Universidade do Estado do Rio de Janeiro
Centro de Tecnologia e Ciências
Instituto de Matemática e Estatística

Artur Chiaperini Grover

Inteligência Computacional Usando Máquinas de Boltzmann

Rio de Janeiro
2019

Artur Chiaperini Grover

Inteligência Computacional Usando Máquinas de Boltzmann

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Curso, da Universidade do Estado do Rio de Janeiro.

Orientador: Dra. Roseli Suzi Wedemann

Rio de Janeiro

2019

CATALOGAÇÃO NA FONTE
UERJ / REDE SIRIUS / BIBLIOTECA CTC/D

D979 Chiaperini Grover, Artur
Inteligência Computacional Usando Máquinas de Boltzmann / Artur Chiaperini Grover. – Rio de Janeiro, 2019-
32 f.

Orientador: Dra. Roseli Suzi Wedemann
Dissertação (Mestrado) – Universidade do Estado do Rio de Janeiro, Instituto de Matemática e Estatística, Programa de Pós-Graduação em Curso, 2019.

1. máquina de Boltzmann.. 2. máquina restrita de Boltzmann.. 3. divergência de Kullback-Leibler.. I. Dra. Roseli Suzi Wedemann. II. Universidade do Estado do Rio de Janeiro. III. Instituto de Matemática e Estatística. IV. Título

CDU 02:141:005.7

Autorizo, apenas para fins acadêmicos e científicos, a reprodução total ou parcial desta dissertação, desde que citada a fonte.

Assinatura Data

Artur Chiaperini Grover

Inteligência Computacional Usando Máquinas de Boltzmann

Dissertação apresentada, como requisito parcial para obtenção do título de Mestre, ao Programa de Pós-Graduação em Curso, da Universidade do Estado do Rio de Janeiro.

Aprovada em 22 de Novembro de 2019.
Banca Examinadora:

Dra. Roseli Suzi Wedemann (Orientador)
IME/CComp — UERJ

Dra. Patrícia Nunes da Silva
IME/CComp — UERJ

Dra. Zochil González Arenas
IME/CComp — UERJ

Rio de Janeiro
2019

If no one died, it is just another story to be told.
[Daniel Mirolhaum]

RESUMO

CHIAPERINI GROVER, A. C. G. *Inteligência Computacional Usando Máquinas de Boltzmann*. 2019. 32 f. Dissertação (Mestrado em Curso) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

Texto do resumo em português.

Palavras-chave: máquina de Boltzmann. máquina restrita de Boltzmann. divergência de Kullback-Leibler. entropia relativa.

ABSTRACT

CHIAPERINI GROVER, A. C. G. *Computational Intelligence Using Boltzmann Machines*. 2019. 32 f. Dissertação (Mestrado em Curso) – Instituto de Matemática e Estatística, Universidade do Estado do Rio de Janeiro, Rio de Janeiro, 2019.

Abstract in English.

Keywords: Boltzmann machine. restricted Boltzmann machine. Kullback-Leibler divergence. relative entropy.

LIST OF FIGURES

LIST OF ABBREVIATIONS AND ACRONYMS

BM	Boltzmann Machine
RBM	Restricted Boltzmann Machine
SNN	Stochastic Neural Network
ANN	Artificial Neural Network
KL	Kullback-Leibler

CONTENTS

INTRODUCTION

Nowadays, modelling intelligent complex systems uses two main paradigms, commonly referred to as Symbolic and Connectionist, as basic guidelines for achieving your goals of creating intelligent machines and understanding human cognition. These two approaches depart from different positions, each advocating advantages over the other in reproducing intelligent activity. The traditional symbolic approach argues that the algorithmic manipulation of symbolic systems is an appropriate context for modelling cognitive processes. On the other hand, connectionists restrict themselves to brain-inspired architectures and argue that this approach has the potential to overcome the rigidity of symbolic systems by more accurately modeling cognitive tasks that can only be solved, in the best case, approximately. In addition, connectionist paradigm focus on parallel models which have simple and uniform processing elements interconnected (SUN, 2001).

Years of experimentation with both paradigms lead us to the conclusion that the solution lies between these two extremes, and that the approaches must be integrated and unified. In order to establish a proper link between them, much remains to be researched. If in the 1980s the discussion of intelligence was placed at the distinct poles of the symbolists and connectionists, today the connectionists are divided by the reductionist arguments of the structuralists. For this structuralist current, the failure of the symbolists was due to the fact that their models despised brain architecture, and therefore connectionism must continue to explore more deeply the structural aspects of the thinking organ.

The connectionist and structuralist aspects are approached, respectively, through the paradigm of artificial neural networks and realistic models of the brain, within the area called Computational Neuroscience. Through our models, we investigate ancient questions of Artificial Intelligence regarding the understanding of computability aspects of the human mind.

In this project we will continue with the study and implementation of artificial neural network known as Boltzmann Machine and its derivatives models, such as the Restricted Boltzmann Machine, which have been used to solve artificial intelligence problems, in areas such as: computer vision, encoder problem, statistical pattern recognition, speech recognition, and combinatorial optimization (HERTZ; KROGH; PALMER, 1991). Boltzmann machines is a branch of stochastic methods, where randomness is present. (DUDA; HART; STOCK, 2000), it is also used to solve *searching* and *learning* problems (HINTON, 2010).

1 BOLTZMANN MACHINES

In this chapter will expose the theory behind the Boltzmann Machine. A few considerations regarding the notations that will be adopt in this text is required before stepping forward into the main content.

A single random variables are denoted by x and y. A vector of random variables of size n is represented by $\mathbf{x} = (\mathrm{x}_1, \mathrm{x}_2, \ldots, \mathrm{x}_n)$, where each x_i, $i \in \{1, 2, \ldots, n\}$, represents a single variable. Analagously a vector of random variables of size m is represented by $\mathbf{y}$.

In a discrete scenario, the probability $P(\mathrm{x}_i = x_i)$ represents the probability that a random variable x_i assumes value x_i.

The first section of this chapter introduces the Hopfield Network. We believe Hopfield Networks is an important theoretical basis before Boltzmann Machines, as some authors refer to the last as a generalization of the first one. This theory is mainly based on (HERTZ; KROGH; PALMER, 1991). Following, the second section introduces the Boltzmann Machine; most of the theory is also base on (HERTZ; KROGH; PALMER, 1991) and [GOODFELLOW].

We begin this chapter by briefly introducing the Hopfield Network. The reason is because the Boltzmann Machine is a generalization of the Hopfield Network. In Hopfield Networks units are deterministic while, in Boltzmann Machines, units are stochastic.

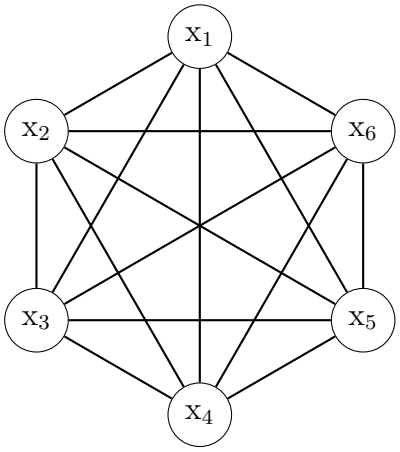
1.1 Hopfield Networks

Hopfield networks are a simple neural network architecture, often referred to as an associative memory network (HERTZ; KROGH; PALMER, 1991). Each unit x_i in a Hopfield network is a binary unit that can assume a value $x_i \in \{0, 1\}$. As Géron (2017) mentioned, this kind of network is first taught a few patterns, and then when exposed to new patterns it will output the closest learned pattern. In the context of image recognition, we consider that each pixel of a binary image maps to one neuron of the network, then the previous statement can be read as the case where the training set contains images of characters, for instance, which are the stored memories of the network; if a new image of a character is presented, the network will recall from the memory the closest character.

In Figure 1, we can see a diagram of the architecture of a Hopfield network. It is a fully connected graph of binary units, which means that each unit is connected to every other unit of the network. It is a different network arrangement compared to perceptron, for instance, where there is no back-coupling (HOPFIELD, 1982).

Hopfield network is an energy-based model, because there is a global energy func-

Figure 1 - Hopfiel Network diagram.



Legend: Each white circle represents a single unit of the Hopfield Network.
Source: Author.

tion associated to the network. This global energy function evolves to a low-energy state during training phase, i.e., the network connections between units are modified so that the energy decreases; the less energy, the better. When connections ω between units, also know as weights or *synaptic strenght*, are symmetrical, i.e., $\omega_{ij} = \omega_{ji}$, where the subscript identify the connection between unit i and j, then Hopfield (1982) presented that the energy function is given by

$$H(\mathbf{x}) = -\frac{1}{2}\sum_{i}\sum_{j}\omega_{ij}x_i x_j - \sum_{i}\phi_i x_i, \tag{1}$$

where $\mathbf{x} = (x_1, x_2, \ldots, x_n)$ is the vector of binary values that each unit of the network has for in a particular state, and n is the number of units in the network. Equation (1) shows that the energy is the sum of many contributions. Each contribution depends on one connection weight ω_{ij} and the binary state in which each neuron is, x_i and x_j, first term of the equation. And the second term only involves the state of each individual unit weighted by bias ϕ_i.

If we want to store a pattern in a Hopfield network, we require the network to learn the right connection between units, so that this pattern can be accessed later on. To compute de proper connection in a network with N units, we can proceed as follows

$$\omega_{ij} = \frac{1}{N}x_i x_j. \tag{2}$$

The quadratic energy function, equation (1), makes it possible for each unit to compute locally how its state affects the global energy, in another words, how each unit affects the global energy when its state is changed. Define the energy gap ΔH_i for a unit

x_i, as the measure of the global energy function difference when the unit x_i has its state changed.

$$\Delta H_i = H(x_i = 0) - H(x_i = 1) = \sum_j \omega_{ij} x_j + \phi_i, \tag{3}$$

the energy gap equation can also be read as the difference between the energy when x_i is off, $x_i = 0$, and the energy when x_i is on, $x_i = 1$. In addition, it can also be computed by differenciating the energy function H, equation (1). [Derivation to be added to appendix]. The Hopfield network will go down hill in this global energy. To find the energy minimum, start from a random state, update each unit one at a time in random order. Update each unit to whichever of its two states gives the lowest global energy.

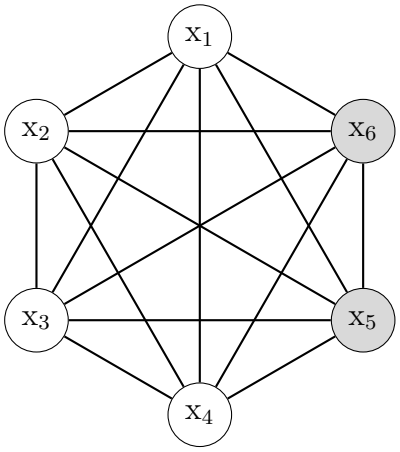
According to Hopfield (1982), memories can be seen as energy minima. Furthermore, by just knowing some parts of an energy minima, its possible to access that memory. An interesting analogy, is that Hopfield networks are like reconstructing a dinosaur from a few bones.

1.2 **Boltzmann Machines**

Boltzmann Machines are a type of stochastic neural networks where the connections between units, which are described by ω, are symmetrical, i.e., $\omega_{ij} = \omega_{ji}$ (HERTZ; KROGH; PALMER, 1991). This kind of stochastic neural networks are capable of learning internal representation and to model an input distribution. Boltzmann Machines were named after the Boltzmann distribution. Due to its stochatics behaviour, the probability of the state of the system to be found in a certain configuration is given by previous mentioned distribution (HERTZ; KROGH; PALMER, 1991). According to (MONTÚFAR, 2018), Boltzmann machines can be seen as an extension of Hopfield networks to include hidden units and units with a stochastic behaviour.

Boltzmann Machines have two kind of units x_i: the visible and hidden units. The visible units are linked to the external world and they correspond to the components of an observation, where data is input. On the other hand, the hidden units do not have any connection outside of the network and they model the dependencies between the components of the observations (FISCHER; IGEL, 2012), i.e., they are responsible for finding the data relation from the input. In Boltzmann machines, there is no connection restriction, this means that every unit, visible or hidden, can be connected to every other unit as in a complete graph, this pattern is not mandatory as some of the connections may not exists depending on the network layout. Regardless of how the connections are, if there is a connection bewteen two units, this connection is symmetric. Training a Boltzmann Machines means finding the right connections ω_{ij} between the units.

Figure 2 - Boltzmann machine diagram.



Legend: Gray circles represent the hidden units of the Boltzmann Machine, while the white circles are the visible units.

Source: Author.

In Figure 2, we have a diagram of a Boltzmann machine layout. We can see that each unit is connected to every other unit in the network regardless of it being a visible or hidden unit.

Stochastics units x_i compose the Boltzmann machine. Stochastics units are random variables x that can assume a binary value with a certain probability. We will consider that a random variable x_i can assume a value of $x_i = 1$ with probability $g(h_i)$, and $x_i = 0$, otherwise, i.e.,

$$x_i = \begin{cases} 1 \text{ with probability } g(h_i) \\ 0 \text{ with probability } 1 - g(h_i) \end{cases}, \tag{4}$$

where the probability $g(h_i)$ is given by

$$g(h_i) = \frac{1}{1 + e^{-2\beta h_i}}, \tag{5}$$

and

$$h_i = \sum_j \omega_{ij} x_j + \phi_i, \tag{6}$$

equation (6) stands for the input a unit x_i receives.

Likewise the Hopfield network, due to the symmetrical connections, there is an energy function which is also given by equation (1). This energy function has minimum

when there is a stable state characterised by

$$x_i = step(h_i). \tag{7}$$

For stochastics neural networks the probability P of finding the system in a given state $\mathbf{x}$ after the equilibrium is reached is

$$P(\mathbf{x}) = \frac{1}{Z} e^{-\beta H(\mathbf{x})}, \tag{8}$$

where

$$Z = \sum_{\mathbf{x}'} e^{-\beta H(\mathbf{x}')} \tag{9}$$

is the partition function. The vector $\mathbf{x}$ represents the state in which the units of the network are, for instance, the following state, in a 3 units stochastic network, $\mathbf{x} = (x_1, x_2, x_3) = (1, 0, 1)$.

Now let us consider the Boltzmann machine case where there are two kinds of units, i.e., the visible and the hidden units. Let us identify the state of the visible units by an index v and the state of the hidden units by an index h. Considering a particular system with N visible units and K hidden units, the whole system have 2^{N+K} possibilities of states in which it can be found.

The joint probability P_{vh} is the probability of finding the visible and hidden units in the states v and h, respectively. This probability distribution is given by the Boltzmann distribution

$$P_{vh} = \frac{1}{Z} e^{-\beta H_{vh}}, \tag{10}$$

where

$$Z = \sum_u \sum_k e^{-\beta H_{uk}}, \tag{11}$$

and

$$H_{vh} = -\frac{1}{2} \sum_i \sum_j \omega_{ij} x_i^{(vh)} x_j^{(vh)} - \sum_i \phi_i x_i^{(vh)}. \tag{12}$$

Equations (10), (11) and (12) are ajustments to equations (8), (9) and (1), respectively, where now the different kind of units state is taken into consideration. In equation (11), the indexes u and k refer to visible and hidden units states, respectively, different indexes are used to avoid bewilderment.

As previously mentioned, the problem a Boltzmann Machine is trying to solve is

determining the connections ω_{ij} between units such that the visible units have a certain probability distribution. In order to do that, we need to find the marginal probability of the state v in which the visible units are found regardless of the state h of the hidden units. The marginal probability P_v is given by

$$P_v = \sum_h P_{vh} = \sum_h \frac{e^{-\beta H_{vh}}}{Z}. \tag{13}$$

Although we know that P_v is a function of the connections ω_{ij}, and that this is the probability of finding the visible units in the state v. We want the states to have a certain probability R_v, i.e., a desired probability. This means that ideally we would like to match the empirical distribution of the data, even though we do not have access to the correct distribution, only to what the observed data has given us as an input to training the model.

One way to evaluate the difference between two probability distribution, for example, P_v and R_v, is using the Kullback-Leibler divergence $D_{KL}(R_v||P_v)$, which can also be referred to as relative entropy, E, which will be our cost function. Further comments about the Kullback-Leibler divergence can be found on appendix A.

$$E = \sum_v R_v \ln\left(\frac{R_v}{P_v}\right). \tag{14}$$

The relative entropy E has the property of always being equal or greater than zero. It reaches zero only if $P_v = R_v$, which means that we are able to retrieve the exactly desired probability distribution at the visible units from the input data. We have to minimise E using the gradient descent (HERTZ; KROGH; PALMER, 1991), relative to the weights ω_{ij} and the bias ϕ_i,

$$\Delta\omega_{ij} = -\eta\frac{\partial E}{\partial\omega_{ij}}, \tag{15}$$

and

$$\Delta\phi_i = -\eta\frac{\partial E}{\partial\phi_i}, \tag{16}$$

where

$$\begin{aligned} E &= \sum_v R_v \ln\left(\frac{R_v}{P_v}\right) \\ &= \sum_v \left[\ln\left(R_v\right) - \ln\left(P_v\right)\right]. \end{aligned} \tag{17}$$

In the following steps, we present the derivation of the gradient descent, based on

the derivation in (HERTZ; KROGH; PALMER, 1991)

$$\begin{aligned}
\Delta\omega_{ij} &= -\eta\frac{\partial E}{\partial\omega_{ij}} \\
&= -\eta\frac{\partial}{\partial\omega_{ij}}\left[\sum_v R_v\left(\ln\left(R_v\right)-\ln\left(P_v\right)\right)\right] \\
&= \eta\frac{\partial}{\partial\omega_{ij}}\left[\sum_v R_v\ln\left(P_v\right)\right] \\
&= \eta\sum_v R_v\frac{\partial}{\partial\omega_{ij}}\left[\ln\left(P_v\right)\right] \\
\Rightarrow\Delta\omega_{ij} &= \eta\sum_v\frac{R_v}{P_v}\frac{\partial P_v}{\partial\omega_{ij}}.
\end{aligned} \tag{18}$$

To continue with the computation of $\Delta\omega_{ij}$, we have to find the derivative of $\partial P_v/\partial\omega_{ij}$, from the marginal probability, equation (13),

$$P_v = \frac{\sum_h e^{-\beta H_{vh}}}{\sum_u\sum_k e^{-\beta H_{uk}}}, \tag{19}$$

thus the derivative of P_v follows

$$\begin{aligned}
\frac{\partial P_v}{\partial\omega_{ij}} &= \frac{\partial}{\partial\omega_{ij}}\left[\frac{\sum_h e^{-\beta H_{vh}}}{\sum_u\sum_k e^{-\beta H_{uk}}}\right] \\
&= \frac{1}{\sum_u\sum_k e^{-\beta H_{uk}}}\sum_h(-\beta)e^{-\beta H_{vh}}\frac{\partial H_{vh}}{\partial\omega_{ij}} \\
&- \sum_h e^{-\beta H_{vh}}\frac{1}{\left(\sum_u\sum_k e^{-\beta H_{uk}}\right)^2}\sum_u\sum_k e^{-\beta H_{uk}}(-\beta)\frac{\partial H_{uk}}{\partial\omega_{ij}}.
\end{aligned} \tag{20}$$

Following equation (20), we need to compute the term $\partial H_{vh}/\partial\omega_{ij}$,

$$\begin{aligned}
\frac{\partial H_{vh}}{\partial\omega_{ij}} &= \frac{\partial}{\partial\omega_{ij}}\left[-\frac{1}{2}\sum_m\sum_n\omega_{mn}x_m^{(vh)}x_n^{(vh)} - \sum_m\phi_{mm}x_m^{(vh)}\right] \\
&= \frac{\partial}{\partial\omega_{ij}}\left[-\frac{1}{2}\sum_{m\neq i,j}\sum_{n\neq i,j}\omega_{mn}x_m^{(vh)}x_n^{(vh)} - \frac{1}{2}\omega_{ij}x_i^{(vh)}x_j^{(vh)} - \frac{1}{2}\omega_{ji}x_j^{(vh)}x_i^{(vh)} - \sum_m\phi_m x_m^{(vh)}\right],
\end{aligned} \tag{21}$$

as the connections between units are symmetric, i.e., $\omega_{ij} = \omega_{ji}$, we can simplify equa-

tion (21), **RRUMAR OS ÍNDICES!!!**

$$\begin{aligned}
\frac{\partial H_{vh}}{\partial \omega_{ij}} &= \frac{\partial}{\partial \omega_{ij}}\left[-\frac{1}{2}\sum_{m\neq i,j}\sum_{n\neq i,j}\omega_{mn}x_m^{(vh)}x_n^{(vh)} - \omega_{ij}x_i^{(vh)}x_j^{(vh)} - \sum_m \phi_m x_m^{(vh)}\right] \\
&= \frac{\partial}{\partial \omega_{ij}}\left[-\frac{1}{2}\sum_{m\neq i,j}\sum_{n\neq i,j}\omega_{mn}x_m^{(vh)}x_n^{(vh)}\right] + \frac{\partial}{\partial \omega_{ij}}\left[-\omega_{ij}x_i^{(vh)}x_j^{(vh)}\right] + \frac{\partial}{\partial \omega_{ij}}\left[-\sum_m \phi_m x_m^{(vh)}\right] \\
\Rightarrow &\frac{\partial H_{vh}}{\partial \omega_{ij}} = -x_i^{(vh)}x_j^{(vh)}.
\end{aligned} \tag{22}$$

Analagous to $\partial H_{vh}/\partial \omega_{ij}$, we have the derivative of H_{uk} which is

$$\frac{\partial H_{uk}}{\partial \omega_{ij}} = -x_i^{(uk)}x_j^{(uk)}. \tag{23}$$

Going back to equation (20), we can replace the derivatives of H, and solve the derivative of the marginal probability P_v,

$$\begin{aligned}
\frac{\partial P_v}{\partial \omega_{ij}} &= \frac{1}{Z}\sum_h e^{-\beta H_{vh}}(-\beta)(-x_i^{(vh)}x_j^{(vh)}) - \frac{1}{Z^2}\sum_h e^{-\beta H_{vh}}\sum_u\sum_k e^{-\beta H_{uk}}(-\beta)(-x_i^{(uk)}x_j^{(uk)}) \\
&= \beta\left[\sum_h \frac{e^{-\beta H_{vh}}}{Z}x_i^{(vh)}x_j^{(vh)} - \sum_h \frac{e^{-\beta H_{vh}}}{Z}\sum_u\sum_k \frac{e^{-\beta H_{uk}}}{Z}x_i^{(uk)}x_j^{(uk)}\right] \\
&= \beta\left[\sum_h P_{vh}x_i^{(vh)}x_j^{(vh)} - P_v\sum_u\sum_k P_{uk}x_i^{(uk)}x_j^{(uk)}\right] \\
\Rightarrow &\frac{\partial P_v}{\partial \omega_{ij}} = \beta\left[\sum_h P_{vh}x_i^{(vh)}x_j^{(vh)} - P_v\langle x_i x_j\rangle\right].
\end{aligned} \tag{24}$$

Given the derivative of P_v in relation to ω_{ij}, we can compute the learning term $\Delta\omega_{ij}$, from equation (18),

$$\begin{aligned}
\Delta\omega_{ij} &= \eta\sum_v \frac{R_v}{P_v}\beta\left[\sum_h P_{vh}x_i^{(vh)}x_j^{(vh)} - P_v\langle x_i x_j\rangle\right] \\
&= \eta\beta\left[\sum_v\sum_h \frac{R_v}{P_v}P_{vh}x_i^{(vh)}x_j^{(vh)} - \sum_v \frac{R_v}{P_v}P_v\langle x_i x_j\rangle\right] \\
&= \eta\beta\left[\sum_v\sum_h R_v\frac{P_{vh}}{P_v}x_i^{(vh)}x_j^{(vh)} - \sum_v R_v\langle x_i x_j\rangle\right] \\
&= \eta\beta\left[\sum_v\sum_h R_v P_{h|v}x_i^{(vh)}x_j^{(vh)} - \langle x_i x_j\rangle\right].
\end{aligned} \tag{25}$$

In the above derivation, we have used the following relations,

$$P_{h|v} = \frac{P_{vh}}{P_v}, \tag{26}$$

which is the conditional probability equation. In our scenario this equation means that the probability distribution of the hidden units in state h given the state v of the visible units is the joint probability distribution of both states, v and h, divided by the marginal probability distribution of the visible units in state v.

The second term in equation (25), is the average of units i and j over all combinations of states v and h of the system. In other words, we would have to compute all possible combination of states of visible and hidden units, v and h, and then average over the specific units i and j.

The first term can be simplified by

$$\sum_v R_v \sum_h P_{h|v} x_i^{(vh)} x_j^{(vh)} = \sum_v R_v \langle x_i x_j \rangle^{(v)} = \langle \langle x_i x_j \rangle^{(v)} \rangle, \tag{27}$$

which is the average value according to the probability distribution R_v of the correlation between variables x_i and x_j when the visibles units are fixed at state v.

Then equation (25) becomes

$$\Delta\omega_{ij} = \eta\beta \left[\langle \langle x_i x_j \rangle^{(v)} \rangle_{clamped} - \langle x_i x_j \rangle_{free} \right], \tag{28}$$

it is important to notice that the subscripts *clamped* means that we have to fix a certain v state on the visible units otherwise the second term in the equation does not have a reference to whom it should be trying to match. On the other hand, the subscript *free* identify the case where the variables are allowed to vary freely without any restriction (DUDA; HART; STOCK, 2000). When the first and the second terms are equal, i.e., $\langle \langle x_i x_j \rangle^{(v)} \rangle_{clamped} = \langle x_i x_j \rangle_{free}$, the update in the weight is zero, $\Delta\omega_{ij} = 0$, thus the desired value of that connection is obtained.

2 EXTRA STUFF

2.1 New Ideas

2.1.1 ANSARI

The energy computed in Boltzmann Machines is a measure of self-consistency of the system. Randomness is fundamental to avoid local optima. Exponential schedule for annealing instead of linear approach, does not have the drawback of premature freezing. Premature freezing means that the system reaches minimum temperature and becomes deterministic before a global minimum has been reached.

Boltzmann machines can be seem as a system where each unit represents a hypothesis and each connection a constrain among hypothesis.

The ability of the system to escape local optimum depends on the path chosen for the temperature, the annealing schedule. The success of simulated annealing depends on the proper choice of the annealing schedule. There is a need to have an annealing schedule where the temperature never gets stuck at a particular value, but varies dynamically in such a way so to increase the system optimality.

2.1.2 SMOLENSKY

$$W_{i\alpha} = (k_\alpha)_i \frac{\sigma_\alpha}{|\mathbf{k}_\alpha|}, \tag{29}$$

where $W_{i\alpha}$ is the weight between atom α and feature i, $\mathbf{k}_\alpha$ is the connection vector of the atom α, $(k_\alpha)_i$ is the connection vector to a unit i, and σ_α is the strenght of the atom α.

The harmony function assigns a real number $H_{\mathbf{K}}(\mathbf{r}, \mathbf{a})$ to each state of the system, which is determined by a pair $(\mathbf{r}, \mathbf{a})$, where $\mathbf{r}$ is the representation vector and $\mathbf{a}$ the activation vector.

$$H_{\mathbf{K}}(\mathbf{r}, \mathbf{a}) = \sum_\alpha \sigma_\alpha a_\alpha h_\kappa(\mathbf{r}, \mathbf{k}_\alpha), \tag{30}$$

where a_α is the connection to atom α, and $h_\kappa(\mathbf{r}, \mathbf{k}_\alpha)$ is the harmony contribution by activating the atom α given the current representation vector $\mathbf{r}$.

$$h_\kappa(\mathbf{r}, \mathbf{k}_\alpha) = \frac{\mathbf{r} \cdot \mathbf{k}_\alpha}{|\mathbf{k}_\alpha|} - \kappa = \frac{\langle \mathbf{r}, \mathbf{k}_\alpha \rangle}{|\mathbf{k}_\alpha|} - \kappa. \tag{31}$$

Putting equation (31) into equation (30),

$$\begin{aligned} H_{\mathbf{K}}(\mathbf{r},\mathbf{a}) &= \sum_{\alpha} \sigma_\alpha a_\alpha \left[\frac{\langle \mathbf{r}, \mathbf{k}_\alpha \rangle}{|\mathbf{k}_\alpha|} - \kappa \right] \\ &= \sum_{\alpha} a_\alpha \left[\frac{\sigma_\alpha \langle \mathbf{r}, \mathbf{k}_\alpha \rangle}{|\mathbf{k}_\alpha|} - \sigma_\alpha \kappa \right]. \end{aligned} \tag{32}$$

Considering the inner product

$$\langle \mathbf{r}, \mathbf{k}_\alpha \rangle = \sum_i r_i (k_\alpha)_i, \tag{33}$$

and replacing equation (33) into equation (32),

$$\begin{aligned} H_{\mathbf{K}}(\mathbf{r},\mathbf{a}) &= \sum_{\alpha} a_\alpha \left[\frac{\sigma_\alpha}{|\mathbf{k}_\alpha|} \sum_i r_i (k_\alpha)_i - \sigma_\alpha \kappa \right] \\ &= \sum_{\alpha} a_\alpha \left[\sum_i r_i (k_\alpha)_i \frac{\sigma_\alpha}{|\mathbf{k}_\alpha|} - \sigma_\alpha \kappa \right]. \end{aligned} \tag{34}$$

Equation (34) we can be reorder thus it becomes a function of the weights $W_{i\alpha}$, equation (29),

$$H_{\mathbf{K}}(\mathbf{r},\mathbf{a}) = \sum_{\alpha} \sum_i r_i W_{i\alpha} a_\alpha - \sum_{\alpha} a_\alpha \sigma_\alpha \kappa, \tag{35}$$

where it is possible to see the harmony function as a function of the representation vector $\mathbf{r}$, the activation vector $\mathbf{a}$ just as in equation (30), but also as a function of the weights $W_{i\alpha}$ and the bias $\gamma_\alpha = \sigma_\alpha \kappa$ which is connected to each atom. Both weights and bias carry the information of the atom strength σ_α, which is related to the frequency of the atom α: the more a particular atom α is activated, the more its frequency, thus increasing the importance of this atom to describe a feature i from the representation vector $\mathbf{r}$.

The maximum-likelihood completion function $c(\iota)$ determined by the probability distribution p is defined by

$$\begin{aligned} c(\iota) = \{ \mathbf{r} \in \mathbf{R} \mid &\text{for some } \mathbf{a} \in \mathbf{A}, \text{and all } (\mathbf{r}', \mathbf{a}') \in \mathbf{R} \times \mathbf{A} \\ &\text{such that } \mathbf{r}' \supset \iota : p(\mathbf{r}, \mathbf{a}) \geqslant p(\mathbf{r}', \mathbf{a}') \}, \end{aligned} \tag{36}$$

where $\mathbf{r}$ is the completion of a point ι.

In information theory, the concept of entropy (missing information) can be interpreted as the average of uncertainty of a variable outcome, similarly in physics, entropy is relate to the randomness of a system or the lack of information about it. Entropy is a

function of the probability distribution of a random variable x:

$$S(P(x)) = -\sum_{x \in X} P(x) \ln (P(x)). \tag{37}$$

Theorem 1: *competence.*

A cognitive system should performe completion according to Boltzmann distribution for statistical extrapolation and inference. Considering an experiment, each pattern can be observed with a particular frequency. Assuming the probability distribution of the experiment R is consistent with the Boltzmann distribution P with harmony function given by equation (35), the maximum-likelihood completions of this distribution are the same as

$$P(\mathbf{r}, \mathbf{a}) \propto e^{H(\mathbf{r},\mathbf{a})}. \tag{38}$$

Theorem 2: *realizability.*

Nodes are updates stochastically, every node within the same layer is updated in parallel, and then every node in the other layer is updated in parallel. During the update process the computational temperature is lowered via simulated annealing.

2.1.3 AGGARWAL

RBM the probabilistic states of the network are learned for a set of inputs. This network models the joint probability distribution os the observed attributes together with some hidden attributes. The connections are undirected because, RBM are designed to learn te probabilistic relationship rather than input-output mapping. RBM are probabilistic models that create latent representations of the underlying data points. It is similar to an autoencoder, but the latent representation is created stochastically instead of deterministically.

Boltzmann machines use probabilistic states to represent Bernoulli distributions of the binary attributes. It contains both visible and hidden states.

Hopfield: given the weights and biases in the network at any particular time, it is possible to find a local energy minimum in terms of the states by repeatedly using the update rule:

$$s_i = \begin{cases} 1 \text{ if } \sum_{j:j \neq i} \omega_{ij} s_j + b_i \geq 0 \\ 0 \text{ otherwise} \end{cases} . \tag{39}$$

For a particular set of parameters (ω_{ij}, b_i), the Boltzmann machine defines a prob-

ability distribution over various state configurations. In Boltzmann machines, the dependence between all pairs of states is undirected: the visible states depend as much on the hidden states as the hidden states depend on visible states. Boltzmann machine iteratively samples the states using a conditional distribution generated from the state values in the previous iteration unil thermal equilibrium is reached.

Learnings weights in the Boltzmann machine requires to maximize the likelihood of the specific training dataset observed. It is also possible to maximize the log-likelihood, by taking the ln of the probability distribution.

$$\ln(P(s)) = -E(s) - \ln(Z) \tag{40}$$

To maximize the log-likelihood, equation (39) is derived in terms of the weights ω_{ij}, then it follows

$$\frac{\partial \ln(P(s))}{\partial \omega_{ij}} = \langle s_i, s_j \rangle_{data} - \langle s_i, s_j \rangle_{model}. \tag{41}$$

Weights' update rule rely on two terms: $\langle s_i, s_j \rangle_{data}$ which represents the correlations between the states of the nodes i and j when the visible vectors are fixed to a vector in the training data, and the hidden states are allowed to vary; the second term, $\langle s_i, s_j \rangle_{model}$ is an average product of s_i and s_j from the model-centric samples obtained from Gibbs sampling.

$$\omega_{ij} \leftarrow \omega_{ij} \alpha(\langle s_i, s_j \rangle_{data} - \langle s_i, s_j \rangle_{model}), \tag{42}$$

The subtraction of the model-centric correlations (second term) is required in order to remove the effect of the partition function within the expression of the log probability. Monte Carlo sampling requires a large amount of samples to reach thermal equilibrium. Restricted Boltzmann machine is a simplification of the Boltzmann machine.

The probability of each hidden unit taking on the value 1 is:

$$P(h_j|\mathbf{v}) = \frac{1}{1 + e^{-b_j^{(h)} - \sum_i v_i \omega_{ij}}}, \tag{43}$$

where energy gap ΔE_j of a hidden unit j, between values $h_j = 0$ and $h_j = 1$ is

$$\Delta E_j = b_j + \sum_i v_i \omega_{ij}, \tag{44}$$

analogously we have the probability of each visible unit.

Equation (43) can also be writen in terms of the sigmoid

$$P(h_j|\mathbf{v}) = \sigma\left(b_j + \sum_i v_i \omega_{ij}\right). \tag{45}$$

For both the hidden and visible units, it is interesting to notice that the hidden unit j depends only on the state of the visible units. On the other hand, the visible unit i depends only on the state of the hidden units. This aspect makes it possible to update all hidden units in parallel and then all the visible units in parallel as well.

The weights encode the affinities between the visible and the hidden states. Large positive weights implies the two states are likely to be on together. Weights are initialized to a small value each. Learning can be performed as follows:

- *Positive phase*: considering a mini-batch of training instances, for each element in this mini-batch, every hidden unit is updated according to equation (43). Once every hidden unit is updated, the average product between each pair of visible and hidden unit is computed, denoted as $\langle v_i, h_j \rangle_{pos}$.

- *Negative phase*: for each training instance, it goes through a phase of Gibbs sampling after starting at with randomly initialized states. Is is achieved by repeatedly using equation (44) and its version for the visible units. $\langle v_i, h_j \rangle_{neg}$ is computed after the values of v_i and h_j are at thermal equilibrium.

- Weights and biases can be updated as follows:

$$\begin{aligned}
\omega_{ij} &\leftarrow \omega_{ij} + \alpha(\langle v_i, h_j \rangle_{pos} - \langle v_i, h_j \rangle_{neg}), \\
b_i^{(v)} &\leftarrow b_i^{(v)} + \alpha(\langle v_i, 1 \rangle_{pos} - \langle v_i, 1 \rangle_{neg}), \\
b_j^{(h)} &\leftarrow b_j^{(h)} + \alpha(\langle 1, h_j \rangle_{pos} - \langle 1, h_j \rangle_{neg}),
\end{aligned} \tag{46}$$

  where α is the learning rate.

When visible unit i and hidden unit j are highly correlated, the weight ω_{ij} will be more positive, and the other way around, when visible unit i and hidden unit j are not correlated. To obtain the negative samples, Monte Carlo sampling would be needed in order to reach thermal equilibrium. It is possible to use an approximation of Monte Carlo sampling for only a short time and obtain the gradient, this method is called Contrastive Divergence.

The visible biases are initialized to $\ln(p_i/(1-p_i))$, where p_i is the fraction of data points in which the ith dimension takes on the value of 1. The hidden biases are initialized to 0.

2.1.3.1 Application of RBM

RBM is only designed to learn probability distributions. A mapping from input to output is required to deal with real-world problems.

The state of a node is a binary value sampled from the Bernoulli probabilities defined by equation (43). On the other hand, the activation of a node in the associeted neural network is the probability value derived from the use of the sigmoid function in equation (45).

If RBM are used in a supervised application, a final phase of backpropagation is crucial. In this case, the RBM are used as a pretraining for the supervised learning.

2.1.4 Rubik's Code

https://rubikscode.net/2018/10/01/introduction-to-restricted-boltzmann-machines/

Reason of statistical modelling and machine learning is to detect dependencies and connections between input variables. In physics, energy is the capacity to do some sort of work. The Energy-Based Models (EBMs) use the concept that the dependecies between variables can be determined by associating a scalar value to the whole energy of the system. The associate value represents a measure of the probability that the system will be in a certain state.

Inference: process of making a prediction or a decision (when network is used).

Learning: process of finding an energy function that will associete low energies to correct values of the remaining variables and higher energy to incorrect values. (when parameters of the network are computed).

Boltzmann distribution states the probability of a state vector $\mathbf{s}$ is determined by the energy of that state vector relative to the energies of all possible binary state vectors.

The input do not need to be filled for every neuron; The Boltzmann machine will generate missing values.

Considering the neuron i is given the chance to change its binary state, initially the total input of this neuron i is computed by the weighted sum z_i of the other active neurons connected to i added by its own bias,

$$z_i = b_i + \sum_j \omega_{ij} s_j, \tag{47}$$

where b_i is the bias of neuron i, s_j is the current state of the connected neurons to i, and ω_{ij} is the weight, i.e., the connection strength between neurons j and i. The probability

that neuron i will be active is given by:

$$p(s_i = 1) = \frac{1}{1 + e^{z_i}}. \tag{48}$$

Neurons update theirselves randomly which means that the network will eventually reach a Boltzmann distribution, in which the probability of a state vector $\mathbf{v}$ is determined by the energy associated to that state vector relative to the energies of all possible binary state vectors:

$$P(\mathbf{v}) = \frac{e^{-E(\mathbf{v})}}{\sum_{\mathbf{u}} e^{-E(\mathbf{u})}}. \tag{49}$$

The energy of the state of state vector $\mathbf{v}$ is defined as

$$E(\mathbf{v}) = -\sum_i b_i s_i^{\mathbf{v}} - \frac{1}{2}\sum_i \sum_j s_i^{\mathbf{v}} \omega_{ij} s_j^{\mathbf{v}}, \tag{50}$$

where s_i is the state assigned to neuron i by state vector $\mathbf{v}$, analogously for neuron j. The Boltzmann machine is resource-demanding as there can be connections between a neuron to every other neuron in the network. Calculations can take a long time.

In the RBM architecture, there are no connections among visible units and no connections among hidden units, i.e., there are no visible-to-visible connection and no hidden-to-hidden connection. There can be only connections between visible and hidden neurons, i.e., visible-to-hidden and the other way around as a connection between two units is symmetrical.

The energy function on RBM theory is modified to

$$E(\mathbf{v}, \mathbf{h}) = -\sum_i a_i v_i - \sum_j b_j h_j - \sum_i \sum_j v_i \omega_{ij} h_j, \tag{51}$$

where a_i is the bias of the visible neuron i in state v_i, b_j is the bias of the hidden neuron j in state h_j and $_{ij}$ is the weight of the connection between visible neuron i and hidden neuron j.

The joint probability of the intire system, when hidden neurons are in state vector $\mathbf{h}$ and visible neurons in state vector $\mathbf{v}$, is

$$P(\mathbf{v}, \mathbf{h}) = \frac{e^{-E(\mathbf{v},\mathbf{h})}}{Z}, \tag{52}$$

where Z is known as the partition function

$$Z = \sum_{\mathbf{u}} \sum_{\mathbf{k}} e^{-E(\mathbf{u},\mathbf{k})}, \tag{53}$$

this partition function normalizes the probability and refers to the sum of all possible pairs of visible and hidden vectors of the system.

The RBM can be interpreted as a specialization architecture of the original Boltzmann machine, where a restriction to the neurons' connections have been put.

Given an input vector $\mathbf{v}$, the probability of a single hidden unit j of being activated is

$$P(h_j = 1|\mathbf{v}) = \frac{1}{1 + e^{-\left(b_j + \sum_i v_i \omega_{ij}\right)}} = \sigma\left(b_j + \sum_i v_i \omega_{ij}\right), \tag{54}$$

where σ is the sigmoid function. Analogous, the probability that a visible neuron i is active given a hidden vector $\mathbf{h}$ is

$$P(v_i = 1|\mathbf{h}) = \frac{1}{1 + e^{-\left(a_i + \sum_j \omega_{ij} h_j\right)}} = \sigma\left(a_i + \sum_j \omega_{ij} h_j\right). \tag{55}$$

2.1.5 Contrastive Divergence Algorithm

Here I try to compile the most information possible about Contrastive Divergence. Each author is put in a separated subsubsection. In the literature the contrastive divergence algorithm appears as CD_1, if it is a single iteration, or CD_k for k iterations.

2.1.5.1 Aggarwal

Contrastive divergence is a approximation approach to Monte Carlo sampling, it is usually applied as its fastest variant where only a single iteration is performed.

For a fixed training point of the dataset, i.e., a fixed visible state vector, for each hidden unit, its activation is computed, thus the hidden state vector is generated. Then the visible units are generated again from the previous computed hidden state. These values of the visible units are used as the sampled states stead of the ones obtained at thermal equilibrium. The hidden units are generated again from the last visible units computed.

In the positive phase, only half an iteration of simply computing hidden states is used. In the negative phase, there is at least one additional iteration to re-compute the hidden state. This difference in the number of iterations is what causes the contrastive divergence between state distributions in among both phases.

An increase in the number of iterations causes the distribution to diverge from the data-conditioned states to what is proposed by the current weight vector. The value of

$(\langle v_i, h_j \rangle_{pos} - \langle v_i, h_j \rangle_{neg})$ quantifies the amount of contrastive divergence. It is possible to increase the number of iterations k, which will improve the accuracy of the contrastive divergence. Increasing k will lead to better gradient at the cost of speed. One approach is to progressively increase the value of k as while applying the training.

- weights as initialized to small values. In each iteration, only one additional step of contrastive divergence is used. One step is enough because the difference between weights are low in early iterations.
- as gradient descent improves near a better solution, increasing k will improve accuracy. So two or three steps of CD can be applied.

2.2 **Hinton 2002**

Minimizing the log likelihood is equivalent to minimizing the Kullback-Liebler divergence between the data distribution, R_α, and the equilibrium distribution over the visible variables, P_α,

$$D(R_\alpha || P_\alpha) = \sum_\alpha R_\alpha \ln(R_\alpha) - \sum_\alpha R_\alpha \ln(P_\alpha) = E(R_\alpha) - \langle \ln(P_\alpha) \rangle_{R_\alpha}, \tag{56}$$

where the first term, $E(R_\alpha)$, is the entropy of the data distribution, and the second term is the average over the distribution given by the data distribution, R_α.

In equation (56), only the average term depends on the parameters os the model. Therefore, when minimizing the Kullback-Liebler divergence, the first term will be ignored.

FUTURE DEVELOPMENT

To be studied...

Following up the current status of the project, the next steps includes:

1. Finalization details of Boltzmann Machines, such as the update equations of the state of a unit and the description of the BM algorithm.
2. Studying Restricted Boltzmann Machines and how its learning algorithm works.
3. Dummy test to compare results of Hopfield, Boltzmann Machine and Restricted Boltzmann Machine
4. Identify an interesting problem where the application of RBM is suitable.

Chronogram

Aiming to conclude this masters dissertation the chronogram below seems to be the most coherent:

- Finalization of both Boltzmann and Restricted Boltzmann Machines by the end of **January 2020**.
- Definition of the case to apply RBM by the end of **January 2020**.
- Implementation and testing of RBM algorithm to solve proposed task by the end of **May 2020**.
- Meanwhile keep with the development of the written text in parallel which will take up to about **June 2020**.

BIBLIOGRAPHY

DUDA, R. O.; HART, P. E.; STOCK, D. G. *Patter Classification*. 2. ed. USA: Wiley-Interscience, 2000.

FISCHER, A.; IGEL, C. An introduction to restricted boltzmann machines. In: ALVAREZ, L. et al. (Ed.). *Progress in Pattern Recognition, Image Analysis, Computer Vision and Applications*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012. p. 14–36.

GÉRON, A. *Hands-On Machine Learning with Scikit-Learn and TensorFlow*. 1. ed. USA: O'Reilly Media Inc., 2017.

GOODFELLOW, I.; BENGIO, Y.; COURVILLE, A. *Deep Learning*. Cambridge, MA, USA: MIT Press, 2016. (Adaptive computation and machine learning series).

HERTZ, J.; KROGH, A.; PALMER, R. D. *Introduction to the theory of neural computation*. USA: Westview Press, 1991. v. 1. (Santa Fe Institute Series, v. 1).

HINTON, G. E. Boltzmann machines. In: _____. *Encyclopedia of Machine Learning*. Boston, MA, USA: Springer US, 2010. p. 132–136. ISBN 978-0-387-30164-8. Disponível em: ⟨https://doi.org/10.1007/978-0-387-30164-8_83⟩.

HOPFIELD, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, v. 79, p. 2554–2558, Apr 1982.

MONTÚFAR, G. Restricted boltzmann machines: Introduction and review. *arXiv e-prints*, p. arXiv:1806.07066, Jun 2018.

SUN, R. Artificial intelligence: connectionist and symbolic approaches. In: SMELSER, N. J.; BALTES, P. B. (Ed.). *International Encyclopedia of the Social and Behavioral Sciences*. Oxford: Pergamon/Elsevier, 2001. p. 783–789.

APPENDIX A – Kullback-Leibler Divergence or Relative Entropy

A.1 Kullback-Leibler Divergence

The entropy computes how uncertainty events are. If a system is quite stable, the entropy will be close to zero and will not vary a lot, while if the system is quite unstable, then the entropy will be very large. The Kullback-Leibler divergence, or relative entropy, is the measure of how different two separate probability distributions, R and P, are from each other (GOODFELLOW; BENGIO; COURVILLE, 2016). Considering these probability distribution are both discrete and are defined on the same probability space χ, the Kullback-Leibler divergence $D_{KL}(R||P)$ is given by

$$D_{KL}(R||P) = \sum_{x\in\chi} R(x) \ln\left(\frac{R(x)}{P(x)}\right), \tag{57}$$

the equation above can be read as the divergence from P to R.

It is important to notice that this divergence measurement is not symmetric,

$$\begin{aligned} D_{KL}(R||P) &\neq D_{KL}(P||R), \\ \sum_{x\in\chi} R(x) \ln\left(\frac{R(x)}{P(x)}\right) &\neq \sum_{x\in\chi} P(x) \ln\left(\frac{P(x)}{R(x)}\right), \end{aligned} \tag{58}$$

and that KL divergence, relative entropy, is always a non-negative value,

$$\begin{aligned} E &= D_{KL}(R||P) \\ D_{KL}(R||P) &= \sum_{x\in\chi} R(x) \ln\left(\frac{R(x)}{P(x)}\right) \\ &\geq \sum_{x\in\chi} R(x) \left(1 - \frac{P(x)}{R(x)}\right) \\ &= \sum_{x\in\chi} (R(x) - P(x)) \\ &= \sum_{x\in\chi} R(x) - \sum_{x\in\chi} P(x) = 1 - 1 \\ &\Rightarrow D_{KL}(R||P) \geq 0. \end{aligned} \tag{59}$$

Notice that $D_{KL}(R||P)$ will only reach zero if and only if $P(x) = R(x)$, which means that we are able to retrieve the exactly desired probability distribution at the visible units from the input data.

APPENDIX B – Smolensky Word Completion Example

B.1 Smolensky MAKE Example

(CITAR SMOLENSKY) presents a word completion example with a Restricted Boltzmann Machine. In other words, the RBM has to complete missing or unknown letters of a given word. Based on previous example that had been shown to the RBM, the machine would complete the unknown information of the word with the most probable letter taking into account the other letters known in the word.

This example uses the word 'MAKE'. (CITAR SMOLENSKY) calls the units on the hidden layer as 'knowledge atoms', and the units on the visible layer as 'feature processors'. In this appendix we will keep the nomenclature used by Smolensky.

The input to a completion task is provided by fixing some of the 'feature processors' — the known information — while the unknown information is allowed to be updated. Fixed 'features' are the units that are assigned the values -1 or $+1$, while the unknown 'features' have value 0.

The entropy computes how uncertainty events are. If a system is quite stable, the entropy will be close to zero and will not vary a lot, while if the system is quite unstable, then the entropy will be very large. The Kullback-Leibler divergence, or relative entropy, is the measure of how different two separate probability distributions, R and P, are from each other (GOODFELLOW; BENGIO; COURVILLE, 2016). Considering these probability distribution are both discrete and are defined on the same probability space χ, the Kullback-Leibler divergence $D_{KL}(R||P)$ is given by

$$D_{KL}(R||P) = \sum_{x \in \chi} R(x) \ln \left(\frac{R(x)}{P(x)} \right), \tag{60}$$

the equation above can be read as the divergence from P to R.

It is important to notice that this divergence measurement is not symmetric,

$$\begin{gathered} D_{KL}(R||P) \neq D_{KL}(P||R), \\ \sum_{x \in \chi} R(x) \ln \left(\frac{R(x)}{P(x)} \right) \neq \sum_{x \in \chi} P(x) \ln \left(\frac{P(x)}{R(x)} \right), \end{gathered} \tag{61}$$

and that KL divergence, relative entropy, is always a non-negative value,

$$\begin{aligned} E &= D_{KL}(R||P) \\ D_{KL}(R||P) &= \sum_{x\in\chi} R(x)\ln\left(\frac{R(x)}{P(x)}\right) \\ &\geq \sum_{x\in\chi} R(x)\left(1-\frac{P(x)}{R(x)}\right) \\ &= \sum_{x\in\chi}(R(x)-P(x)) \\ &= \sum_{x\in\chi} R(x) - \sum_{x\in\chi} P(x) = 1-1 \\ &\Rightarrow D_{KL}(R||P) \geq 0. \end{aligned} \tag{62}$$

Notice that $D_{KL}(R||P)$ will only reach zero if and only if $P(x) = R(x)$, which means that we are able to retrieve the exactly desired probability distribution at the visible units from the input data.