# Associative Memories (II)
# Stochastic Networks and Boltzmann Machines

Davide Bacciu

Dipartimento di Informatica
Università di Pisa
bacciu@di.unipi.it

Applied Brain Science - Computational Neuroscience (CNS)

# Deterministic Networks

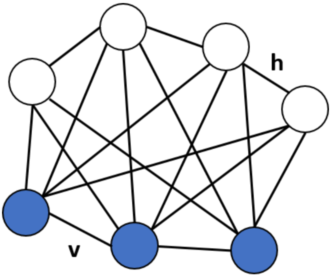
Consider the Hopfield network model introduced so far

- Neuron output is a deterministic function of its inputs
- Recurrent network of all visible units
- Learns to encode a set of fixed training patterns $\mathbf{V} = [\mathbf{v^1} \dots \mathbf{v^P}]$

What models do we obtain if we relax the deterministic input-output mapping?

# Stochastic Networks

- A network of units whose activation is determined by a stochastic function
  - The state of a unit at a given timestep is sampled from a given probability distribution
  - The network learns a probability distribution $P(\mathbf{V})$ from the training patterns




- Network includes both visible **v** and hidden **h** units
- Network activity is a sample from posterior probability given inputs (visible data)

# Neural Sampling Hypothesis

The activity of each neuron can be sampled from the network distribution

- No distinction between input and output
- Natural way to deal with incomplete stimuli
- Stochastic nature of activation is coherent with neural response variability
- Spontaneous neural activity can be explained in terms of prior/marginal distributions

# Probability and Statistics Refresher

On the blackboard

# Stochastic Binary Neurons

- Spiking point neuron with binary output $s_j$
- Typically discrete time model with time into small $\Delta t$ intervals
- At each time interval($t + 1 \equiv t + \Delta t$), the neuron can emit a spike with probability $p_j^{(t)}$

$$s_j^{(t)} = \begin{cases} 1, & \text{with probability } p_j^{(t)} \\ 0, & \text{with probability } 1 - p_j^{(t)} \end{cases}$$

The key is in the definition of the spiking probability (need to be a function of local potential)

$$p_j^{(t)} \approx \sigma(x_j^{(t)})$$

# General Sigmoidal Stochastic Binary Network

Network of $N$ neurons with binary activation $s_j$

- Weight matrix $\mathbf{M} = [M_i j]_{i,j \in \{1,\ldots,N\}}$
- Bias vector $\mathbf{b} = [b_j]_{j \in \{1,\ldots,N\}}$

Local neuron potential $x_j$ defined as usual

$$x_j^{(t+1)} = \sum_{i=1}^{N} M_{ij} s_i^{(t)} + b_j$$

A chosen neuron fires with spiking probability

$$p_j^{(t+1)} = P(s_j^{(t+1)} = 1 | \mathbf{s}^t) = \sigma(x_j^{(t+1)}) = \frac{1}{1 + e^{-x_j^{(t+1)}}}$$

Formulation highlights Markovian dynamics

# Neurobiological Foundations

- Variability in transmitter release in synaptic vescicles $\approx$ Gaussian Distribution (Katz 1954)
- Assuming independent distributions and large number of synaptic connections

  $\Rightarrow$ Central limit theorem

  $\Rightarrow$ Local (membrane) potential $\approx$ Gaussian Distribution
- Conditional probability of neuron spiking is a Gaussian CDF $\approx$ scaled sigmoid

# Network Dynamics (I)

How does the network state (activation of all neurons) evolve in time?

Assume neurons to be updated in parallel every $\Delta t$ (Parallel dynamics)

$$P(\mathbf{s}^{(t+1)}|\mathbf{s}^{(t)}) = \prod_{j=1}^{N} P(s_j^{(t+1)}|\mathbf{s}^{t}) = T(\mathbf{s}^{(t+1)}|\mathbf{s}^{(t)})$$

Yielding a Markov process for state update

$$P(\mathbf{s}^{(t+1)} = \mathbf{s}') = \sum_{\mathbf{s}} T(\mathbf{s}'|\mathbf{s})P(\mathbf{s}^{(t)} = \mathbf{s})$$

# Network Dynamics (II)

- Parallel dynamics assumes a synchronization clock exist (biologically non plausible)
- Alternatively, one neuron at random can be chosen for update at each step (Glauber Dynamics)
- No fixed-point guarantees for **s** but it has a stationary distribution for the network at equilibrium state when its connectivity is symmetric

Given $F_j$ as state flip operator for $j$-th neuron $\mathbf{s}^{(t+1)} = F_j\mathbf{s}^{(t)}$

$$T(\mathbf{s}^{(t+1)}|\mathbf{s}^{(t)}) = \frac{1}{N}P(s_j^{(t+1)}|\mathbf{s}^{t})$$

While if $\mathbf{s}^{(t+1)} = \mathbf{s}^{(t)}$

$$T(\mathbf{s}^{(t+1)}|\mathbf{s}^{(t)}) = 1 - \frac{1}{N}\sum_j P(s_j^{(t+1)}|\mathbf{s}^{t})$$

# The Boltzmann-Gibbs Distribution

Symmetric connectivity enforces detailed balance condition

$$P(\mathbf{s})T(\mathbf{s}'|\mathbf{s}) = P(\mathbf{s}')T(\mathbf{s}|\mathbf{s}')$$

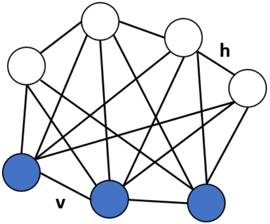
Ensures reversible transitions guaranteeing existence of equilibrium (Boltzmann-Gibbs) distribution

$$P_\infty(\mathbf{s}) = \frac{e^{-E(\mathbf{s})}}{Z}$$

where

- $E(\mathbf{s})$ is the energy function
- $Z = \sum_{\mathbf{s}} e^{-E(\mathbf{s})}$ is the partition function

# Boltzmann Machines




A stochastic recurrent network where binary unit states are also random variables

- visible $\mathbf{v} \in \{0, 1\}$
- latent $\mathbf{h} \in \{0, 1\}$
- $\mathbf{s} = [\mathbf{vh}]$

Boltzmann-Gibbs distribution having linear energy function

$$E(\mathbf{s}) = -\frac{1}{2}\sum_{ij} M_{ij} s_i s_j - \sum_j b_j s_j = -\frac{1}{2}\mathbf{s}^T\mathbf{M}\mathbf{s} - \mathbf{b}^T\mathbf{s}$$

with symmetric and no self-recurrent connectivity

# Learning

**Ackley, Hinton and Sejnowski (1985)**

Boltzmann machines can be trained so that the equilibrium distribution tends towards any arbitrary distribution across binary vectors given samples from that distribution

A couple of simplifications to start with

- Bias $\mathbf{b}$ absorbed into weight matrix $\mathbf{M}$
- Consider only visible units $\mathbf{s} = \mathbf{v}$

Use probabilistic learning techniques to fit the parameters, i.e. maximizing the log-likelihood

$$\mathcal{L}(\mathbf{M}) = \frac{1}{L} \sum_{l=1}^{L} \log P(\mathbf{v}^l | \mathbf{M})$$

given the $P$ visible training patterns $\mathbf{v}^l$

# Gradient Approach

- First, the gradient for a single pattern

$$\frac{\partial P(\mathbf{v}|\mathbf{M})}{\partial M_{ij}} = -\langle v_i v_j \rangle + v_i v_j$$

with free expectations $\langle v_i v_j \rangle = \sum_{\mathbf{v}} P(\mathbf{v}) v_i v_j$

- Then, the log-likelihood gradient

$$\frac{\partial \mathcal{L}}{\partial M_{ij}} = -\langle v_i v_j \rangle + \langle v_i v_j \rangle_c$$

with clamped expectations $\langle v_i v_j \rangle_c = \frac{1}{L} \sum_{l=1}^{p} v_i^l v_j^l$

# Something we have already seen..

It is Hebbian learning again!

$$\underbrace{\langle v_i v_j \rangle_c}_{\text{wake}} - \underbrace{\langle v_i v_j \rangle}_{\text{dream}}$$

- wake part is the usual Hebb rule applied to the empirical distribution of data that the machine *sees* coming in from the outside world
- dream part is an anti-hebbian term concerning correlation between units when generated by the internal dynamics of the machine

Can only capture quadratic correlation!

# Learning with Hidden Units

- To efficiently capture higher-order correlations we need to introduce hidden units $\mathbf{h}$
- Again log-likelihood gradient ascent ($\mathbf{s} = [\mathbf{vh}]$)

$$\frac{\partial P(\mathbf{v}|\mathbf{M})}{\partial M_{ij}} = \sum_{\mathbf{h}} s_i s_j P(\mathbf{h}|\mathbf{v}) - \sum_{\mathbf{s}} s_i s_j P(\mathbf{s})$$

$$= \langle s_i s_j \rangle_c - \langle s_i s_j \rangle$$

- Expectations generally become intractable due to the partition function $Z$ (exponential complexity)

# Boltzmann Mean-field Approximation

- Approximate the expectations using a simpler distribution than $P(\mathbf{s})$

$$Q(\mathbf{s}) = \prod_{i=1}^{N} m_i^{s_i}(1 - m_i)^{s_i}$$

- New parameter $m_i$ describing the probability of unit $i$ being on, that are chosen to minimize

$$\mathbf{m}^* = \arg\min_{\mathbf{m}} KL[Q(\mathbf{s})||P(\mathbf{s})]$$

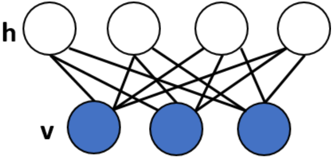
- The solution is obtained by the $N$ dimensional system

$$m_i = \sigma(\sum_j M_{ij} m_j)$$

- The $m_i$'s allow to approximate the intractable expectations

$$\langle s_i s_j \rangle \approx m_i m_j$$

# Restricted Boltzmann Machines (RBM)




A special Boltzmann machine

- Bipartite graph
- Connections only between hidden and visible units

- Energy function, highlighting bipartition in hidden (**h**) and visible (**v**) units

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{v}^T \mathbf{M} \mathbf{h} - \mathbf{b}^T \mathbf{v} - \mathbf{c}^T \mathbf{h}$$

- Learning (and inference) becomes tractable due to graph bipartition which factorizes distribution

# The RBM Catch

Hidden units are conditionally independent given visible units, and viceversa

$$P(h_j|\mathbf{v}) = \sigma(\sum_i M_{ij} v_i + c_j)$$

$$P(v_i|\mathbf{h}) = \sigma(\sum_j M_{ij} h_j + b_i)$$

They can be updated in batch!

# Training Restricted Boltzmann Machines

Again by likelihood maximization, yields

$$\frac{\partial \mathcal{L}}{\partial M_{ij}} = \underbrace{\langle v_i h_j \rangle_c}_{\text{data}} - \underbrace{\langle v_i h_j \rangle}_{\text{model}}$$

A Gibbs sampling approach

Wake

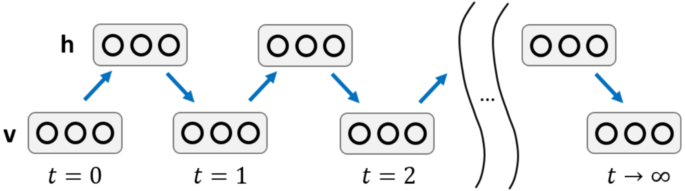
- Clamp data on **v**
- Sample $v_i h_j$ for all pairs of connected units
- Repeat for all elements of dataset

Dream

- Don't clamp units
- Let network reach equilibrium
- Sample $v_i h_j$ for all pairs of connected units
- Repeat many times to get a good estimate

# Gibbs-Sampling RBM

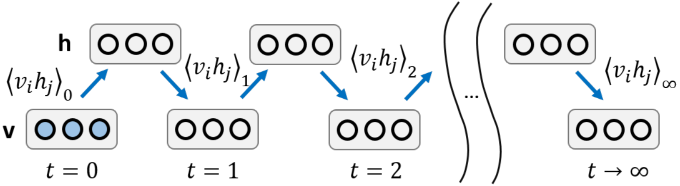
$$\frac{\partial \mathcal{L}}{\partial M_{ij}} = \underbrace{\langle v_i h_j \rangle_c}_{\text{data}} - \underbrace{\langle v_i h_j \rangle}_{\text{model}}$$




It is difficult to obtain an unbiased sample of the second term

# Gibbs-Sampling RBM

## Plugging-in Data

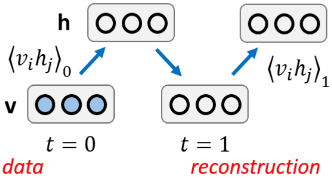



1. Start with a training vector on the visible units
2. Alternate between updating all the hidden units in parallel and updating all the visible units in parallel (iterate)

$$\frac{\partial \mathcal{L}}{\partial M_{ij}} = \underbrace{\langle v_i h_j \rangle_0}_{\text{data}} - \underbrace{\langle v_i h_j \rangle_\infty}_{\text{model}}$$

# Contrastive-Divergence Learning

Gibbs sampling can be painfully slow to converge




1. Clamp a training vector $\mathbf{v}^l$ on visible units
2. Update all hidden units in parallel
3. Update the all visible units in parallel to get a *reconstruction*
4. Update the hidden units again

$$\underbrace{\langle v_i h_j \rangle_0}_{\text{data}} - \underbrace{\langle v_i h_j \rangle_1}_{\text{reconstruction}}$$

# What does Contrastive Divergence Learn?

- A very crude approximation of the gradient of the log-likelihood
  - It does not even follow the gradient closely
- More closely approximating the gradient of a objective function called the Contrastive Divergence
  - It ignores one tricky term in this objective function so it is not even following that gradient
- Sutskever and Tieleman (2010) have shown that it is not following the gradient of any function

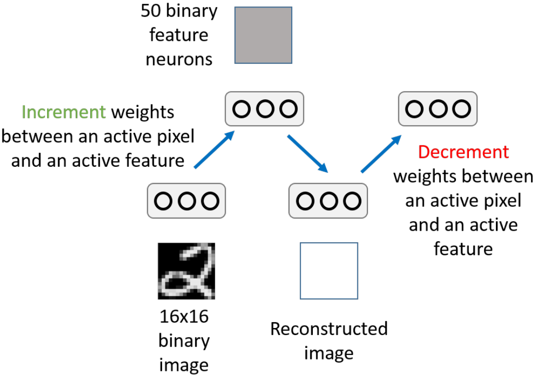
# So Why Using it?

Because He says so!

It works well enough in many significant applications

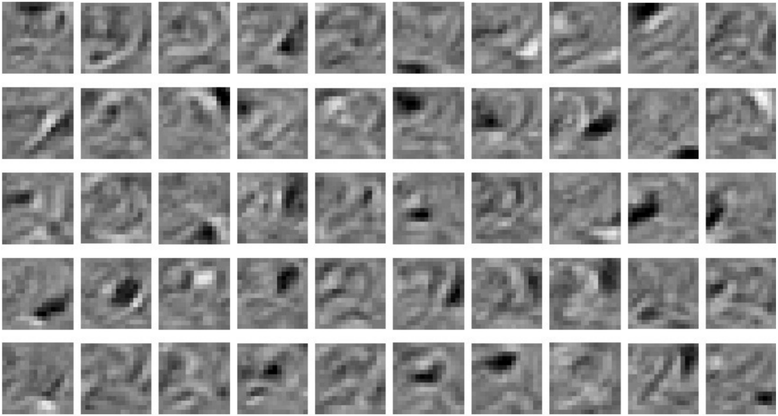
# Character Recognition

Learning good features for reconstructing images of number 2 handwriting

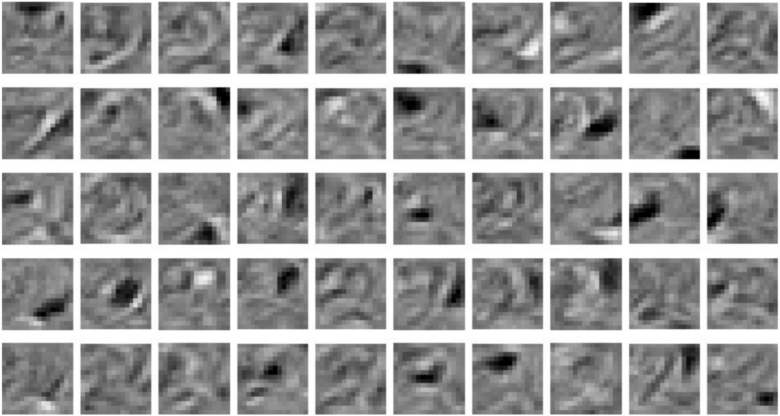



Slide credit goes to G. Hinton

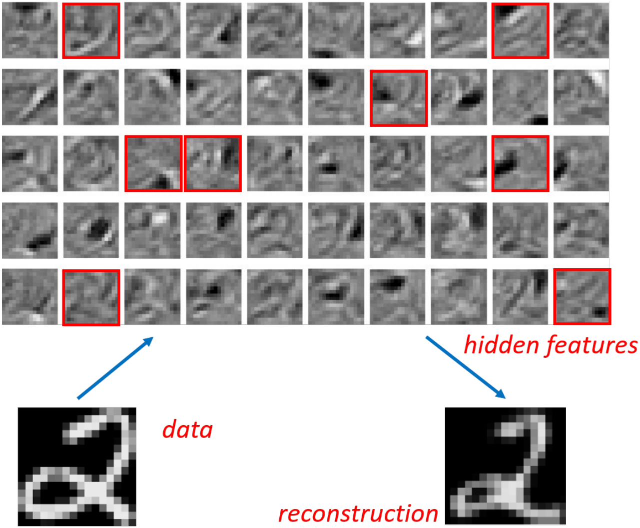
# Weight Learning



# Final Weights

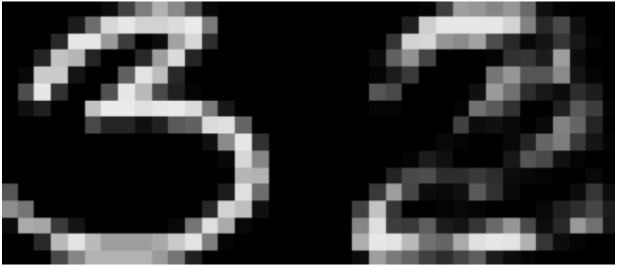


# Digit Reconstruction



# Digit Reconstruction (II)

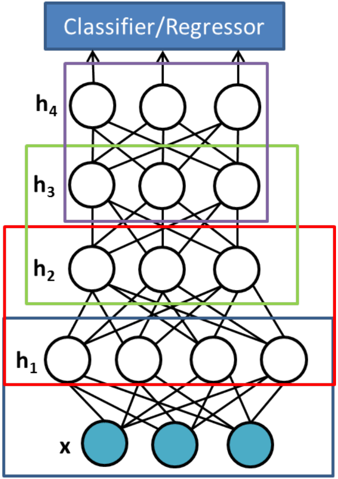
What would happen if we supply the RBM with a test digit that it is not a 2?



It will try anyway to see a 2 in whatever we supply!

# One Last Final Reason for Introducing RBM

## Deep Belief Network




The fundamental building block for one of the most popular deep learning architectures

A network of stacked RBM trained layer-wise by Contrastive Divergence plus a supervised read-out layer

# Take Home Messages

- Stochastic networks as a paradigm that can explain neural variability and spontaneous activity in terms of distributions
- Boltzmann Machines
  - Hidden neurons required to explain high-order correlations
  - Training is a mix of Hebbian and anti-Hebbian
  - Multifaceted nature (recurrent network, undirected graphical model and energy-based network)
- Restricted Boltzmann Machines
  - Tractable model thanks to bipartite connectivity
  - Trained by a very short Gibbs sampling (contrastive divergence)
  - Can be very powerful if stacked (deep learning)

# Next Lecture

- Part I - Lecture (1h)
  - Incremental learning in competitive networks
  - Stability-plasticity dilemma
  - Adaptive Resonance Theory (ART)
- Part II - Lab (2h)
  - Restricted Boltzmann Machines
  - I suggest you have a good look at reference [1] on the course wiki (pages 3 – 6)