A Practical Comparison Between Hopfield Networks and Restricted Boltzmann Machines as Content-Addressable Autoassociative Memories

Javier Beltrán JAVIERBELTRANJ@GMAIL.COM

Guillermo Bernárdez GBG1441@GMAIL.COM

Juan Lao-Tebar NITZING@GMAIL.COM

Jorge Rodríguez J.RODRIGUEZ.MOLINUEVO@GMAIL.COM

Abstract

In this paper we analyze the performance of Hopfield Networks and Restricted Boltzmann Machines when they are used as autoassociative memories for several kinds of addressable binary content. The aim of this work is to recommend to the reader a model that offers the best results for a specific given pattern data set, based on the results obtained after testing several parameterized models against different random pattern data sets that satisfy the same statistical properties.

Keywords: Artificial Neural Networks, Recurrent Neural Networks, Hopfield Networks, Restricted Boltzmann Machine, Autoassociative Memory, Addressable Content

1. Introduction

Attractor networks such as Hopfield Networks [Hopfield (1982)] and Restricted Boltzmann Machines [Smolensky (1987)] are widely used as binary content-addressable memory systems, [DAI (1998)][SCH (1995)][Krizhevsky and Hinton] and the theory behind them has been deeply studied in the past years. [S. V. B. Aiyer and Fallside (1990)][Zhuang and Huang (1993)][Nagatani and Hagiwara (2014)] Both models have been compared on specific problems [R. Sammouda and Nishitani (1996)] and mathematical relations of equivalence have been demonstrated. [Bar (2012)][Agl (2013)]

In this paper we analyze the behaviour of both models after being trained with several random pattern data sets that satisfy certain statistical properties. These properties can be extracted from any kind of binary data set and, in conjunction with the result tables presented in this paper, the reader can have an approximate idea about what model offers the best results for his specific problem. In addition, if the reader has a sample of expected input for the system, it is possible to extract other statistical properties from the sample and compare them with the statistical properties of the tested random input pattern data sets.

Hopfield Networks are known to have a limited number of patterns that can be successfully stored, a.k.a. *capacity*, as well as to be very sensitive to correlations between the training samples, which aggravates their ability to generate correct attractors and also reduces the capacity of the network[Hopfield (1982)][Storkey (1997)]. For this reason, we expect bad performances of these networks both when considering training samples very correlated and when considering large amounts of patterns to store.

Regarding Restricted Boltzmann Machines, we are not aware of any limitation, We presumably expect this networks to have a higher ability to memorize large amounts of patterns than Hopfield Networks, but perhaps at the expense of a smaller ability to recall these memories from similar ones.

2. Analyzed Models

Hopfield Networks are well-known content-addressable memories for which the Hebbian learning rule has been the traditional training approach. [Hopfield (1982)][Hebb (1950)] This paper analyzes the Storkey learning rule [Storkey (1997)] as well as the previously mentioned Hebbian. Storkey is being considered because it has been proved to provide a great increase in the capacity of the network, that is, it is able to recall more patterns.

Additionally, units in the Hopfield Network may be updated either synchronously or asynchronously. This paper only contains an analysis of the asynchronous method, since the synchronous is considered less realistic based on the absence of observed global clock influencing analogous biological or physical systems of interest. [MacKay (2003)]

Boltzmann Machines are usually considered a stochastic analogue of the Hopfield Networks. [Ack (1985)] Despite their theoretical usefulness, it is well known that in practice they cannot learn properly for large enough problems. Thats why we are considering its constrained version, the Restricted Boltzmann Machine, that permits an efficient training using the Contrastive Divergence algorithm. [Carreira-Perpinan and Hinton (2005)]

Learning in a Restricted Boltzmann Machine is dependent on several parameters that have to be tuned appropriately. Our experiments try to follow Hintons recommendations [Hinton (2012)] on the following issues, although some details have been implemented in a different way:

- An appropriate estimation of the number of hidden units required is necessary to prevent overfitting, but hard to define. Our decisions are inspired in Hintons recommendations.

- A mini-batch containing one element of each class is proposed by Hintons, but we find that smaller mini-batches work better in our study. In fact, we use an online strategy for training.

- Stopping criteria of the training algorithm. Given the norm of the delta for the weights in each epoch, the training algorithm stops when the summation of the norm of the delta weights for the last 1000 epochs do not decrease during other 1000 epochs. At this point it is supposed that the weights are oscillating around their target value.

- The weights and offsets initialization follows the recommended values: small random values for the weights chosen from a normal $N(0, 0.01)$ distribution, hidden offsets equal to 0, and visible offsets set to $\log \frac{p_i}{(1-p_i)}$, where p_i is the proportion of training vectors in which unit i is on.

- The learning rate parameter is also crucial for a good performance of the network. We fix it to 0.001 in all our models since this value provides better results than any other checked.
- Momentum is an improvement on the speed of the learning rule. We follow the recommendations, setting its value to 0.5 during the first stages of the learning process (5 epochs) and then incrementing it to 0.9.
- Theoretically, weight decay helps learning and improves generalization. However, this seems not to apply in our case, as the obtained results are worse when we use it. For this reason, we decide to disable this parameter in our final models.

Appendix B presents a table of the analyzed models in this experiment, with explicit values for each of its parameters.

3. Pattern Data Set Statistical Properties

In abstract, a set of statistical properties can be extracted from any kind of binary pattern data set. For this experiment we generated random pattern data sets that have a predefined set of statistical property values.

Let $d(A, B)$ be the distance[1] between the couple of patterns A, B. If we choose randomly (with a uniform distribution) a couple of 2 different patterns A, B from a pattern data set P and we define $D_P = d(A, B)$, we conclude that D_P is a random variable that follows a normal distribution, with mean μ_P and standard deviation σ_P.

Basically, the following statistical properties define a pattern data set P:

- $|P|$, the number of patterns in the set.
- n, the dimension of the patterns in P.
- $\frac{\mu_P}{n}$
- $\frac{\sigma_P}{n}$

We point out that we use this mean distance between two random patterns of the Pattern Data Set as a direct measure of their correlation; the smaller the distance is, the higher the number of components shared, i.e. the higher the correlation.

The reader of this paper can extract these statistical properties from any desired pattern data set, and use as a reference the most similar pattern data set analyzed in this experiment. Appendix A presents a table of the generated and analyzed pattern data sets, with explicit values for their statistical properties.

1. For every pair $A, B \in \{0,1\}^n$, we define the distance d between A and B, $d(A, B)$, as the number of different components between both vectors; formally, $d(A, B) = \sum_{i=1}^{n} |A_i - B_i|$. Note that, with this definition, d is the restriction of the well-known L_1 distance to the space $\{0,1\}^n$.

4. Input Data Set Statistical Properties

Input data sets have a strong relation with the original pattern data set used to train the model. It is possible to extract some statistical properties from this relation, and they can be used to have an idea about the performance of the model for input data sets with the same properties.

For each input vector v, let $minD(v, P) = d(v, p)$, where p is the pattern from the pattern data set P such that $d(v, p) \leq d(v, p')$, $\forall p' \in P$, $p' \neq p$.

For a given pattern data set P, if we choose randomly (with a uniform distribution) an input vector v from an input data set I and we define $D_I = minD(v, P)$, we conclude that D_I is a random variable that follows a normal distribution, with mean μ_I and standard deviation σ_I.

Basically, the only proposed statistical property that defines an input data set is $\frac{\mu_I}{\mu_P}$, the mean minimum distance between a random input v and the closest pattern to v, expressed as a proportion of the mean distance between 2 random patterns of the pattern data set. We consider that σ_I has not huge relevance for our analysis.

The reader of this paper can extract these statistical properties from any sample input data set passed to the system, and use as a reference the most similar input data set analyzed in this experiment. Appendix C presents a table of the generated and analyzed input data sets, with explicit values for their statistical property.

5. Experiment

In this experiment we generated stochastically 28 different pattern data sets (with different statistical properties), 6 different models (with different determined combinations of tuned parameters) and 5 different random input data sets (with different statistical properties). The characteristics of each of these items are defined in appendices A, B, C.

The next step of the experiment consisted in training each of the 6 models with each of the 28 pattern data sets, obtaining 168 training and validation results that are presented in appendix D.

Finally, we tested each of the 168 trained models with each of the 5 input data sets, obtaining 840 test results that are presented in appendix E.

This process has been executed 8 times, and the tables presented in appendices A, B, C contain the mean and standard deviation of the generated and obtained values.

This experiment can be reproduced following the steps explained in appendix F.

6. Analysis of the Results

Before presenting the obtained results after training and testing the models, we need to make an explanation, in regards to the dimensionality of the pattern data sets:

Assumption 1 *We are computing many measures as a proportion of the dimension of the patterns, and we assume that the obtained results may not depend on that pattern dimension. Some tests that we have performed considering both lower and higher pattern dimensions suggest that the behaviour does not change when the dimension varies, but this should be checked more rigorously in Future Work.*

We have separated the results for both phases mentioned above, first we will start with the training phase results.

6.1. Training and Validation Results

Appendix D presents the training and validation results we obtained for the analyzed models. The first thing we need to note is the possible variations in the learning rules. Hopfield Networks trained with the Storkey rule clearly outperformed the ability to memorize patterns of those trained with the Hebbian rule, independently of the correlation (or distance) between the patterns of the pattern data set. The results suggest an increase of the capacity using the Storkey rule, as we expected.

An issue that has been commented before and is presented in several publications is the correlation between patterns, which clearly affects the capacity of Hopfield Networks. As the mean distance between two random patterns of the pattern data set is increased (i.e. the correlation is reduced), we observe that these networks are able to successfully store a higher number of patterns

Furthermore, results show a much higher robustness of the RBM with respect to both the correlation of the patterns of the pattern data set and the number of patterns that must be stored. In some configurations, such as M3 and M4, RBM is able to remember almost all patterns even when they are numerous and highly correlated. Hence, for pattern data sets whose μ_P is lower or equal than 30% of the dimension of the patterns (i.e. whose patterns exhibit a strong correlation), Restricted Boltzmann Machines properly configured provide much better ratios of stored patterns than any Hopfield Network.

When dealing with pattern data sets with less correlated patterns (i.e. with mean distances between patterns higher than 30% of the dimension of the patterns), Hopfield Networks trained with the Storkey rule achieve similar ratios of successfully stored patterns than those of RBM, all of this as long as the capacity of the Hopfield Network is not largely exceeded by the number of patterns to store [Storkey (1997)]. The increase of the capacity using the Storkey rule allows us to obtain very good results with Hopfield Networks for a considerably wide range of numbers of patterns to be memorized.

Finally, we also notice that RBM properly configured seem to have a very large capacity compared to Hopfield Networks. Therefore, if the considered pattern data set contains a higher number of patterns to be stored than the relative fractions analyzed in this paper, it will be reasonable that the reader considers the use of an RBM instead of a Hopfield Network.

6.2. Testing results

Appendix E presents the testing results we obtained for the analyzed models. These results can be summarized as follows.

To begin with, as we have described in the previous subsection, Hopfield Networks trained with the Hebbian rule show very bad performances in terms of memorizing patterns when they are compared to those trained with the Storkey rule, mainly because of its lower capacity. The difficulties of the learning process are obviously inherited in these testing results, where we test the ability of the models to recall the original patterns: they provide the worst performance by far.

We have observed a general trend shared by all models with respect to the input properties: the ability to recall the original patterns decreases as the inputs differ more from them. This behaviour was expected.

In our specific case, both RBM configurations considered (M_3 and M_4) only differ from the number of hidden neurons. We note that they obtain similar results in the majority of cases, though the model with the higher number of hidden neurons provides a slightly better overall performance.

We also observe that the ability of RBM to recall the training patterns given similar input vectors now depends on both the number of stored patterns —more patterns imply lower ability— and the correlation between the training patterns —the more decorrelated the patterns are, the higher ability to recall them—. The latter dependency seems stronger.

However, as happened in training, with the most correlated pattern data sets RBM still performs better than Hopfield Networks. Since Hopfield can store a lower number of patterns than RBM, this behaviour was expected. Nevertheless, as the correlation between patterns decreases, results suggest that Hopfield Networks using the Storkey rule outperform RBM when restoring patterns that highly differ from the original ones.

It seems then that Hopfield Networks generate deeper attractors than RBM. It can be observed that, given a pattern data set, the ability of Hopfield Networks to recall patterns remains approximately constant when augmenting the variability of the input patterns with respect to the original stored patterns. Yet, in RBM there is a clear loss of restoration power as the variability of the inputs increases. Hence, if the patterns of the pattern data set are sufficiently decorrelated and it is expected to receive highly variable inputs, Hopfield Networks appear to be more robust for recalling, as long as the number of patterns to be stored does not overtake the capacity of the network.

7. Conclusions and future work

In this project we do a practical analysis on the performance of two models used as autoassociative memories for several kinds of addressable binary content: Hopfield Networks and Restricted Boltzmann Machines. For this purpose we generate random pattern data sets —with different properties—, we train the models and then perform several tests in order to extract some conclusions from the results.

As we observed in the training and validation process, in all the cases, Hopfield Networks trained with the Storkey learning rule are able to store a higher number of patterns than those trained with the Hebbian rule. Additionally, as the distance between patterns increases (i.e. the correlation decreases), Hopfield Networks are able to store a larger amount of patterns. We also observed that RBM is able to store a large amount of patterns (in comparison with Hopfield Networks), independently of the correlation between them.

On the testing process, we observed that Hopfield Networks are able to recall almost all remembered patterns, independently on the distance between the given input and the pattern. On the other hand, RBM recall performance is inversely proportional to the distance between the input and the pattern, not being able to recall it in several cases. However, a larger number of hidden neurons provides a slightly better overall performance. RBM recall performance also depends on the number of stored patterns, as more stored patterns imply a slight reduction of successful recalls.

More testing cases are still pending (e.g. several combinations of pattern properties), but due to the limited resources of our team, we could not perform a larger experiment. We assume that our conclusions are valid for any pattern dimension, but more research is needed so as to validate it.

It is also possible to tune several parameters of RBM to offer better results. The optimal tuning of RBM is far from the scope of this paper, but in the future it could be done automatically using an optimization algorithm, such as simplex or genetic algorithms. In the past years some techniques for improving Hopfield Networks appeared, [Paik and Katsaggelos (1992)] and it would be interesting to test them too.

Additionally, although Hintons guide for training RBM [Hinton (2012)] recommends the usage of a weight decay, we were unable to train any model using it, so we have not included it in this paper. It is possible that this part of our RBM implementation is not correct, causing this problem.

Taking everything into account, it would be interesting to reinforce the conclusions of this paper in a real scenario, such as classifying the MNIST database of handwritten digits. [LeCun et al. (1998)]

References

A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147 – 169, 1985. ISSN 0364-0213.

Simple encoding of infrared spectra for pattern recognition part 2. neural network approach using back-propagation and associative hopfield memory. *Analytica Chimica Acta*, 316 (2):145 – 159, 1995. ISSN 0003-2670.

Recognition of facial images with low resolution using a hopfield memory model. *Pattern Recognition*, 31(2):159 – 167, 1998. ISSN 0031-3203.

On the equivalence of hopfield networks and boltzmann machines. *Neural Networks*, 34:1 – 9, 2012. ISSN 0893-6080.

Parallel retrieval of correlated patterns: From hopfield networks to boltzmann machines. *Neural Networks*, 38:52 – 63, 2013. ISSN 0893-6080.

Miguel A Carreira-Perpinan and Geoffrey E Hinton. On contrastive divergence learning. In *AISTATS*, volume 10, pages 33–40. Citeseer, 2005.

D. O. Hebb. The organization of behavior: A neuropsychological theory. *Science Education*, 34(5):336–337, 1950. ISSN 1098-237X.

Geoffrey E. Hinton. *A Practical Guide to Training Restricted Boltzmann Machines*, pages 599–619. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012. ISBN 978-3-642-35289-8.

John J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the national academy of sciences*, 79(8):2554–2558, 1982.

Alex Krizhevsky and Geoffrey E. Hinton.

Yann LeCun, Corinna Cortes, and Christopher JC Burges. The mnist database of handwritten digits, 1998.

David J. C. MacKay. *Hopfield Networks*. Cambridge University Press, 2003. ISBN 0521642981.

K. Nagatani and M. Hagiwara. Restricted boltzmann machine associative memory. In *2014 International Joint Conference on Neural Networks (IJCNN)*, pages 3745–3750, 2014.

J. K. Paik and A. K. Katsaggelos. Image restoration using a modified hopfield network. *IEEE Transactions on Image Processing*, 1(1):49–63, 1992.

N. Niki R. Sammouda and H. Nishitani. A comparison of hopfield neural network and boltzmann machine in segmenting mr images of the brain. *IEEE Transactions on Nuclear Science*, 43(6):3361–3369, 1996.

M. Niranjan S. V. B. Aiyer and F. Fallside. A theoretical investigation into the performance of the hopfield model. *IEEE Transactions on Neural Networks*, 1(2):204–215, 1990.

P. Smolensky. Information processing in dynamical systems: Foundations of harmony theory. In D. E. Rumelhart, J. L. McClelland, et al., editors, *Parallel Distributed Processing: Volume 1: Foundations*, pages 194–281. MIT Press, Cambridge, 1987.

Amos Storkey. *Increasing the capacity of a hopfield network without sacrificing functionality*, pages 451–456. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-540-69620-9.

X. Zhuang and Y. Huang. Design of hopfield content-addressable memories. In *IEEE International Conference on Neural Networks*, pages 1069–1074 vol.2, 1993.

Appendix A. Annalyzed Pattern Data Sets

	P_1	P_2	P_3	P_4	P_5	P_6	P_7
$\|P_i\|$	5	10	15	20	25	30	35
n	100	100	100	100	100	100	100
$\frac{\mu_{P_i}}{n}$	0.20	0.20	0.20	0.20	0.20	0.20	0.20
$\frac{\sigma_{P_i}}{n}$	0.02	0.02	0.02	0.02	0.02	0.02	0.02

Table 1: Analyzed pattern data sets (1 of 4)

	P_8	P_9	P_{10}	P_{11}	P_{12}	P_{13}	P_{14}
$\|P_i\|$	5	10	15	20	25	30	35
n	100	100	100	100	100	100	100
$\frac{\mu_{P_i}}{n}$	0.30	0.30	0.30	0.30	0.30	0.30	0.30
$\frac{\sigma_{P_i}}{n}$	0.03	0.03	0.03	0.03	0.03	0.03	0.03

Table 2: Analyzed pattern data sets (2 of 4)

	P_{15}	P_{16}	P_{17}	P_{18}	P_{19}	P_{20}	P_{21}
$\|P_i\|$	5	10	15	20	25	30	35
n	100	100	100	100	100	100	100
$\frac{\mu_{P_i}}{n}$	0.40	0.40	0.40	0.40	0.40	0.40	0.40
$\frac{\sigma_{P_i}}{n}$	0.04	0.04	0.04	0.04	0.04	0.04	0.04

Table 3: Analyzed pattern data sets (3 of 4)

	P_{22}	P_{23}	P_{24}	P_{25}	P_{26}	P_{27}	P_{28}
$\|P_i\|$	5	10	15	20	25	30	35
n	100	100	100	100	100	100	100
$\frac{\mu_{P_i}}{n}$	0.50	0.50	0.50	0.50	0.50	0.50	0.50
$\frac{\sigma_{P_i}}{n}$	0.05	0.05	0.05	0.05	0.05	0.05	0.05

Table 4: Analyzed pattern data sets (4 of 4)

Appendix B. Analyzed Models

	M_1	M_2	M_3	M_4
(1)	Hopfield	Hopfield	RBM	RBM
(2)	Hebbian	Storkey	CD	CD
(3)	n/a	n/a	50	100
(4)	n/a	n/a	1	1

Table 5: Analyzed models

(1) Model

(2) Learning rule.

(3) Number of hidden neurons.

(4) Patterns per batch

Appendix C. Tested Input Data Sets

	I_1	I_2	I_3	I_4	I_5
(1)	10	10	10	10	10
$\frac{\mu_{I_i}}{\mu_{P_j}}$	0.1 ± 0.0	0.2 ± 0.0	0.3 ± 0.0	0.4 ± 0.0	0.5 ± 0.0

Table 6: Randomly generated and tested input data sets

(1) Number of generated inputs per each pattern of the pattern data set, $\frac{|I_i|}{|P_j|}$

Appendix D. Training and Validation Results

	M_1	M_2	M_3	M_4
P_1	n/a	n/a	18124 ± 7069	14541 ± 2420
P_2	n/a	n/a	29057 ± 8731	11242 ± 2781
P_3	n/a	n/a	33583 ± 11031	12570 ± 3410
P_4	n/a	n/a	28940 ± 16282	12377 ± 5766
P_5	n/a	n/a	35530 ± 14682	9834 ± 2972
P_6	n/a	n/a	25162 ± 9013	12035 ± 2854
P_7	n/a	n/a	22319 ± 8792	12006 ± 3489
P_8	n/a	n/a	31690 ± 8718	21964 ± 4678
P_9	n/a	n/a	35074 ± 8181	18915 ± 3348
P_{10}	n/a	n/a	42173 ± 21376	23287 ± 9620
P_{11}	n/a	n/a	42339 ± 25028	26494 ± 15153
P_{12}	n/a	n/a	64959 ± 39081	36511 ± 14325
P_{13}	n/a	n/a	38354 ± 22668	24570 ± 11618
P_{14}	n/a	n/a	36352 ± 23636	30642 ± 14164
P_{15}	n/a	n/a	44274 ± 14970	44308 ± 13092
P_{16}	n/a	n/a	60494 ± 12280	37955 ± 7551
P_{17}	n/a	n/a	86497 ± 46676	50112 ± 22391
P_{18}	n/a	n/a	64698 ± 33553	37774 ± 12024
P_{19}	n/a	n/a	95758 ± 71621	60481 ± 30881
P_{20}	n/a	n/a	76866 ± 60918	98700 ± 16541
P_{21}	n/a	n/a	58147 ± 41255	74267 ± 29266
P_{22}	n/a	n/a	74411 ± 38141	59317 ± 11266
P_{23}	n/a	n/a	69858 ± 33073	66048 ± 10678
P_{24}	n/a	n/a	132268 ± 57657	82277 ± 23651
P_{25}	n/a	n/a	217777 ± 271439	93176 ± 38089
P_{26}	n/a	n/a	53481 ± 55669	114696 ± 31592
P_{27}	n/a	n/a	43289 ± 35583	131611 ± 65733
P_{28}	n/a	n/a	56949 ± 69490	151782 ± 41599

Table 7: Number of epochs needed to train model M_i with pattern data set P_j

	M_1	M_2	M_3	M_4
P_1	0.00 ± 0.00	0.42 ± 0.12	1.00 ± 0.00	1.00 ± 0.00
P_2	0.00 ± 0.00	0.17 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
P_3	0.00 ± 0.00	0.07 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
P_4	0.00 ± 0.00	0.05 ± 0.00	0.87 ± 0.33	1.00 ± 0.00
P_5	0.00 ± 0.00	0.04 ± 0.00	0.95 ± 0.09	1.00 ± 0.01
P_6	0.00 ± 0.00	0.04 ± 0.01	0.99 ± 0.01	0.99 ± 0.01
P_7	0.00 ± 0.00	0.03 ± 0.01	0.80 ± 0.33	1.00 ± 0.01
P_8	0.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_9	0.00 ± 0.00	0.47 ± 0.10	1.00 ± 0.00	1.00 ± 0.00
P_{10}	0.00 ± 0.00	0.30 ± 0.06	1.00 ± 0.00	1.00 ± 0.00
P_{11}	0.00 ± 0.00	0.17 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
P_{12}	0.00 ± 0.00	0.10 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
P_{13}	0.00 ± 0.00	0.10 ± 0.02	0.98 ± 0.04	1.00 ± 0.00
P_{14}	0.00 ± 0.00	0.05 ± 0.02	1.00 ± 0.01	1.00 ± 0.00
P_{15}	0.82 ± 0.16	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{16}	0.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{17}	0.00 ± 0.00	0.93 ± 0.03	1.00 ± 0.00	1.00 ± 0.00
P_{18}	0.00 ± 0.00	0.81 ± 0.09	1.00 ± 0.00	1.00 ± 0.00
P_{19}	0.00 ± 0.00	0.61 ± 0.06	1.00 ± 0.00	1.00 ± 0.00
P_{20}	0.00 ± 0.00	0.44 ± 0.07	1.00 ± 0.00	1.00 ± 0.00
P_{21}	0.00 ± 0.00	0.25 ± 0.04	1.00 ± 0.00	1.00 ± 0.00
P_{22}	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{23}	0.99 ± 0.03	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{24}	0.72 ± 0.12	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{25}	0.42 ± 0.09	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{26}	0.08 ± 0.03	1.00 ± 0.00	1.00 ± 0.00	1.00 ± 0.00
P_{27}	0.03 ± 0.03	0.99 ± 0.02	1.00 ± 0.00	1.00 ± 0.00
P_{28}	0.01 ± 0.02	0.96 ± 0.03	1.00 ± 0.00	1.00 ± 0.00

Table 8: Number of stored patterns, proportional to $|P_i|$

Appendix E. Testing Results

E.1. Testing Results for Pattern Data Set P_1

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.370 ± 0.066	0.375 ± 0.071	0.305 ± 0.046	0.273 ± 0.042	0.258 ± 0.045
M_3	1.000 ± 0.000	0.998 ± 0.007	0.995 ± 0.009	0.948 ± 0.041	0.855 ± 0.060
M_4	1.000 ± 0.000	1.000 ± 0.000	0.998 ± 0.007	0.972 ± 0.026	0.840 ± 0.036

Table 9: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_1, proportional to $|I_i|$

E.2. Testing Results for Pattern Data Set P_2

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.164 ± 0.040	0.149 ± 0.042	0.140 ± 0.035	0.128 ± 0.025	0.115 ± 0.022
M_3	0.991 ± 0.014	0.938 ± 0.055	0.752 ± 0.101	0.490 ± 0.104	0.254 ± 0.064
M_4	0.998 ± 0.007	0.966 ± 0.017	0.801 ± 0.065	0.528 ± 0.049	0.230 ± 0.040

Table 10: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_2, proportional to $|I_i|$

E.3. Testing Results for Pattern Data Set P_3

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.068 ± 0.004	0.068 ± 0.002	0.067 ± 0.000	0.067 ± 0.000	0.067 ± 0.003
M_3	0.930 ± 0.053	0.700 ± 0.091	0.383 ± 0.085	0.162 ± 0.036	0.063 ± 0.022
M_4	0.981 ± 0.008	0.833 ± 0.067	0.494 ± 0.061	0.195 ± 0.049	0.054 ± 0.018

Table 11: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_3, proportional to $|I_i|$

E.4. Testing Results for Pattern Data Set P_4

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.050 ± 0.000	0.050 ± 0.000	0.050 ± 0.000	0.050 ± 0.000	0.050 ± 0.000
M_3	0.746 ± 0.288	0.443 ± 0.185	0.179 ± 0.077	0.053 ± 0.024	0.007 ± 0.006
M_4	0.907 ± 0.044	0.586 ± 0.065	0.219 ± 0.047	0.048 ± 0.020	0.008 ± 0.004

Table 12: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_4, proportional to $|I_i|$

E.5. Testing Results for Pattern Data Set P_5

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.040 ± 0.000	0.040 ± 0.000	0.040 ± 0.000	0.040 ± 0.001	0.040 ± 0.001
M_3	0.662 ± 0.103	0.283 ± 0.059	0.075 ± 0.017	0.018 ± 0.008	0.003 ± 0.003
M_4	0.829 ± 0.049	0.378 ± 0.074	0.087 ± 0.024	0.011 ± 0.011	0.002 ± 0.002

Table 13: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_5, proportional to $|I_i|$

E.6. Testing Results for Pattern Data Set P_6

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.033 ± 0.000	0.033 ± 0.002	0.033 ± 0.000	0.032 ± 0.002	0.032 ± 0.003
M_3	0.609 ± 0.064	0.175 ± 0.042	0.042 ± 0.013	0.006 ± 0.005	0.001 ± 0.002
M_4	0.746 ± 0.086	0.248 ± 0.056	0.035 ± 0.011	0.004 ± 0.003	0.000 ± 0.000

Table 14: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_6, proportional to $|I_i|$

E.7. Testing Results for Pattern Data Set P_7

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.032 ± 0.009	0.031 ± 0.006	0.029 ± 0.001	0.029 ± 0.000	0.026 ± 0.003
M_3	0.388 ± 0.162	0.112 ± 0.049	0.020 ± 0.011	0.003 ± 0.004	0.000 ± 0.000
M_4	0.672 ± 0.097	0.185 ± 0.041	0.018 ± 0.007	0.000 ± 0.001	0.000 ± 0.000

Table 15: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_7, proportional to $|I_i|$

E.8. Testing Results for Pattern Data Set P_8

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.998 ± 0.007	0.968 ± 0.042	0.890 ± 0.087	0.843 ± 0.087	0.735 ± 0.086
M_3	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.985 ± 0.017	0.902 ± 0.064
M_4	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.995 ± 0.009	0.950 ± 0.035

Table 16: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_8, proportional to $|I_i|$

E.9. Testing Results for Pattern Data Set P_9

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.440 ± 0.077	0.395 ± 0.072	0.345 ± 0.034	0.295 ± 0.036	0.273 ± 0.038
M_3	1.000 ± 0.000	0.996 ± 0.005	0.921 ± 0.033	0.736 ± 0.074	0.471 ± 0.073
M_4	1.000 ± 0.000	1.000 ± 0.000	0.966 ± 0.015	0.820 ± 0.039	0.474 ± 0.037

Table 17: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_9, proportional to $|I_i|$

E.10. Testing Results for Pattern Data Set P_{10}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.264 ± 0.041	0.213 ± 0.043	0.172 ± 0.027	0.132 ± 0.026	0.109 ± 0.017
M_3	0.994 ± 0.006	0.928 ± 0.027	0.709 ± 0.059	0.377 ± 0.079	0.133 ± 0.042
M_4	0.999 ± 0.002	0.978 ± 0.014	0.822 ± 0.038	0.470 ± 0.076	0.152 ± 0.033

Table 18: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{10}, proportional to $|I_i|$

E.11. Testing Results for Pattern Data Set P_{11}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.153 ± 0.030	0.127 ± 0.028	0.108 ± 0.026	0.089 ± 0.025	0.078 ± 0.023
M_3	0.981 ± 0.013	0.814 ± 0.049	0.473 ± 0.045	0.166 ± 0.024	0.037 ± 0.009
M_4	0.992 ± 0.018	0.918 ± 0.049	0.587 ± 0.084	0.210 ± 0.043	0.031 ± 0.012

Table 19: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{11}, proportional to $|I_i|$

E.12. Testing Results for Pattern Data Set P_{12}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.083 ± 0.026	0.074 ± 0.023	0.060 ± 0.017	0.049 ± 0.009	0.042 ± 0.010
M_3	0.948 ± 0.029	0.611 ± 0.071	0.232 ± 0.027	0.054 ± 0.017	0.007 ± 0.004
M_4	0.988 ± 0.008	0.820 ± 0.046	0.365 ± 0.041	0.079 ± 0.017	0.006 ± 0.002

Table 20: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{12}, proportional to $|I_i|$

E.13. Testing Results for Pattern Data Set P_{13}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.082 ± 0.013	0.074 ± 0.008	0.051 ± 0.009	0.041 ± 0.007	0.028 ± 0.009
M_3	0.823 ± 0.063	0.436 ± 0.080	0.096 ± 0.024	0.015 ± 0.010	0.003 ± 0.003
M_4	0.969 ± 0.015	0.650 ± 0.064	0.185 ± 0.028	0.023 ± 0.010	0.001 ± 0.001

Table 21: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{13}, proportional to $|I_i|$

E.14. Testing Results for Pattern Data Set P_{14}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.048 ± 0.012	0.042 ± 0.011	0.036 ± 0.009	0.028 ± 0.004	0.023 ± 0.004
M_3	0.754 ± 0.053	0.285 ± 0.052	0.051 ± 0.013	0.006 ± 0.005	0.000 ± 0.001
M_4	0.934 ± 0.032	0.479 ± 0.078	0.098 ± 0.019	0.005 ± 0.003	0.000 ± 0.001

Table 22: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{14}, proportional to $|I_i|$

E.15. Testing Results for Pattern Data Set P_{15}

	I_1	I_2	I_3	I_4	I_5
M_1	0.723 ± 0.103	0.583 ± 0.053	0.425 ± 0.060	0.272 ± 0.055	0.170 ± 0.070
M_2	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.998 ± 0.007	0.990 ± 0.014
M_3	1.000 ± 0.000	1.000 ± 0.000	0.998 ± 0.007	0.970 ± 0.032	0.912 ± 0.077
M_4	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.995 ± 0.009	0.960 ± 0.032

Table 23: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{15}, proportional to $|I_i|$

E.16. Testing Results for Pattern Data Set P_{16}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.988 ± 0.029	0.981 ± 0.033	0.970 ± 0.033	0.925 ± 0.046	0.847 ± 0.033
M_3	1.000 ± 0.000	0.994 ± 0.013	0.954 ± 0.033	0.810 ± 0.108	0.529 ± 0.110
M_4	1.000 ± 0.000	1.000 ± 0.000	0.993 ± 0.008	0.883 ± 0.061	0.631 ± 0.105

Table 24: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{16}, proportional to $|I_i|$

E.17. Testing Results for Pattern Data Set P_{17}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.912 ± 0.033	0.880 ± 0.040	0.802 ± 0.047	0.696 ± 0.058	0.537 ± 0.054
M_3	0.997 ± 0.003	0.955 ± 0.024	0.785 ± 0.084	0.466 ± 0.085	0.214 ± 0.045
M_4	1.000 ± 0.000	0.998 ± 0.005	0.943 ± 0.021	0.645 ± 0.048	0.287 ± 0.028

Table 25: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{17}, proportional to $|I_i|$

E.18. Testing Results for Pattern Data Set P_{18}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.748 ± 0.074	0.689 ± 0.068	0.571 ± 0.057	0.445 ± 0.037	0.292 ± 0.034
M_3	0.989 ± 0.009	0.868 ± 0.067	0.577 ± 0.070	0.251 ± 0.046	0.067 ± 0.025
M_4	0.999 ± 0.002	0.976 ± 0.011	0.790 ± 0.046	0.372 ± 0.038	0.090 ± 0.020

Table 26: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{18}, proportional to $|I_i|$

E.19. Testing Results for Pattern Data Set P_{19}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.518 ± 0.049	0.420 ± 0.044	0.325 ± 0.030	0.205 ± 0.039	0.112 ± 0.022
M_3	0.969 ± 0.014	0.728 ± 0.036	0.344 ± 0.054	0.099 ± 0.030	0.014 ± 0.010
M_4	0.998 ± 0.003	0.933 ± 0.026	0.607 ± 0.046	0.201 ± 0.030	0.023 ± 0.013

Table 27: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{19}, proportional to $|I_i|$

E.20. Testing Results for Pattern Data Set P_{20}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.326 ± 0.045	0.253 ± 0.027	0.173 ± 0.029	0.104 ± 0.025	0.048 ± 0.013
M_3	0.929 ± 0.028	0.559 ± 0.063	0.188 ± 0.033	0.041 ± 0.009	0.005 ± 0.004
M_4	0.996 ± 0.006	0.877 ± 0.028	0.420 ± 0.027	0.079 ± 0.016	0.005 ± 0.003

Table 28: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{20}, proportional to $|I_i|$

E.21. Testing Results for Pattern Data Set P_{21}

	I_1	I_2	I_3	I_4	I_5
M_1	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000	0.000 ± 0.000
M_2	0.204 ± 0.039	0.156 ± 0.028	0.095 ± 0.020	0.055 ± 0.015	0.023 ± 0.010
M_3	0.876 ± 0.037	0.437 ± 0.060	0.101 ± 0.029	0.011 ± 0.007	0.001 ± 0.002
M_4	0.987 ± 0.008	0.742 ± 0.038	0.258 ± 0.034	0.029 ± 0.008	0.002 ± 0.002

Table 29: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{21}, proportional to $|I_i|$

E.22. Testing Results for Pattern Data Set P_{22}

	I_1	I_2	I_3	I_4	I_5
M_1	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.988 ± 0.010
M_2	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.995 ± 0.009
M_3	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.998 ± 0.007	0.950 ± 0.020
M_4	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.970 ± 0.026

Table 30: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{22}, proportional to $|I_i|$

E.23. Testing Results for Pattern Data Set P_{23}

	I_1	I_2	I_3	I_4	I_5
M_1	0.968 ± 0.045	0.953 ± 0.065	0.926 ± 0.088	0.884 ± 0.089	0.802 ± 0.129
M_2	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.994 ± 0.007
M_3	1.000 ± 0.000	0.998 ± 0.004	0.981 ± 0.013	0.881 ± 0.061	0.635 ± 0.056
M_4	1.000 ± 0.000	1.000 ± 0.000	0.996 ± 0.005	0.924 ± 0.044	0.726 ± 0.067

Table 31: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{23}, proportional to $|I_i|$

E.24. Testing Results for Pattern Data Set P_{24}

	I_1	I_2	I_3	I_4	I_5
M_1	0.648 ± 0.102	0.603 ± 0.122	0.537 ± 0.110	0.439 ± 0.117	0.307 ± 0.087
M_2	1.000 ± 0.000	1.000 ± 0.000	1.000 ± 0.000	0.991 ± 0.007	0.924 ± 0.018
M_3	1.000 ± 0.000	0.947 ± 0.027	0.791 ± 0.062	0.481 ± 0.095	0.220 ± 0.049
M_4	1.000 ± 0.000	0.997 ± 0.005	0.939 ± 0.031	0.708 ± 0.058	0.338 ± 0.053

Table 32: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{24}, proportional to $|I_i|$

E.25. Testing Results for Pattern Data Set P_{25}

	I_1	I_2	I_3	I_4	I_5
M_1	0.339 ± 0.083	0.268 ± 0.070	0.192 ± 0.058	0.134 ± 0.035	0.064 ± 0.024
M_2	1.000 ± 0.000	0.999 ± 0.002	0.994 ± 0.008	0.926 ± 0.013	0.761 ± 0.036
M_3	0.995 ± 0.004	0.902 ± 0.053	0.639 ± 0.097	0.297 ± 0.055	0.089 ± 0.030
M_4	0.999 ± 0.002	0.984 ± 0.012	0.839 ± 0.057	0.480 ± 0.072	0.161 ± 0.027

Table 33: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{25}, proportional to $|I_i|$

E.26. Testing Results for Pattern Data Set P_{26}

	I_1	I_2	I_3	I_4	I_5
M_1	0.042 ± 0.025	0.025 ± 0.018	0.009 ± 0.008	0.008 ± 0.006	0.000 ± 0.000
M_2	0.998 ± 0.003	0.988 ± 0.011	0.933 ± 0.011	0.748 ± 0.031	0.488 ± 0.026
M_3	0.972 ± 0.025	0.759 ± 0.070	0.389 ± 0.079	0.123 ± 0.049	0.024 ± 0.010
M_4	0.998 ± 0.004	0.952 ± 0.026	0.675 ± 0.093	0.285 ± 0.063	0.055 ± 0.022

Table 34: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{26}, proportional to $|I_i|$

E.27. Testing Results for Pattern Data Set P_{27}

	I_1	I_2	I_3	I_4	I_5
M_1	0.011 ± 0.013	0.007 ± 0.010	0.003 ± 0.004	0.001 ± 0.002	0.000 ± 0.000
M_2	0.970 ± 0.020	0.912 ± 0.023	0.772 ± 0.027	0.519 ± 0.038	0.275 ± 0.040
M_3	0.925 ± 0.022	0.611 ± 0.071	0.223 ± 0.049	0.053 ± 0.015	0.006 ± 0.004
M_4	0.999 ± 0.002	0.890 ± 0.034	0.477 ± 0.056	0.127 ± 0.020	0.014 ± 0.006

Table 35: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{27}, proportional to $|I_i|$

E.28. Testing Results for Pattern Data Set P_{28}

	I_1	I_2	I_3	I_4	I_5
M_1	0.005 ± 0.007	0.001 ± 0.001	0.001 ± 0.002	0.000 ± 0.000	0.000 ± 0.000
M_2	0.914 ± 0.031	0.798 ± 0.016	0.572 ± 0.014	0.310 ± 0.032	0.124 ± 0.022
M_3	0.874 ± 0.048	0.475 ± 0.060	0.152 ± 0.023	0.020 ± 0.006	0.003 ± 0.003
M_4	0.991 ± 0.006	0.789 ± 0.052	0.336 ± 0.055	0.050 ± 0.012	0.005 ± 0.003

Table 36: Number of successful recalls when input data set I_i is given to model M_j, trained with pattern data set P_{28}, proportional to $|I_i|$

Appendix F. Reproducibility

The source code of this project is public on github[2]. The default configured random seed produces the same exact results as the ones published in this paper.

F.1. Requirements

- Python $\geq$ 2.7.0
- NumPy $\geq$ 1.12.0
- UNIX-like system shell.

F.2. Execution

Just execute the following command in the root of the project to generate data sets and test the models:

```
$ sh generate_results.sh
```

This will create several JSON files in `out/results/`. When it finishes, execute the following command:

```
$ sh generate_latex.sh > out/tables.tex
```

This will create a TeX file in `out/tables.tex` containing the result tables used in this paper.

2. https://github.com/juanlao7/bblr-hopfield-boltzmann