

Fundamental properties of Hopfield Networks and Boltzmann Machines for Associative Memories

Agnes Meyder Constantin Kiderlen

Machine Learning, vt 2008

Abstract - Association is a very powerful way to understand and react to the surrounding environment. It was and it is one of the most fascinating abilitys of the human brain. So, the attempt to implementd such a usefull tool was early discussed.

In the first chapter we will discuss some common features of implemented associative memories as a concept. To give some methods the second and third chapters deal with Hopfield networks and Boltzmann machines. The fourth chapter is about other architectures regarding associative memories. The essay closes with our conclusion concerning the future developement of this field.

Contents

1	Associative Memory	2
2	Hopfield Networks	3
2.1	Architecture and principles	3
2.2	Applications for the Hopfield model	4
3	Boltzmann Machines	6
3.1	Architecture and principles	6
3.2	Stochastic considerations	7
3.3	Learning and processing in Boltzmann machines	9
3.4	Restricted Boltzmann machines	10
3.5	Annealing and simulated annealing	11
4	Other Architectures	12
4.1	Bidirectional Associative Memory	12
4.2	One Shot and Exponential Correlation Memories	12
4.3	Hamming Associative Memory	13
5	Conclusions	14

1 Associative Memory

Association is an important ability of the human brain. To understand our world we learn patterns of creatures and things to determine the sort of our surrounding - to associate the correct pattern with the things we see, even if they slightly differ. But also, human beings are associating other observations correctly with a known pattern. As example we learn to know a wesp as a threat to our sweets.

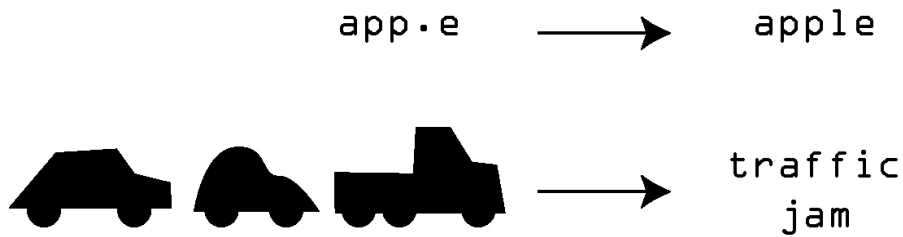


Figure 1: Auto- and hetero-association in real life.

Formally spoken, association can be an autoassociative recall: replenish an incomplete pattern into a complete stored one; and a heteroassociative recall, which produces an output pattern in response to the input pattern [11]. Such an associative memory can be implemented by a neural network.

These networks use as a base Hebb's rule:

If two nodes in a network are active simultaneously, reinforce the connection between them.

The nodes of these networks are usually fully interconnected, but also hierarchical neural networks can work [5]. This gives them, additional to the strong resistance against noise, a high computational speed because of parallelism.

We will discuss Hopfield networks and Boltzmann machines because of their important role as a base in this field. Then we will take a short glimpse into some modern associative memories.

2 Hopfield Networks

2.1 Architecture and principles

Hopfield introduced 1984 a fully interconnected network with symmetric weights ($w_{ij} = w_{ji}$) and without self connections ($\forall i : w_{ii} = 0$).

For the subsequent formulas, the following notation will be used: In a network with n units the states of the i -th unit will be denoted by u_i ($i = 1, \dots, n$), while $u \in \{0, 1\}^n$ is the configuration vector containing all the u_i ; let $W = (w_{ij})$ ($i, j = 1, \dots, n$) the symmetric weight matrix with $w_{ii} = 0$, and $\theta = (\theta_1, \dots, \theta_n)$ the threshold vector containing a threshold θ_i for each unit i .

So the energy function is as follows for a binary Hopfield network:

$$E(u) = -\frac{1}{2}uWu^T + \theta u^T \quad (1)$$

$$= -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n u_i w_{ij} u_j + \sum_{i=1}^n u_i \theta_i \quad (2)$$

and for a continuous network:

$$E(u) = -\frac{1}{2}uWu^T + \theta u^T + \frac{1}{\beta} \sum_i \int_0^{V_i} g^{-1} v dv \quad (3)$$

$$= -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n u_i w_{ij} u_j + \sum_{i=1}^n u_i \theta_i - \sum_i \int_0^{\theta_i} g^{-1} v dv \quad (4)$$

The binary network is updated in the following way [8]:

1. pick a node i randomly
2. check if $x_i = \sum_j u_j w_{ij} - \theta_i$ positive, then let $u_i = 1$, else $u_i = 0$
3. repeat for the whole net

A continuous Hopfield network uses the above shown energy equation and, as a difference in step 2 of the update procedure for binary Hopfield networks, a continuous function for x_i and for $u_i = g(x_i)$ a monotonic increasing differentiable function.

Hopfield has stated a theorem about the convergence of a Hopfield net [18]:

Hopfield theorem: *If (w_{ij}) is a symmetric matrix and if $\forall i : u_i = g(x_i)$ is a monotonic increasing differentiable function, then E is a Lyapunov function for motion equations.*

A Lyapunov function is known to be convex and so, under the given conditions, the theorem guarantees convergence to a local minimum of the neural net, where the following equalities hold:

$$\forall i : u_i = g(x_i) \wedge x_i = \sum_j w_{ij} u_j + \theta_i$$

It is also known that the amount of storable patterns is restricted to about 15% of the amount of the nodes in the network [12].

As only decreasing of the system's energy is possible, Hopfield networks can get stuck in local minima. For this reason, Boltzmann Machines offer improved prospects.

2.2 Applications for the Hopfield model

There are manifold examples to illustrate the properties of Hopfield networks, many of them handle combinatorial problems such as the Picking Stone Problem [9], the Travelling Salesman Problem [14] and Perelman's Nine Flies Problem treated in [10].

Another combinatorial task involving constraints and inhibitory units is the N-Queens Problem [14]. Consider a $N \times N$ chess board and N queens for $N \geq 4, N \in \mathbb{N}$. The task is to find the possible solutions to place all queens on the board such that none of them can beat each other, i.e. exactly one queen in each row, column and diagonal.

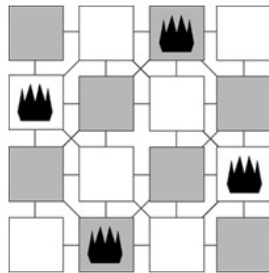


Figure 2: A solution to the four queens problem.

To model the chess board, a $N \times N$ -layer of units is used. Denote the states of the units corresponding to the chess-board fields by x_{ij} where $i, j \in \{1, \dots, N\}$ refer to column and row number. Let $x = (x_{11}, \dots, x_{NN})$ and w_{ij}^{kl} the connection weight between units ij and kl . Then $x_{ij} = 1$ iff a queen is placed on square ij .

The given constraint is realized by inhibitory connections within rows, columns and diagonals, i.e. the input to each unit has to be smaller than zero if another unit in the same row, column or diagonal is set to state 1. This is achieved by a set of objective functions E_r, E_c, E_d as follows:

$$\begin{aligned}
 E_r(x) &= \sum_i \left(\sum_j x_{ij} - 1 \right)^2 \\
 E_c(x) &= \sum_j \left(\sum_i x_{ij} - 1 \right)^2 \\
 E_d(x) &= \sum_{\mathcal{D}_k \in \mathcal{D}} \left(\sum_{(i,j) \in \mathcal{D}} x_{ij} - 1 \right)^2
 \end{aligned}$$

where \mathcal{D} is the index set of diagonals and \mathcal{D}_k contains the index pairs for the k -th diagonal.

These functions have their minima for configurations which satisfy the problem constraints and are assembled to

$$E = E_r + E_c + E_d$$

which is to be minimized. The weights w_{ij}^{kl} are all set to -2 for coherent (in the sense of the constraints) units, 0 otherwise, and the thresholds all to -1 . Thus, a unit is set to state 1 iff none of the coherent units inhibits it by a input of -2 .

The energy function stated above has more than one minimum - local minima may get adopted, if less than N queens are posed on the board. Hence, the network won't find suitable solutions only.

3 Boltzmann Machines

The Boltzmann machine extends the concept of Hopfield networks by a stochastic update method. We will describe the basic architecture, followed by deeper explanations of the stochastic foundations and a description of the learning and processing algorithms. Another section is dedicated to the modification towards the restricted Boltzmann machine. A short overview of the (simulated) annealing technique is supplied in the last section.

3.1 Architecture and principles

Similar to a Hopfield network, a Boltzmann machine is a network of units which are fully interconnected by bidirectional connections with symmetric weights, whereas no self-connections are allowed. These units have binary values $\{0, 1\}$, referring to states OFF and ON for each unit. That means, the whole system is in a certain state, sometimes called configuration, at every time. Due to the configuration the system has a corresponding value of its energy function. The crucial difference from Hopfield networks is the way of updating the states of the units which is determined by stochastic decisions here.

For the energy function we obtain, as for Hopfield networks,

$$E(u) = -\frac{1}{2}uWu^T + \theta u^T \tag{5}$$

$$= -\frac{1}{2} \sum_{j=1}^n \sum_{i=1}^n u_i w_{ij} u_j + \sum_{i=1}^n u_i \theta_i \tag{6}$$

with the same notation as used in section 2.1. Due to the probabilistic update rule, a Boltzmann machine is able to transit to states on a higher energy level - in contrast to a Hopfield network. This avoids getting stuck in local minima of the energy function in minimization problems.

Remark on the threshold θ : Ackley et al. suggest in [1] to replace the threshold by a unit in an always-ON state, which is connected to all other units i with weights $-\theta_i$, called bias b_i of the unit. The effect would be the same, but the modification may simplify calculations.

For updating the network a unit i is chosen at random and its input

$$G := \Delta E_i = \sum_{j=1}^n u_j w_{ij} - \theta_i$$

is calculated. G is the energy gap for unit i , i.e. the difference of the systems energy if $u_i = 0$ and the energy if $u_i = 1$, which can be seen from (6).

Thus, $G > 0$ means, that the energy of the whole system is higher if $u_i = 0$ and vice versa (i.e. higher energy for $G < 0$ with $u_i = 1$). So it is preferable for the system to have the unit's state ON or OFF, respectively, since the aim is to minimize $E(u)$.

This fact comes out in the decision of the unit's next state, which is taken independently from the current unit state, namely the probability

$$p := P(\text{unit } i \text{ will be ON}) = P(u_i = 1) = \frac{1}{1 + e^{-G/T}}$$

where $T \in \mathbb{R}^+$ is a "temperature" parameter, chosen arbitrary but fixed. Let $q := 1 - p$ the probability that unit i will be OFF.

The temperature parameter T is important for the method of simulated annealing, which is described in section 3.5. Here, only its relation to thermodynamics and its meaning for the update decision shall be mentioned: The reason to call T temperature is the origin of the model in thermodynamics, studied by the physicist Ludwig Boltzmann in the 19th century.

Let S the state space for a system and U, V the events that the system is in state u or v , respectively. From the so-called Boltzmann distribution, giving the probability

$$P(U) = \frac{e^{-E(u)/T}}{\sum_{s \in S} e^{-E(s)/T}} \quad (7)$$

one can derive

$$\frac{p}{q} = e^{(E(v) - E(u))/T} = e^{G/T}$$

with $U, V =$ "current unit will be in state ON or OFF, respectively".

It follows reasonably, that if $G = 0$, i.e. the state of the unit has no effect on the energy of the system configuration at the moment, the probabilities p, q for a "change" to unit state 1 or 0 are equally $\frac{1}{2}$. Furthermore, a high energy gap G causes p to be closer to 1, whereas a low and negative G causes p to be small. One may interpret that fact by saying that the probability of the system to move upwards a high amount in the energy function is small, while small increasings are more probable.

Considering T , it is obvious that the smaller T is, the smaller becomes $e^{-G/T}$ both for positive and for negative values of G . That means, jumping to higher energy levels is more unlikely for decreasing T . Hence, the Boltzmann machine approximates the behaviour of a Hopfield network with $T \rightarrow 0$.

3.2 Stochastic considerations

A Boltzmann machine is a discrete Markov chain with the $2^n \times 2^n$ transition matrix

$$M = (P(U|V)) \quad \text{for } u, v \in S$$

i.e. the entries of M are the conditional probabilities for the system to change from state u to state v . Recall from above that

$$p = P(u_i = 1) = \frac{1}{1 + e^{-G/T}} = 1 - q$$

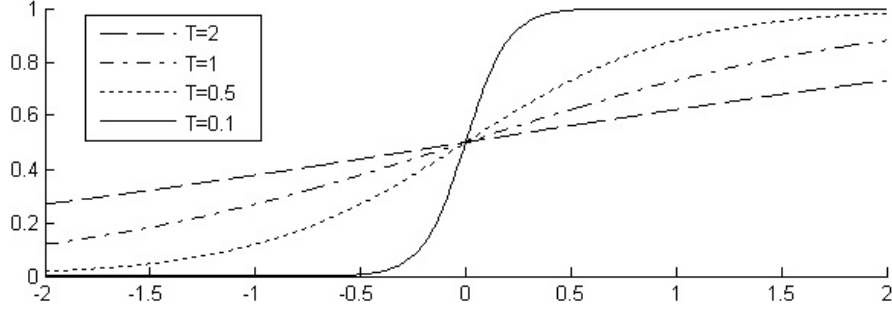


Figure 3: The probabilities p as function of G for different temperatures T

so that the entries of M can be expressed as

$$m_{ij} = \begin{cases} \frac{1}{n} \cdot p & \sum |u_i - v_i| = 1 \text{ and } \sum u_i - \sum v_i = -1 \\ \frac{1}{n} \cdot q & \sum |u_i - v_i| = 1 \text{ and } \sum u_i - \sum v_i = +1 \\ 0 & \text{otherwise} \end{cases}$$

for $i \neq j$; the entries on the diagonal are

$$m_{ii} = 1 - \sum_j m_{ij} = \frac{1}{n} \sum_j q(1 - u_j) \cdot pu_j$$

The prefactor $\frac{1}{n}$ stands for the uniform probability for each unit to be chosen for an update. Next, transitions to configurations with a Hamming distance $d(u, v) > 1$ from the original configuration are impossible ($\Rightarrow m_{ij} = 0$) and all other probabilities depend on from which state the chosen unit changes to which state. The entries on the diagonal result from the sum over all units j , where each unit credits a $\frac{1}{n}p$ to the sum iff $u_j = 1$ and a $\frac{1}{n}q$ iff $u_j = 0$, since these are the only possibilities to leave the configuration unchanged.

Starting at an initial probability distribution $\delta_0 \in [0, 1]^n$ for the $u \in S$, it is possible to describe the distribution δ_k after k updates by

$$\delta_k = \delta_{k-1}M = \delta_{k-2}M^2 = \dots = \delta_0M^k$$

The issue is now to find a stationary distribution π , so that

$$\pi = \pi M$$

The above stated Markov chain is irreducible and positive recurrent, i.e. each configuration can be accessed from each other configuration in one or more steps and the expectation $E(R_i)$ of the number of steps to re-visit a state i is finite (see e.g. [16] for details on Markov chains). Then, as an important result of Markov chain theory, a stationary distribution π exists, and, since the Boltzmann machine is an aperiodic Markov chain (since each configuration can transit to itself with probability greater than zero), it holds that

$$\delta_k \rightarrow \pi \text{ as } k \rightarrow \infty$$

In [14] it is shown that the stationary distribution, equivalent to a thermal equilibrium, for fixed T in Boltzmann machines is the Boltzmann distribution stated above in (7).

3.3 Learning and processing in Boltzmann machines

For the use as a completion machine, e.g. to supply missing parts of corrupted patterns, the set of units is split into a set of visible and a set of hidden units, which are still fully interconnected.

The learning algorithm for Boltzmann machines introduced by Ackley, Hinton, Sejnowski in [1] and refined in [7] deals with a measure indicating to what degree the weights in the network are appropriate for modeling the environment. That measure will be the objective function and shall be minimized by adjusting the weights. The environment itself consists of configurations appearing due to a certain distribution.

First one clamps a training configuration vector over the visible units, long enough to allow the system to reach its thermal equilibrium. Denote the probability for such a vector v to appear on the visible units by $P^+(v)$. The distribution is independent from the weights, but determined by the environment.

Next, the distance measure between actual, given distribution and the distribution P^- which is obtained by running the system freely (i.e. without clamped units) is given by

$$\Gamma = \sum_v P^+(v) \ln \frac{P^+(v)}{P^-(v)}$$

and refers to the information gain (Kullback, Renyi). This distance measure is not a metric, since it is not symmetric - the fact that more frequent events should have a heavier effect on the measure, thus by weighting by the actual probability, is reflected here.

Now, as the objective function Γ is defined, gradient descent with respect to the partial derivatives by w_{ij} is used, i.e.

$$\frac{\partial \Gamma}{\partial w_{ij}} = -\frac{1}{T}(p_{ij}^+ - p_{ij}^-)$$

where p_{ij}^+ is the averaged (over all environment inputs) probability (expectation) that unit i and j are both ON, measured at equilibrium and when clamped, whereas p_{ij}^- is the corresponding value for the free-running system. The calculation for the derivative can be found in [1].

With an arbitrary parameter $\epsilon \in \mathbb{R}^+$ the weights are updated by

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij} = w_{ij} + \epsilon(p_{ij}^+ - p_{ij}^-) \quad (8)$$

For the recall procedure one may divide the visible unit set into input and output units. The result of the training should be that for each configuration v

the probability to converge to v as equilibrium configuration is equal, no matter whether

- a random initial configuration is chosen
- an arbitrary input is clamped to the input units.

Hence, when an incomplete input vector is presented and the available pattern parts are clamped to the input units, the Boltzmann machine converges to the thermal equilibrium and presents the missing pattern parts on the output units. This is achieved by stepwise simulated annealing, where the temperature steps and the number of update cycles, referred as annealing schedule by [17], have to be chosen arbitrary for the current problem.

3.4 Restricted Boltzmann machines

The Boltzmann machine entails a certain problem, namely the slow and extensive learning procedure, among others induced by the large number of necessary learning and annealing steps and the required accuracy of the calculations to avoid loss of significance.

Especially for multi-dimensional tasks the velocity will decrease; the number of bidirectional connections is $\frac{(n^2-n)}{2}$ for a network with n units.

As an improvement, in [2] a parallel implementation of the Boltzmann machine is described, but here the more recent concept of the so-called restricted Boltzmann machine as discussed in [15] and [21] shall be introduced.

The restricted Boltzmann machine (RBM) divides the visible and hidden units more consequently, in fact by introducing a visible and a hidden layer. Inside each layer there are no connections between units allowed, thus each visible unit connects to each hidden unit. For image recognition one may correspond them to image pixels and feature detectors respectively.

Thus it is reasonable to use "joint" configurations (v, h) of visible and hidden unit states. Denote the sets of visible and hidden units by \mathcal{V}, \mathcal{H} and the sets of the corresponding possible partial configurations by V, H . Then the energy function of the system for a joint configuration (v, h) yields

$$E(v, h) = - \sum_{i,j} w_{ij} v_i h_j - \sum_i b_i v_i - \sum_j b_j h_j$$

with $0 \leq i \leq |\mathcal{V}|$ and $0 \leq j \leq |\mathcal{H}|$.

The distribution over V is given by

$$p(v) = \sum_{h \in H} e^{-E(v,h)} \left(\sum_{u \in V, g \in H} e^{-E(u,g)} \right)^{-1}$$

To reach the stationary distribution, the units are updated by a probability underlied by the logistic function $\sigma(t) = (1 + e^{-t})^{-1}$. The state of hidden unit

j or visible unit i is set to 1 or ON with

$$p_1(j) = \sigma \left(b_j + \sum_i w_{ij} v_i \right)$$

$$p_1(i) = \sigma \left(b_i + \sum_j w_{ij} h_j \right)$$

After gradient descent on $\ln p(v)$ with respect to w_{ij} the weight change during training is, similar to (8), given by

$$\Delta w_{ij} = \epsilon (p_{ij}^+ - p_{ij}^-)$$

with $p_{ij} = \text{Expectation}(v_i = 1 = h_i)$, where $+$ denotes the expectation on the actual data, $-$ on the modeled data.

Restricted Boltzmann machines were, for example, trained on character recognition and collaborative filtering, i.e. a system which suggests certain objects based on ratings by different users. More precisely, the machine guessed which movies a user might like, according to his and other users' ratings on a set of movies. Hinton et. al. succeeded even with datasets of more than 100 million ratings [15].

3.5 Annealing and simulated annealing

Annealing, a process from metallurgy, works as follows:

A metallic material is heated and then cooled down slowly, so that, at each temperature step, the crystalline lattice can find its optimal position. By gradually decreasing the materials temperature, the molecule system becomes more and more stable, but is still able to compensate structural irregularities by changing to higher-energy states and thus overcoming local energy minima.

The behaviour of the Boltzmann machine is, as already mentioned, a special case of simulated annealing. To simulate this process in general, one uses the energy function of the considered model and assigns acceptance probabilities

$$p_{\Delta E(u,v)} = \frac{1}{1 + e^{\Delta E(u,v)/T}}$$

for the energy gap ΔE for two configurations u, v as stated in 3.1. In this general model it's not necessary that the units have binary states and one is not restricted to change the state of only one single unit per time step.

4 Other Architectures

The basic ideas of the Hopfield network and the Boltzmann machine are the more vintage architectures for associative memories. In the following we will present three new ideas how to implement associative memories in short.

4.1 Bidirectional Associative Memory

A bidirectional associative memory (BAM) is a neural network that uses forward and backward bidirectional search for pattern recognition.

Bidirectional search is a common term for starting from goal and start point and searching from both sides a path to the other side respectively. In the optimal situation, these paths should meet in the middle of the search tree. Here the start is the input pattern and the goal one of the stored patterns.

BAM is based on correlation matrix summation (W correlation matrix, x^i , y^i stored pattern):

$$W = \sum_i (x^i)^T y^i$$

Correlation is a synonym for the strength and direction of the relatedness between two vectors. This algorithm was first proposed 1988 by Kosko and some new features were added by Zheng et al. [6].

4.2 One Shot and Exponential Correlation Memories

The One Shot Associative Memory (OSM) and the Exponential Correlation Associative Memory (ECAM) have both the same amount and structure of nodes. Both of them have an input and an output layer of n nodes as well as hidden layers with m nodes, when m patterns should be learned [20].

Regardless of that, OSM is a feedforward network but ECAM a recurrent network.

OSM learns the patterns in the training phase through calculating hamming distances (bit by bit) for each pattern in combination with a learning rate for each component of each pattern. A high component importance results in a high learning rate for this component.

When the OSM is trained, in the test phase a pattern is presented. After getting all patterns, that can be the winners, the Hamming distance is computed to find the best matching pattern.

As OSM uses a static domain of attraction, it seems to be better for typed letters and do bad on handwritten letters [13].

For further information about the detailed learning algorithm we refer to the paper of Wu et al.[20].

ECAM learns the patterns with the help of only one learning rate for each pattern.

In the test phase, ECAM takes the input pattern and with a correlation equation it assigns each stored pattern a negated normalized hamming distance towards

the input. These values are then distributed over a greater range in the weighting function and the average pattern is represented to the network again. This iteration is done until the net stabilizes or the maximal number of rounds is reached.

In opposite to OSM the ECAM has dynamic attractive domains that are defined in the test phase. ECAM was tested against OSM and did better in handwritten letters [13].

ECAM was introduced in the paper of Chiueh et al. [4].

4.3 Hamming Associative Memory

A Hamming Associative Memory is a feedforward neural net. It is based on computing the Hamming distance between the input and the stored patterns. The output will be the one stored pattern with the minimal distance to the input pattern. There are several variants available, some which are summarized by Watta et al. in [19].

5 Conclusions

The time of using an associative memory in the fields of computer science seems to be over since some years ago. Most of the papers and books we looked through were so old (1975 - 1995) that their focus was mostly on the actual implementation of an associative memory than a full exploration of its abilities in huge projects. So we actually checked much more material, than that was used in this essay.

And even if there is the Free Lunch Theorem, which states that there actually exists a specific subset of problems for associative memories, where they will perform the best, in most of associative memory application areas different algorithms are used due to some reasons like [19]:

1. too high computation time because of a high amount of weights that need to be updated
2. limited storage capacity

But nevertheless some researchers work in this area and probably a new application field will be found.

For Boltzmann machines, there are indeed topics where they can succeed - even in psychology or in studying chaotic behaviour of neural circuits. This is probably due to their ability to discover non-trivial correlations in data sets which a human might never expect. Especially the Restricted Boltzmann Machine offers a new approach to certain problems. For example, the formerly discussed rating-suggestion-system is a very common marketing tool on certain online-shops.

References

- [1] D.H. Ackley, G.E. Hinton, and T.J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9:147–169, 1985.
- [2] B. Apolloni and D. de Falco. Learning by parallel boltzmann machines. *IEEE Transactions on Information Theory*, 37:1162–1165, 1991.
- [3] D. E. Van Den Bout and T. K. Miller III. Improving the performance of the hopfield-tank neural network through normalization and annealing. *Biological Cybernetics*, 62:129–139, 1989.
- [4] Tzi-Dar Chiueh and Rodney M. Goodman. Recurrent correlation associative memories. *IEEE Transactions on Neural Networks*, 2, 1991.
- [5] Kunihiko Fukushima. A hierarchical neural network model for associative memory. *Biological Cybernetics*, 50:105–113, 1984.
- [6] Sidney Nascimento Givigi Gengsheng Zheng and Weiyu Zheng. A new strategy for designing bidirectional associative memories. *Springer Verlag*, 2005.
- [7] G.E. Hinton and T.J. Sejnowski. Learning and relearning in boltzmann machines. *Parallel distributed processing: explorations in the microstructure of cognition*, 1: foundations:282–317, 1986.
- [8] John J. Hopfield. Hopfield networks. *Scholarpedia*, 2(5):1977:revision 39877, 2007.
- [9] J.-E. Hu and P. Siy. The ordering-oriented hopfield network. *IEEE International Conference on Neural Networks. IEEE World Congress on Computational Intelligence.*, 7:4693–4698, 1994.
- [10] John G. Keating. Hopfield networks, neural data structures and the nine flies problem : Neural network programming projects for undergraduate s. 1993.
- [11] T. Kohonen. *Sel - Organization and Associative Memory*. Springer Verlag, 1989.
- [12] Matsuoka. A model of orthogonai auto-associative networks. *Biological Cybernetics*, 62:243–248, 1990.
- [13] Tahar Kechadi Orla McEnery, Alex Cronin and Franz Geiselbrechtinger. Suitability of two associative memory neural networks to character recognition. *Springer Verlag*, 2004.
- [14] Raïj, $\frac{1}{2}$ l Rojas. *Neural Networks*. Springer-Verlag, Berlin, 1996.
- [15] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. Restricted boltzmann machines for collaborative filtering. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 791–798, New York, NY, USA, 2007. ACM.
- [16] David Stirzaker. *Stochastic Processes and Models*. Oxford University Press, 2005.

- [17] Tafsir Thiam. The boltzmann machine. *International Joint Conference on Neural Networks IJCNN*, 6:4428–4431, 1999.
- [18] Jan van den Berg. The most general framework of continuous hopfield neural networks. 1996.
- [19] Paul Watta and Mohamad Hassoun. Generalizations of the hamming associative memory. *Neural Processing Letters*, 13:183–194, 2001.
- [20] Wu and Batalama. An efficient learning algorithm for associative memories. *IEEE Transactions on Neural Networks*, 11, 2000.
- [21] Xian-Hua Zeng, Si-Wei Luo, and Jiao Wang. Auto-associative neural network system for recognition. *International Conference on Machine Learning and Cybernetics*, 5:2885–2890, 2007.