 1$$

we can obtain

$$H[\{v_i\}] = H_{\mathsf{RBM},\theta}[\{v_i\}]$$

# Summery



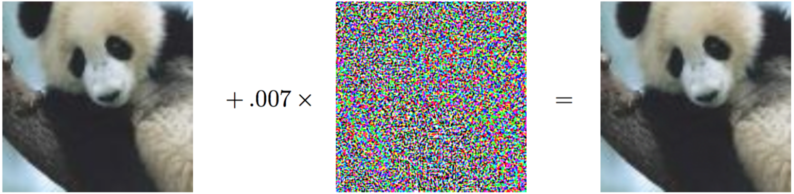




Deep architectures in neural networks are analogous to RG transformations:

- Layers correspond to successive RG steps.
- Hidden units represent coarse-grained variables.

# Adversarial Examples

# Adversarial Examples




$\boldsymbol{x}$
"panda"
57.7% confidence

$\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"nematode"
8.2% confidence

$\boldsymbol{x} + \epsilon\text{sign}(\nabla_{\boldsymbol{x}} J(\boldsymbol{\theta}, \boldsymbol{x}, y))$
"gibbon"
99.3 % confidence

# Adversarial Examples

- We are to perturb all images such that they get misclassified.
- Both previously seen and unseen data are vulnerable.
- Defensive methods based on gradient obfuscation don't work.
- Over the years more advanced attacks have been introduced.
- The general idea is to generate adversarial examples using:

$$x_{adv} = x + \delta, \quad \delta = \arg\max L(x + \delta, y),\ \|\delta\| \leq \epsilon$$

# Adversarial Training

- While most other defenses fail, adversarial training works (kind of):

$$\min_{\theta} \mathbb{E}_{(x,y)\in\mathcal{D}} \left[ \max_{\delta\in\mathcal{S}} L(\theta, x+\delta, y) \right]$$

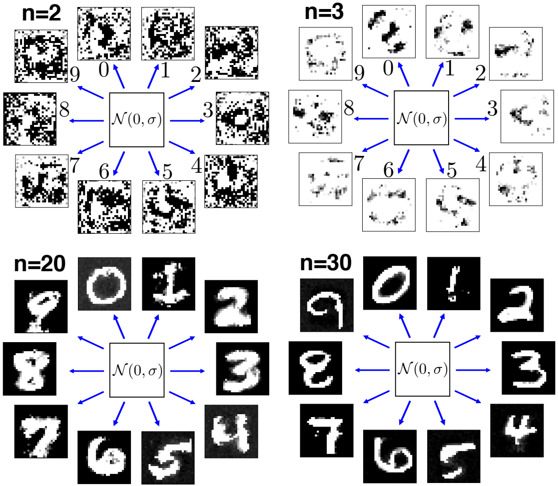
- In AT we are faced with a minmax optimization problem.
- Sadly, AT reduces clean accuracy and is very time consuming.
- More recent works address these issues.

# Restricted Hopfield Networks are Robust to Adversarial Attack

# Dense Associative Memory

- DAM was proposed to improve the storage limitation of the Hopfield Neural Network.
- DAM uses super-linear memory storage capacity as a function of the number of feature neurons.
- Using a gradient decent in the pixel space, a set of rubbish images is constructed that correspond to the minima of the objective function used in training.
- As the power of the interaction vertex is increased the images gradually become less speckled and more semantically meaningful.

# Image Generation



# Adversarial Examples