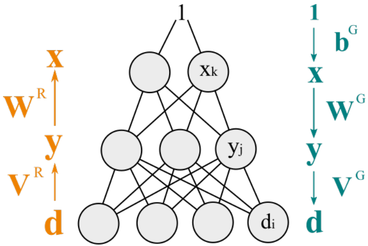
The Helmholtz Machine

André Pacheco
Federal University of Espirito Santo



April 2018

Summary

INTRODUCTION

- The Helmholtz machine (HM) is a **probabilistic model** that is trained to create a generative model of a dataset
- It attempts to build a probability density models of the input data
- Main characteristics:
  - The network is composed by "two binary nets"
  - The training phase also has two parts (wake and sleep phases)
  - The whole learning is carried out by an unsupervised method
  - It is based on the statistical physics and information theory
  - Heavily related with the restricted Boltzmann machine

PROBABILITY

- Let us consider a bit vector $\mathbf{d} \in \{0, 1\}^N$
  - Where N is the nº of bits
  - Ex: $N = 2 \rightarrow \mathbf{d} = \{00, 01, 10, 11\}$
- The HM is all about assigning probabilities to bit vectors like $\mathbf{d}$
  - $p : \{0, 1\}^N \rightarrow [0, 1]$ with $p(\mathbf{d}) \geq 0$ and $\sum_{\mathbf{d}} = 1$
  - $p(\mathbf{d}_1), p(\mathbf{d}_2), \ldots, p(\mathbf{d}_N)$
- Such probability assignment gives the distribution of a discrete random variable $\mathbf{D}$
  - $p(\mathbf{d}) = \text{Prob}\,[\mathbf{D} = \mathbf{d}]$
  - *The probability that the random bit vector* $\mathbf{D}$ *takes on specific value* $\mathbf{d}$

PROBABILITY

- Considering a pair of bit vectors:
  - $p : \{0,1\}^L \times \{0,1\}^M \rightarrow [0,1]$
  - We describe it as $p(\mathbf{x}, \mathbf{y})$
- The joint probability distribution of two random bit vectors $\mathbf{X}$ and $\mathbf{Y}$
  - $p(\mathbf{x}, \mathbf{y}) = \text{Prob}[\mathbf{X} = \mathbf{x} \text{ and } \mathbf{Y} = \mathbf{y}]$
- Several definitions come from this distribution:
  - Marginalization
    - $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}, \mathbf{y})$ and $p(\mathbf{y}) = \sum_{\mathbf{x}} p(\mathbf{x}, \mathbf{y})$
  - Independence
    - $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}).p(\mathbf{y})$
  - Conditional probability
    - $p(\mathbf{x}|\mathbf{y}) = \frac{p(\mathbf{x},\mathbf{y})}{p(\mathbf{y})}$

PROBABILITY

- From the previous definitions:
  - $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x}|\mathbf{y}).p(\mathbf{y})$
  - $p(\mathbf{x}) = \sum_{\mathbf{y}} p(\mathbf{x}|\mathbf{y}).p(\mathbf{y})$
  - $p(\mathbf{x}, \mathbf{y}|\mathbf{d}) = \frac{p(\mathbf{x},\mathbf{y},\mathbf{d})}{p(\mathbf{d})}$
  - $p(\mathbf{y}|\mathbf{x}) = \frac{p(\mathbf{y})}{p(\mathbf{x})}p(\mathbf{x}|\mathbf{y})$ (Bayes Theorem)

INDEPENDENT AND IDENTICALLY DISTRIBUTED RANDOM VARIABLES

- A random bit vector **D** with distribution p(**d**) can be described as a joint distribution over all N bits:
  - $\mathbf{d} : p(\mathbf{d}) = p(d_1, d_2, \ldots, d_N)$
- It is often the case in which that bit is independent and identically distributed (IID) random variable
  - They are mutually independent
    - $p(d_1, d_2, \ldots, d_N) = p(d_1)p(d_2)\ldots p(d_N)$
  - They are identically distributed
    - $p(d_i = 1) = p_0$ and $p(d_i = 0) = 1 - p_0$
- For any specific bit vector **d**:

$$p(\mathbf{d}) = \prod_i p_0^{d_i}(1 - p_0)^{1-d_i} \tag{1}$$

SURPRISE AND ENTROPY

- It is often convenient to recast a probability value p into a quantity called **surprise**

$$s = -\log(\mathbf{d}) \tag{2}$$

  - An event with $p = 1 \Rightarrow$ zero surprise
  - An event with $p = 0 \Rightarrow$ infinity surprise
  - Surprise is never negative

- Let us consider a function $f(\mathbf{D})$, where $\mathbf{D}$ is a random variable
  - The expectation value of this function is given by

$$\langle f(\mathbf{D}) \rangle = \sum_{\mathbf{d}} p(\mathbf{d}) f(\mathbf{d}) \tag{3}$$

SURPRISE AND ENTROPY

- The expected value of the surprise is called **entropy**

$$H(\mathbf{D}) = \langle -\log p(\mathbf{D}) \rangle = -\sum_{\mathbf{d}} p(\mathbf{d}) \log p(\mathbf{d}) \tag{4}$$

  where $\log 0 = 0$

- **Low** entropy $\Rightarrow$ **low** surprise $\Rightarrow$ the system's probabilities tend to be **uniform**

- **High** entropy $\Rightarrow$ **high** surprise $\Rightarrow$ the system's probabilities are **unequal**

- For conditional probabilities we have conditional entropy

$$H(\mathbf{X}|\mathbf{Y}) = -\sum_{\mathbf{d}} p(\mathbf{x}|\mathbf{y}) \log p(\mathbf{x}|\mathbf{y}) \tag{5}$$

KULLBACK-LEIBLER DIVERGENCE

- It is one way to quantify how different two probability distribution are

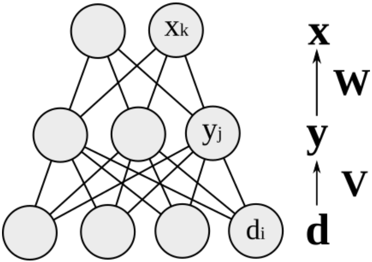
$$KL[p_A(\mathbf{D}), p_B(\mathbf{D})] = \sum_{\mathbf{d}} p_A(\mathbf{d}) \log \frac{p_A(\mathbf{d})}{p_B(\mathbf{d})} \tag{6}$$

- If $p_A(\mathbf{d}) = p_B(\mathbf{d}) \Rightarrow KL = 0$
- KL is never negative and $KL[p_A(\mathbf{D}), p_B(\mathbf{D})] \neq KL[p_B(\mathbf{D}), p_A(\mathbf{D})]$
- Using the logarithm's properties and the expectation value's definition

$$KL[p_A(\mathbf{D}), p_B(\mathbf{D})] = \langle -\log p_B(\mathbf{D}) \rangle_A - \langle -\log p_A(\mathbf{D}) \rangle_A \tag{7}$$

- Thus, the KL from A to B is simply the difference in surprise averaged by A.

LAYERED NEURAL NETWORKS



- Each neuron computes:

$$d_i = \sum_{j=1}^{L} w_{ij} y_j + b_j^d \tag{8}$$

$$y_j = \sum_{k=1}^{L} w_{jk} x_k + b_k^y \tag{9}$$

- To ease the computation we attach the bias into **W** and **V** and appened 1 at the end of **x** and **y**
- Adding a nonlinearity: $\mathbf{y} = \sigma(\mathbf{W}\mathbf{x})$ and $\mathbf{y} = \sigma(\mathbf{V}\mathbf{y})$
  - Where $\sigma(a) = \frac{1}{1+e^{-a}}$

LAYERED NEURAL NETWORKS

- Due to the sigmoid, the neuron output value is in [0,1]
- We can interpret this value as the probability that a binary-value neuron produces the output 1
  - The probability it *"fires"* or *"turn on"*

$$\mathbf{y} = \text{sample}\left[p_y\right], \text{where } p_y = \sigma(\mathbf{W}\mathbf{x}) \tag{10}$$

$$\mathbf{d} = \text{sample}\left[p_d\right], \text{where } p_d = \sigma(\mathbf{V}\mathbf{y}) \tag{11}$$

- Sample $[p]$ is a stochastic function that yields 1 with probability p and 0 with $1 - p$
- As σ never reaches 0 or 1, a neuron will never *fire* with complete certainty
- This layered stochastic network will be the starting point for the Helmholtz machine

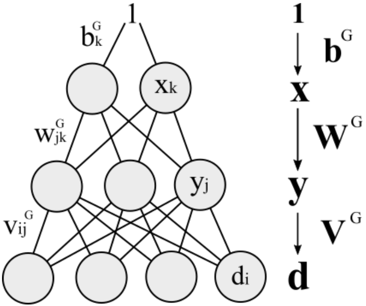
TOP-DOWN PATTERN GENERATION

- The HM sees the world as patterns made of flickering bits
  - Each bit pattern $\mathbf{d}$ appears with $p(\mathbf{d})$
- Since $\mathbf{d}$ can assume a huge number os values, specifying $p\mathbf{d}$ requires a lot of information
  - However, the world is not completly random and the HM attempts to exploit it
- In order to determine the data distribution $p(\mathbf{d})$ we do the following:
  1. From a generative distribution we produce $\mathbf{d}$ using the chain $\mathbf{1} \to \mathbf{x} \to \mathbf{y} \to \mathbf{d}$
  2. This chain will be implemented as a layered neural network

TOP-DOWN PATTERN GENERATION

3. The pattern **x** is stochastically generated from a constant bias
   - It is not regarded to the input network

4. Only the layer **d** is connected to the real data. **x** and **y** are hidden layers



TOP-DOWN PATTERN GENERATION

- A generative distribution G for our chain requires the sprecification of three distributions:
  - $p_G(\mathbf{x})$, $p_G(\mathbf{y}|\mathbf{x})$ and $p_G(\mathbf{d}|\mathbf{y})$
- Writing the chain $\mathbf{x} \rightarrow \mathbf{y} \rightarrow \mathbf{d}$ is a way to say we have conditional independence of $\mathbf{x}$ and $\mathbf{d}$ given $\mathbf{y}$:

$$p_G(\mathbf{x}, \mathbf{d}|\mathbf{y}) = p_G(\mathbf{x}|\mathbf{y}) p_G(\mathbf{d}|\mathbf{y}) \tag{12}$$

- In other words, $\mathbf{x}$ inlfuences $\mathbf{d}$ only through $\mathbf{y}$:

$$p_G = (\mathbf{d}|\mathbf{x}, \mathbf{y}) = p_G(\mathbf{d}|\mathbf{y}) \tag{13}$$

TOP-DOWN PATTERN GENERATION

- From Eq. 12:

$$
\begin{gathered}
p_G(\mathbf{d}|\mathbf{y}) = \frac{p_G(\mathbf{x},\mathbf{d}|\mathbf{y})}{p_G(\mathbf{x}|\mathbf{y})} \\
p_G(\mathbf{x},\mathbf{y},\mathbf{d}) = p_G(\mathbf{y},\mathbf{d}|\mathbf{x})p_G(\mathbf{x})
\end{gathered} \tag{14}
$$

- From Eq. 14 we can get the 3 generation distributions required:

$$
\begin{gathered}
p_G(\mathbf{x},\mathbf{y},\mathbf{d}) = p_G(\mathbf{y},\mathbf{d}|\mathbf{x})p_G(\mathbf{x}) \\
p_G(\mathbf{x},\mathbf{y},\mathbf{d}) = [p_G(\mathbf{d}|\mathbf{x},\mathbf{y})p_G(\mathbf{y}|\mathbf{x})]p_G(\mathbf{x}) \\
\boxed{p_G(\mathbf{x},\mathbf{y},\mathbf{d}) = p_G(\mathbf{d}|\mathbf{y})p_G(\mathbf{y}|\mathbf{x})p_G(\mathbf{x})}
\end{gathered} \tag{15}
$$

TOP-DOWN PATTERN GENERATION

- Moreover, we can get them only from the joint distribution:

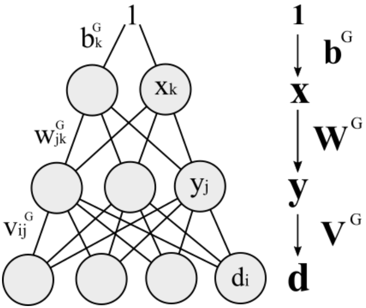
$$p_G(\mathbf{x}) = \sum_{\mathbf{y},\mathbf{d}} p_G(\mathbf{x}, \mathbf{y}, \mathbf{d})$$

$$p_G(\mathbf{y}|\mathbf{x}) = \frac{p_G(\mathbf{x},\mathbf{y})}{p_G(\mathbf{x})} = \frac{\sum_{\mathbf{d}} p_G(\mathbf{x},\mathbf{y},\mathbf{d})}{\sum_{\mathbf{y},\mathbf{d}} p_G(\mathbf{x},\mathbf{y},\mathbf{d})} \tag{16}$$

$$p_G(\mathbf{d}|\mathbf{y}) = \frac{p_G(\mathbf{d},\mathbf{y})}{p_G(\mathbf{y})} = \frac{\sum_{\mathbf{d}} p_G(\mathbf{x},\mathbf{y},\mathbf{d})}{\sum_{\mathbf{x},\mathbf{d}} p_G(\mathbf{x},\mathbf{y},\mathbf{d})}$$

- Thus, we just speak of $p_G(\mathbf{x}, \mathbf{y}, \mathbf{d})$ as the generative distribution pack
- The ultimate quatity of interest is $p_G(\mathbf{d})$
  - The goal is to make the network's $p_G(\mathbf{d})$ as close as possible to the real data distribution $p(\mathbf{d})$

THE GENERATIVE MODEL AS A NEURAL NETWORK



- The set of connections $\{\mathbf{b}^G, \mathbf{W}^G, \mathbf{V}^G\}$ is a **way to specify** $p_G(\mathbf{x}, \mathbf{y}, \mathbf{d})$
  - Of course, it is a **constrained** form for a probability distribution
  - The machine attempts to **approximate** it as much as possible to the reality
- We get the directional probabilities using Eq. 1 and the weights:

$$p_G(\mathbf{x}) = \prod_k p_G(x_k)^{x_k}[1 - p_G(x_k)]^{1-x_k}, \text{ where } p_G(x_k) = \sigma(b_k^G) \tag{17}$$

THE GENERATIVE MODEL AS A NEURAL NETWORK

$$p_G(\mathbf{y}|\mathbf{x}) = \prod_k p_G(y_j|x)^{y_j}[1-p_G(y_j|x)]^{1-y_j}, \text{ where: } p_G(y_j|x) = \sigma(\sum_{k=1}^{L} w^G_{jk}x_k) \tag{18}$$

$$p_G(\mathbf{d}|\mathbf{y}) = \prod_k p_G(d_i|y)^{d_i}[1-p_G(d_i|y)]^{1-d_i}, \text{ where: } p_G(d_i|y) = \sigma(\sum_{j=1}^{M} v^G_{ij}y_j) \tag{19}$$

- **x** and **y** are known as the explanation of **d**
- The number of neurons are unrestricted
- There are variants with more layers, different activation functions, lateral connections and so on

ENERGY

- Let us consider the probability of an explanation $\mathbf{x}$ and $\mathbf{y}$ given some fixed piece of generated data $\mathbf{d}$:

$$p_G(\mathbf{x}, \mathbf{y}|\mathbf{d}) = \frac{p_G(\mathbf{x}, \mathbf{y}, \mathbf{d})}{p_G(\mathbf{d})} = \boxed{\frac{p_G(\mathbf{x}, \mathbf{y}, \mathbf{d})}{\sum_{\mathbf{xy}} p_G(\mathbf{x}, \mathbf{y}, \mathbf{d})}} \tag{20}$$

- According to the statistical physics, this is a energy function:

$$E_G(\mathbf{x}, \mathbf{y}|\mathbf{d}) = -\log p_G(\mathbf{x}, \mathbf{y}, \mathbf{d}) \text{ Applying Eq. 20:}$$

$$p_G(\mathbf{x}, \mathbf{y}|\mathbf{d}) = \frac{e^{-E_G(\mathbf{x}, \mathbf{y}|\mathbf{d})}}{\sum_{\mathbf{xy}} e^{-E_G(\mathbf{x}, \mathbf{y}|\mathbf{d})}} \tag{21}$$

ENERGY

- From the generative energy $E_G(\mathbf{x}, \mathbf{y}|\mathbf{d})$ an analogy to statistical physics arises:
  - Suppose the state of a physical system fluctuates among a set of state $\{q_1, q_2, \ldots\}$
  - The system is in **thermal equilibrium** if the probability of finding the system in a state q_i is related to its energy $E(q_i)$
  - This energy behaves according to the **Boltzmann distribution**

$$p(q_i) = \frac{e^{-\frac{E(q_i)}{T}}}{\sum_i e^{-\frac{E(q_i)}{T}}} \tag{22}$$

- Taking $T = 1$, $E_G(\mathbf{x}, \mathbf{y}|\mathbf{d})$ is known as the energy of the explanation **x, y** of the data pattern **d**
- The energy is the **surprise associated** with the ocurrence of a **particular complete state**

FREE ENERGY

- We want to find a generative model, i.e, $\{\mathbf{b}^G, \mathbf{W}^G, \mathbf{V}^G\}$, that makes $p_G(\mathbf{d})$ close to $p(\mathbf{d})$. Therefore:

$$\phi(G) = KL[p(\mathbf{D}), p_G(\mathbf{D})]$$

$$\phi(G) = \sum_{\mathbf{d}} p(\mathbf{d}) \log \frac{p(\mathbf{d})}{p_G(\mathbf{d})} = \sum_{\mathbf{d}} p(\mathbf{d}) \log p(\mathbf{d}) - \boxed{\sum_{\mathbf{d}} p(\mathbf{d}) \log p_G(\mathbf{d})} \tag{23}$$

- It is the expected surprise of the of the data generated by the network weighted by the real-world probability data:

$$\phi_2(G) = \sum_{\mathbf{d}} p(\mathbf{d}) \log p_G(\mathbf{d}) = \langle -\log p_G(\mathbf{D}) \rangle \tag{24}$$

- The surprise will get smaller if the HM learns a model that is close to $p(\mathbf{d})$

FREE ENERGY

- Thus, our optimization problem is to minimize $\phi_2(G)$:

$$\nabla\phi_2(G) = \sum_{\mathbf{d}} p(\mathbf{d}) \boxed{\nabla[\log p_G(\mathbf{d})]} \tag{25}$$

  - We need to focus on the boxed part of Eq. 25

$$\begin{aligned}
-\log p_G(\mathbf{d}) &= -\log p_G(\mathbf{d}) \times 1 \\
-\log p_G(\mathbf{d}) &= -\log p_G(\mathbf{d})[\textstyle\sum_{\mathbf{x},\mathbf{y}} p_G(\mathbf{x},\mathbf{y}|\mathbf{d})] \\
-\log p_G(\mathbf{d}) &= -\textstyle\sum_{\mathbf{x},\mathbf{y}} p_G(\mathbf{x},\mathbf{y}|\mathbf{d}) \log p_G(\mathbf{d}) \\
-\log p_G(\mathbf{d}) &= -\textstyle\sum_{\mathbf{x},\mathbf{y}} p_G(\mathbf{x},\mathbf{y}|\mathbf{d}) \log[\frac{p_G(\mathbf{x},\mathbf{y},\mathbf{d})}{p_G(\mathbf{x},\mathbf{y}|\mathbf{d})}] \\
-\log p_G(\mathbf{d}) &= -\textstyle\sum_{\mathbf{x},\mathbf{y}} p_G(\mathbf{x},\mathbf{y}|\mathbf{d}) \log p_G(\mathbf{x},\mathbf{y},\mathbf{d}) + -\sum_{\mathbf{x},\mathbf{y}} p_G(\mathbf{x},\mathbf{y}|\mathbf{d}) \log p_G(\mathbf{x},\mathbf{y}|\mathbf{d}) \\
-\log p_G(\mathbf{d}) &= -\textstyle\sum_{\mathbf{x},\mathbf{y}} p_G(\mathbf{x},\mathbf{y}|\mathbf{d}) E_G(\mathbf{x},\mathbf{y}|\mathbf{d}) + H_G(\mathbf{X},\mathbf{Y}|\mathbf{d}) \\
-\log p_G(\mathbf{d}) &= \boxed{\langle E_G(\mathbf{X},\mathbf{Y};\mathbf{d})\rangle_G - H_G(\mathbf{X},\mathbf{Y}|\mathbf{d})}
\end{aligned} \tag{26}$$

- In statistical physics, the **Helmholtz free energy** of a system is given by $F = \langle E \rangle - TH$

FREE ENERGY

- Thus, taking $T = 1$:

$$F_G(\mathbf{d}) = -\log p_G(\mathbf{d}) = \langle E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d})\rangle_G - H_G(\mathbf{X}, \mathbf{Y}|\mathbf{d}) \tag{27}$$

- So, minimizing the KL divergence means minimizing the generative free energy $F_G(\mathbf{D})$:

$$\frac{\partial}{\partial b_k^G} F_G(\mathbf{d}) \quad \frac{\partial}{\partial w_{jk}^G} F_G(\mathbf{d}) \quad \frac{\partial}{\partial v_{jk}^G} F_G(\mathbf{d}) \tag{28}$$

- We need to express $F_G(\mathbf{D})$ in terms of the weights, however:
  - It does not work very well
  - The equations are very hard to manipulate

VARIATIONAL FREE ENERGY

- Let us consider another distribution $p_R(\mathbf{x}, \mathbf{y}|\mathbf{d})$, which can be any arbitrary distribution for the moment

- Let us compute:

$$
\begin{gathered}
KL[p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}), p_G(\mathbf{X}, \mathbf{Y}|\mathbf{d})] = \sum_{\mathbf{xy}} p_R(\mathbf{xy}|\mathbf{d}) \log \frac{p_R(\mathbf{x},\mathbf{y}|\mathbf{d})}{p_G(\mathbf{x},\mathbf{y}|\mathbf{d})} \\
KL[p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}), p_G(\mathbf{X}, \mathbf{Y}|\mathbf{d})] = -H_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}) + \langle E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d}) \rangle_R - F_G(\mathbf{d}) \\
\boxed{F_G(\mathbf{d}) = -H_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}) + \langle E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d}) \rangle_R - KL[p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}), p_G(\mathbf{X}, \mathbf{Y}|\mathbf{d})]}
\end{gathered}
\tag{29}
$$

- Now he have a expression for the generative Helmholtz free energy involving a new distribution p_R

- To proceed we need to use a method called *variational method*

VARIATIONAL FREE ENERGY

- Since the KL cannot be negative:

$$\begin{gathered} -H_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}) + \langle E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d})\rangle_R - F_G(\mathbf{d}) \geq 0 \\ F_G(\mathbf{d}) \leq \langle E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d})\rangle_R - H_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}) \end{gathered} \tag{30}$$

- The variational free energy from R to G is defined as:

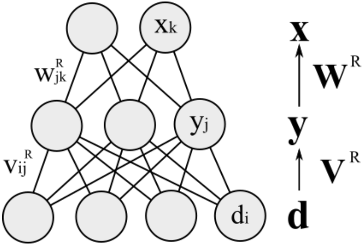
$$\begin{gathered} F_G^R(\mathbf{d}) = \langle E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d})\rangle_R - H_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}) \\ F_G^R(\mathbf{d}) = F_G(\mathbf{d}) + KL[p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}), p_G(\mathbf{X}, \mathbf{Y}|\mathbf{d})] \end{gathered} \tag{31}$$

  - Note that $F_G^G(\mathbf{d}) = F_G(\mathbf{d})$

- Now we need to provide a method to determine $p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d})$

BOTTOM-UP PATTERN RECOGNITION

- In order to determine $p_R(\mathbf{x}, \mathbf{y}|\mathbf{d})$, let us assume the chain $\mathbf{d} \rightarrow \mathbf{x} \rightarrow \mathbf{y}$
  - It means we can factor $p_R(\mathbf{x}, \mathbf{y}|\mathbf{d}) = p_R(\mathbf{x}|\mathbf{y})p_R(\mathbf{y}|\mathbf{d})$
- As previously, let us use the HM upward connections to help us with this task



BOTTOM-UP PATTERN RECOGNITION

- The pattern **d** is the network input, and there is no bias weight coming into it
- Thus, unlike the generative case, we have only two equations:

$$p_R(\mathbf{x}|\mathbf{y}) = \prod_k p_R(x_k|y)^{x_k}[1 - p_R(x_k|y)]^{1-x_k}, \text{ where: } p_R(x_k|y) = \sigma(\sum_{j=1}^{M} w_{kj}^R y_j) \tag{32}$$

$$p_R(\mathbf{y}|\mathbf{d}) = \prod_j p_R(y_j|d)^{y_j}[1 - p_R(y_j|d)]^{1-y_j}, \text{ where: } p_R(y_j|d) = \sigma(\sum_{i=1}^{N} v_{ji}^R d_i) \tag{33}$$

LEARNING

- The HM learning algorithm is based on gradient descent and it will involve two phases:
  1. **Wake-phase:** it involves the generative weights

$$F_G^R(\mathbf{d}) = F_G(\mathbf{d}) + KL[p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d}), p_G(\mathbf{X}, \mathbf{Y}|\mathbf{d})] \tag{34}$$

  2. **Sleep-phase:** it involves the recognition weights

$$\tilde{F}_G^R(\mathbf{d}) = F_G(\mathbf{d}) + KL[p_G(\mathbf{X}, \mathbf{Y}|\mathbf{d}), p_R(\mathbf{X}, \mathbf{Y}|\mathbf{d})] \tag{35}$$

- Both phase use the variational free energy to compute the gradients

LEARNING: WAKE-PHASE

- In this pahse we need to work with $F_G^R(\mathbf{d})$:

$$\begin{aligned} F_G^R(\mathbf{d}) &= \langle E_G(\mathbf{X},\mathbf{Y};\mathbf{d})\rangle_R - H_R(\mathbf{X},\mathbf{Y}|\mathbf{d}) \\ F_G^R(\mathbf{d}) &= \boxed{\sum_{\mathbf{xy}} p_R(\mathbf{x},\mathbf{y}|\mathbf{d})E_G(\mathbf{x},\mathbf{y};\mathbf{d})} - H_R(\mathbf{X},\mathbf{Y}|\mathbf{d}) \end{aligned} \tag{36}$$

- Let us take the derivatives $\frac{\partial}{\partial b^G}$, $\frac{\partial}{\partial w^G_{jk}}$ and $\frac{\partial}{\partial v^G_{ij}}$ as ∇_G:

$$\begin{aligned} \nabla_G F_G^R(\mathbf{d}) &= \nabla_G \textstyle\sum_{\mathbf{xy}} p_R(\mathbf{x},\mathbf{y}|\mathbf{d})E_G(\mathbf{x},\mathbf{y};\mathbf{d}) \\ \nabla_G F_G^R(\mathbf{d}) &= \textstyle\sum_{\mathbf{xy}} p_R(\mathbf{x},\mathbf{y}|\mathbf{d})\nabla_G E_G(\mathbf{x},\mathbf{y};\mathbf{d}) \\ \nabla_G F_G^R(\mathbf{d}) &= \langle \nabla_G E_G(\mathbf{X},\mathbf{Y};\mathbf{d})\rangle_R \end{aligned} \tag{37}$$

- Thus, the algorithm will sample a pattern **d** from the recognition phase then updates the weights throught a small gradient step

LEARNING: WAKE-PHASE

- Now we need to evaluate $\nabla_G E_G$:

$$
\begin{aligned}
\nabla_G \nabla_G E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d}) &= -\nabla_G \log p_G(\mathbf{x}, \mathbf{y}, \mathbf{d}) \\
\nabla_G \nabla_G E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d}) &= -\nabla_G \log p_G(\mathbf{x}) p_G(\mathbf{y}|\mathbf{x}) p_G(\mathbf{d}|\mathbf{y}) \\
\nabla_G \nabla_G E_G(\mathbf{X}, \mathbf{Y}; \mathbf{d}) &= \boxed{-\nabla_G \log p_G(\mathbf{x}) - \nabla_G \log p_G(\mathbf{y}|\mathbf{x}) - \nabla_G \log p_G(\mathbf{d}|\mathbf{y})}
\end{aligned}
\tag{38}
$$

- According to our model:

$$
\begin{aligned}
\log p_G(\mathbf{x}) &= \log \textstyle\prod_k \xi_k^{x_k} (1-\xi_k)^{1-x_k}, \text{ where } \xi_k = \sigma(b_k^G) \\
\log p_G(\mathbf{x}) &= \textstyle\sum_k x_k \log \xi_k + \sum_k (1-x_k) \log(1-\xi_k) \\
\log p_G(\mathbf{y}|\mathbf{x}) &= \log \textstyle\prod_j \psi_j^{y_j} (1-\psi_j)^{1-y_j}, \text{ where } \psi_j = \sigma(\sum_{k=1}^{L} w_{jk}^G x_k) \\
\log p_G(\mathbf{y}|\mathbf{x}) &= \textstyle\sum_j y_j \log \psi_j + \sum_j (1-y_j) \log(1-\psi_j) \\
\log p_G(\mathbf{d}|\mathbf{y}) &= \log \textstyle\prod_i \delta_i^{d_i} (1-\delta_i)^{1-d_i}, \text{ where } \delta_i = \sigma(\sum_{j=1}^{M} v_{jk}^G y_j) \\
\log p_G(\mathbf{d}|\mathbf{y}) &= \textstyle\sum_i d_i \log \delta_i + \sum_i (1-d_i) \log(1-\delta_i)
\end{aligned}
\tag{39}
$$

LEARNING: WAKE-PHASE

- Computing the derivatives (hint: $a = \sigma(b) \Rightarrow \frac{\mathrm{d}a}{\mathrm{d}b} = a(1-a)$)

$$
\begin{aligned}
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \frac{\partial}{\partial b_k^G} \sum_k x_k \log \xi_k + \frac{\partial}{\partial b_k^G} \sum_k (1-x_k) \log(1-\xi_k) \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \frac{\partial}{\partial b_k^G} [x_k \log \xi_k] + \frac{\partial}{\partial b_k^G} [(1-x_k) \log(1-\xi_k)] \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= x_k \frac{\partial}{\partial b_k^G} \log \xi_k + (1-x_k) \frac{\partial}{\partial b_k^G} \log(1-\xi_k) \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \frac{x_k}{\xi_k} \frac{\partial}{\partial b_k^G} \xi_k + \frac{(1-x_k)}{(1-\xi_k)} \frac{\partial}{\partial b_k^G} (1-\xi_k) \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \frac{x_k}{\xi_k} \frac{\partial}{\partial b_k^G} \xi_k - \frac{(1-x_k)}{(1-\xi_k)} \frac{\partial}{\partial b_k^G} \xi_k \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \frac{x_k}{\xi_k} \frac{\partial}{\partial b_k^G} \xi_k - \frac{(1-x_k)}{(1-\xi_k)} \frac{\partial}{\partial b_k^G} \xi_k \text{ **hint} \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \frac{x_k}{\xi_k} \xi_k (1-\xi_k) - \frac{(1-x_k)}{(1-\xi_k)} \xi_k (1-\xi_k) \\
\frac{\partial \log p_G(\mathbf{x})}{\partial b_k^G} &= \boxed{x_k - \xi_k}
\end{aligned}
\tag{40}
$$

LEARNING: WAKE-PHASE

$$\frac{\partial \log p_G(\mathbf{y}|\mathbf{x})}{\partial b_k^G} = 0 \quad \frac{\partial \log p_G(\mathbf{d}|\mathbf{y})}{\partial b_k^G} = 0 \tag{41}$$

- The remaining derivatives are computed similarity to Eq. 40

$$\frac{\partial \log p_G(\mathbf{y}|\mathbf{x})}{\partial w_{jk}^G} = (y_j - \psi_j)x_k \quad \frac{\partial \log p_G(\mathbf{d}|\mathbf{y})}{\partial w_{jk}^G} = 0 \quad \frac{\partial \log p_G(\mathbf{x})}{\partial w_{jk}^G} = 0 \tag{42}$$

$$\frac{\partial \log p_G(\mathbf{d}|\mathbf{y})}{\partial v_{ij}^G} = (d_i - \delta_i)y_j \quad \frac{\partial \log p_G(\mathbf{d}|\mathbf{y})}{\partial v_{ij}^G} = 0 \quad \frac{\partial \log p_G(\mathbf{x})}{\partial v_{ij}^G} = 0 \tag{43}$$

- In a vectorial form:

$$\begin{aligned} \nabla_{\mathbf{b}^G} E_G(\mathbf{x},\mathbf{y};\mathbf{d}) &= -(\mathbf{x} - \xi) \\ \nabla_{\mathbf{W}^G} E_G(\mathbf{x},\mathbf{y};\mathbf{d}) &= -(\mathbf{y} - \psi)\mathbf{x}^T \\ \nabla_{\mathbf{V}^G} E_G(\mathbf{x},\mathbf{y};\mathbf{d}) &= -(\mathbf{d} - \delta)\mathbf{y}^T \end{aligned} \tag{44}$$

LEARNING: WAKE-PHASE

- **b** comes to the world
- **x** and **y** come from the recognition distribution given **d**
- ξ, ψ and δ come from the generative distribution given **x** and **y**
- For each layer, the update rules are:

$$\begin{aligned} \mathbf{b}^G &+= \alpha(\mathbf{x} - \xi) \\ \mathbf{W}^G &+= \alpha(\mathbf{y} - \psi)\mathbf{x}^T \\ \mathbf{V}^G &+= \alpha(\mathbf{d} - \delta)\mathbf{y}^T \end{aligned} \tag{45}$$

where α is the learning rate

LEARNING: WAKE-PHASE

- Pseudocode for wake-phase:

Algorithm 1: Wake-phase

$\mathbf{d}$ = getSampleFromWorld()

$\mathbf{y} = \text{sample}\,[\sigma(\mathbf{V}^R\mathbf{d}^T)]$

$\mathbf{x} = \text{sample}\,[\sigma(\mathbf{W}^R\mathbf{y}^T)]$

$\xi = \sigma(\mathbf{b}^G)$

$\psi = \sigma(\mathbf{W}^G\mathbf{x}^T)$

$\delta = \sigma(\mathbf{V}^G\mathbf{y}^T)$

$\mathbf{b}^G + = \alpha(\mathbf{x} - \xi)$

$\mathbf{W}^G + = \alpha(\mathbf{y} - \psi)\mathbf{x}^T$

$\mathbf{V}^G + = \alpha(\mathbf{d} - \psi)\mathbf{y}^T$

LEARNING: SLEEP-PHASE

- It is similar to wake-phase, but the derivatives are taken from $\tilde{F}_G^R(\mathbf{d}) = F_G(\mathbf{d}) + KL[p_G(\mathbf{X},\mathbf{Y}|\mathbf{d}), p_R(\mathbf{X},\mathbf{Y}|\mathbf{d})]$

$$\begin{aligned} \nabla_R \tilde{F}_G^R(\mathbf{d}) &= \quad KL[p_G(\mathbf{X},\mathbf{Y}|\mathbf{d}), p_R(\mathbf{X},\mathbf{Y}|\mathbf{d})] \\ \nabla_R \tilde{F}_G^R(\mathbf{d}) &= \qquad \langle \nabla_R \log p_R(\mathbf{X},\mathbf{Y}|\mathbf{d}) \rangle_G \end{aligned} \tag{46}$$

- Thus, we need to drawn our attention to $\nabla_R \log p_R(\mathbf{X},\mathbf{Y}|\mathbf{d})$
- The equations for $p_R(\mathbf{x}|\mathbf{y})$ and $p_R(\mathbf{y}|\mathbf{d})$ are obtained just like Eq. 39

LEARNING: SLEEP-PHASE

- The derivatives for $\log p_R(\mathbf{x}|\mathbf{y})$ and $\log p_R(\mathbf{y}|\mathbf{d})$ are obtained similarly to the wake-phase

$$\frac{\partial \log p_R(\mathbf{x}|\mathbf{y})}{\partial w^R_{kj}} = (x_k - \xi_k)x_j \qquad \frac{\partial \log p_R(\mathbf{y}|\mathbf{d})}{\partial w^R_{kj}} = 0 \tag{47}$$

$$\frac{\partial \log p_R(\mathbf{x}|\mathbf{y})}{\partial v^R_{ij}} = 0 \qquad \frac{\partial \log p_R(\mathbf{y}|\mathbf{d})}{\partial v^R_{ij}} = (y_j - \psi_k)d_i \tag{48}$$

- In a vectorial form:

$$\begin{aligned} \nabla_{\mathbf{W}^R} \log p_R(\mathbf{x}|\mathbf{y}) &= (\mathbf{x} - \xi)\mathbf{y}^T \\ \nabla_{\mathbf{V}^R} \log p_R(\mathbf{y}|\mathbf{d}) &= (\mathbf{y} - \psi)\mathbf{d}^T \end{aligned} \tag{49}$$

LEARNING: SLEEP-PHASE

- For each layer, the update rules are:

$$\begin{aligned} \mathbf{W}^R &+= \alpha(\mathbf{x} - \xi)\mathbf{y}^T \\ \mathbf{R}^G &+= \alpha(\mathbf{y} - \psi)\mathbf{d}^T \end{aligned} \tag{50}$$

where α is the learning rate

  - $\mathbf{x}$ and $\mathbf{y}$ come from the generative distribution
  - ξ and ψ come from the recognition distribution given $\mathbf{x}$ and $\mathbf{y}$

LEARNING: SLEEP-PHASE

- Pseudocode for sleep-phase:

Algorithm 2: Wake-phase

$\mathbf{x} = \text{sample}[\sigma(\mathbf{b}^G)]$

$\mathbf{y} = \text{sample}\ [\sigma(\mathbf{W}^G \mathbf{x}^T)]$

$\mathbf{d} = \text{sample}\ [\sigma(\mathbf{V}^G \mathbf{y}^T)]$

$\psi = \sigma(\mathbf{V}^R \mathbf{d}^T)$

$\xi = \sigma(\mathbf{W}^R \mathbf{y}^T)$

$\mathbf{V}^R + = \alpha(\mathbf{y} - \psi)\mathbf{d}^T$

$\mathbf{W}^R + = \alpha(\mathbf{x} - \xi)\mathbf{y}^T$

LEARNING: THE WHOLE ALGORITHM

- All weights are started at 0
  - As $\sigma(0) = 0.5$, every neuron has 50-50% change to fire

Algorithm 3: The whole algorithm

$\mathbf{W}^G, \mathbf{V}^G, \mathbf{b}^G = 0$

$\mathbf{W}^R, \mathbf{V}^R$

repeat

  wake-phase to change $\mathbf{W}^G, \mathbf{V}^G, \mathbf{b}^G$

  sleep-phase to change $\mathbf{V}^R, \mathbf{W}^R$

until *Stopping criteria*;

INSIGTHS

- The restricted Boltzmann machine shares some core concepts:
  - It is a generative model
  - Two training phases (positive and negative)
  - Based on energy functions

- Boltzmann machines use symmetrical weights for recognition and reconstruction whereas in Helmholtz machines the weights organization is different

INSIGTHS

- There is a theory that dreams are samples from a generative model that we generate in order to train the model to be better
- It happens with the RBM, HM and GANs
- So, GANs are also related to the wake-sleep procedure

NEXT STEPS

- Is the RBM a better model than HM?
  - What is the relationship between algorithm wake-sleep and contrastive divergence? Which one is better?
- Is it possible to use some concepts from the wake-sleep to improve the CNN training phase?
  - CNN-RBM-ELM
- Is the capsule network related to RBM and HM training phase?

REFERENCES

1. Kirby, Kevin G. ”A Tutorial on Helmholtz Machines”, Department of Computer Science, Northern Kentucky University, 2006
2. Hinton, G.E., P. Dayan, B.J. Frey and R.M. Neal. The wake-sleep algorithm for unsupervised neural networks, Science 268, 1995
3. Dayan, P., G.E. Hinton, R.M. Neal, and R.S. Zemel. The Helmholtz machine. Neural Computation 7, 889–904, 1995