Autocorrelation in Sparse Distributed Memory

Marcelo S. Brogliato, *Member, IEEE,* and Alexandre Linhares, *Member, IEEE,*

***Abstract*—Sparse Distributed Memory (SDM) is a neuroscientific and psychologically plausible model of human memory. In this paper we present an anomaly between its previously theoretized behavior and its actual behavior, and demonstrate that this anomaly is due to the autocorrelation involved in the model. Our findings have systemwide implications for those willing to implement and apply SDM in computational intelligence settings.**

***Index Terms*—Sparse Distributed Memory, Autocorrelation, Theoretical Neuroscience, Memory, Systemwide properties.**

I. Introduction

SPARSE DISTRIBUTED MEMORY (SDM) [6] (see also [1, 2, 9, 8, 11, 13, 14, 12, 10, 23, 24]) is a mathematical model of long-term memory that has a number of neuroscientific and psychologically plausible dynamics. This model is used in all sort of applications due to its incredible ability to closely reflect the human capacity to remember past experiences from the subtlest of clues. Applications range from call admission control [16, 15], to behavior-based robotics [21, 19, 5], to noise filtering [20], etc.

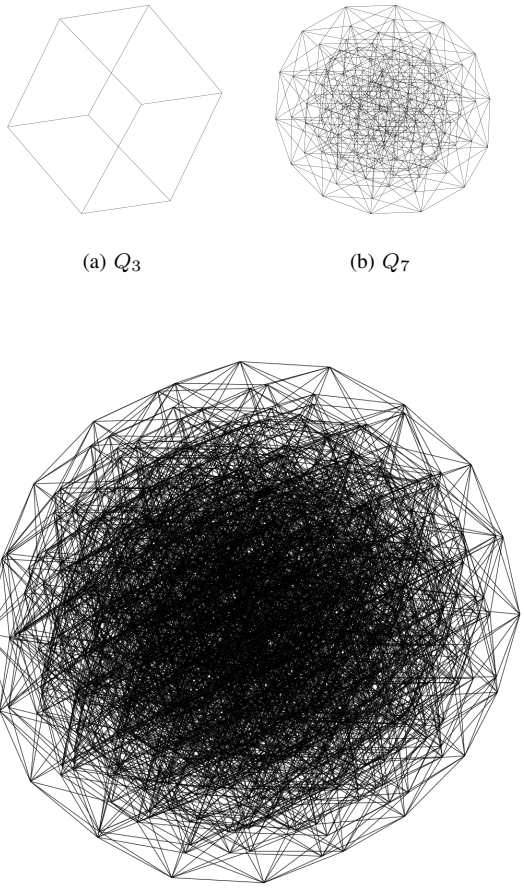
A. Review of the model

IN SDM, the data — and address space on which it is stored — are represented by large *bitstrings*. The *Hamming distance* provides comparisons between bitstrings and is used as a metric for the system. This space is also called the *hypercube graph*, or Q_n, as in Figure 1. For a fixed $n \in \mathbb{Z}$, the graph $G = (V, E)$, in which $v \in V$ iff there is a bijective function $b : V \to \{0,1\}^n$ and $(v_i, v_j) \in E$ iff $H(b(v_i), b(v_j)) = 1$, where H is the Hamming distance. That is, n-sized bitstrings correspond to nodes, and edges exist between nodes iff they flip a single bit. Though Kanerva has derived many combinatorial properties of the space, additional results have been found by the graph-theoretical community: it is a bipartite graph with chromatic number 2, and it is planar only if $n \leq 3$. The proofs, though elementary, illuminate some of its properties: vertices connect bitstrings with an even number of 1's with bitstrings with an odd number of 1's, therefore (i) it is *bipartite* and (ii) it *may be colored with two colors*. As for planarity, the case $n = 3$ is the largest planar one. A good survey is provided by [4] — and some interesting results may be found in [3, 25, 17, 22].

Dr. Brogliato is with sdm.ai and with Behavioral and Decision Sciences, EBAPE, Fundação Getulio Vargas, Rio de Janeiro, Brazil, e-mail: msbrogli@sdm.ai.

Dr. Linhares is with sdm.ai and with Behavioral and Decision Sciences, EBAPE, Fundação Getulio Vargas, Rio de Janeiro, Brazil, e-mail: linhares@sdm.ai.

0000–0000/00$00.00 © 2012 IEEE



(a) Q_3 (b) Q_7

(c) Q_{10}

Fig. 1. Kanerva models the space of *possible incoming neural signals* as an n-dimensional *hypercube*, or Q_n. Here we have Q_n, for $n \in \{3, 7, 10\}$. Each node corresponds to a bitstring in $\{0,1\}^n$, and two nodes are linked iff the bitstrings differ by a single dimension. A number of observations can be made here. First, the number of nodes grows as $O(2^n)$; which makes the space rapidly intractable. Another interesting observation is that most of the space lies 'at the core' of the hypercube, at a distance of around $n/2$ from any given vantage point.

Unlike traditional memory used by computers, SDM performs read and write operations in a multitude of addresses, also called neurons. That is, the data is not written, or it is not read in a single address spot, but in many addresses. These are called activated addresses, or activated neurons.

The activation of addresses takes place according to their distances from the datum. Suppose one is writing datum η at address ξ, then all addresses inside a circle with center ξ and radius r are activated. So, η will be stored in all these activated addresses, which are around address ξ, such as in

Figure 2. An address ξ' is inside the circle if its Hamming distance to the center ξ is less than or equal to the radius r, i.e. $distance(\xi, \xi') \leq r$.

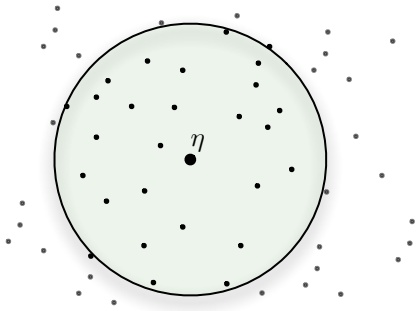


Fig. 2. Activated addresses inside accessradius r around the center address.

Every write or read in SDM memory activates a number of addresses with close distance. The data is written in these activated addresses or read from them. These issues will be addressed in due detail further on, but a major difference from a traditional computer memory is that the data are always stored and retrieved in a multitude of addresses. This way SDM memory has robustness against loss of addresses (e.g., death of a neuron).

In traditional memory, each datum is stored in an address and every lookup of a specific datum requires a search through the memory. In spite of computer scientists having developed beautiful algorithms to perform fast searches, almost all of them do a precise search. That is, if you have an imprecise clue of what you need, these algorithms will simply fail.

In SDM, the data space is the same as the address space, which amounts to a vectorial, binary space, that is, a $\{0,1\}^n$ space. This way, the addresses where the data will be written are the same as the data themselves. For example, the datum $\eta = 00101_b \in \{0,1\}^5$ will be written to the address $\xi = \eta = 00101_b$. If one chooses a radius of 1, the SDM will activate all addresses one bit away or less from the center address. So, the datum 00101_b will be written to the addresses 00101_b, 10101_b, 01101_b, 00001_b, 00111_b, and 00100_b.

In this case, when one needs to retrieve the data, one could have an imprecise cue at most one bit away from η, since all addresses one bit away have η stored in themselves. Extending this train of thought for larger dimensions and radius, exponential numbers of addresses are activated and one can see why SDM is a distributed memory.

When reading a cue η_x that is x bits away from η, the cue shares many addresses with η. The number of shared addresses decreases as the cue's distance to η increases, in other words, as x increases. This is shown in Figure 3. The target datum η was written in all shared addresses, thus they will bias the read output in the direction of η. If the cue is sufficiently near the target datum η, the read output will be closer to η than η_x was. Repeating the read operation increasingly gets results closer to η, until it is precisely the same. So, it may be necessary to perform more than one read operation to converge to the target data η.

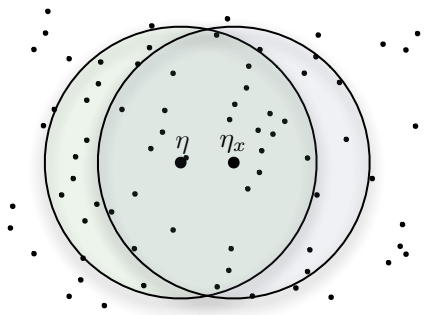


Fig. 3. Shared addresses between thetarget datum η and the cue η_x.

The addresses of the $\{0,1\}^n$ space grow exponentially with the number of dimensions n, i.e., $N = 2^n$. For $n = 100$ we have $N \approx 10^{30}$, which is incredibly large when related to computer memory. Furthermore, [6] suggests n between 100 and 10,000. To solve the feasibility problem of implementing this memory, Kanerva made a random sample of $\{0,1\}^n$, in his work, having N' elements. All these addresses in the sample are called hard locations. Other elements of $\{0,1\}^n$, not in N', are called virtual neurons. This is represented in Figure 4. All properties of reading and write operations presented before remain valid but limited to hard locations. Kanerva suggests taking a sample of about one million hard locations.

Using this sample of binary space, our data space does not exist completely. That is, the binary space has 2^n addresses, but the memory is far away from having these addresses available. In fact, only a fraction of this vectorial space is actually instantiated. Following Kanerva's suggestion of one million hard locations, for $n = 100$, only $100 \cdot 10^6/2^{100} = 7 \cdot 10^{-23}$ percent of the whole space exists, and for $n = 1,000$ only $100 \cdot 10^6/2^{1000} = 7 \cdot 10^{-294}$ percent.

Kanerva also suggests the selection of a radius that will activate, on average, one-thousandth of the sample, which is 1,000 hard locations for a sample of one million addresses. In order to achieve his suggestion, a 1,000-dimension memory uses an access radius $r = 451$, and a 256-dimensional memory, $r = 103$. We think that a 256-dimensional memory may be important because it presents conformity to Miller's magic number [18].

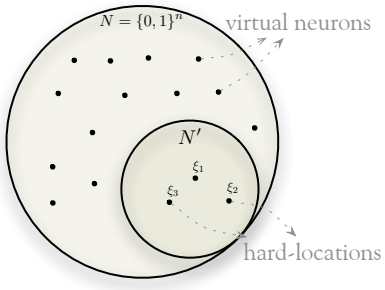


Fig. 4. Hard-locations randomly sampled from binary space.

Since a cue η_x near the target bitstring η shares many hard locations with η, SDM can retrieve data from imprecise cues. Despite this feature, it is very important to know how imprecise this cue could be while still giving accurate results. What is the maximum distance from our cue to the original data that still retrieves the right answer? An interesting approach is to perform a read operation with a cue η_x, that is x bits away from the target η. Then measure the distance from the read output and η. If this distance is smaller than x we are converging. Convergence is simple to handle, just read again and again, until it converges to the target η. If this distance is greater than x we are diverging. Finally, if this distance equals x we are in a tip-of-the-tongue process. A tip-of-the-tongue psychologically happens when you know that you know, but you can't say what exactly it is. In SDM mathematical model, a tip-of-the-tongue process takes infinite time to converge. **(author?)** [6] called this x distance, where the read's output averages x, the critical distance. Intuitively, it is the distance from which smaller distances converge and greater distances diverge. In Figure 5, the circle has radius equal to the critical distance and every η_x inside the circle should converge. The figure also shows convergence in four readings.

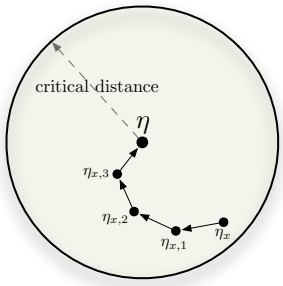


Fig. 5. In this example, four iterative readings were required to converge from η_x to η.

The $\{0,1\}^n$ space has $N = 2^n$ locations from which we instantiate N' samples. Each location in our sample is called a hard location. On these hard locations, we do operations of read and write. One of the insights of SDM is exactly the way we read and write: using data as addresses in a distributed fashion. Each datum η is written in every activated hard location inside the access radius centered on the address, that equals datum, $\xi = \eta$. Kanerva suggested using an access radius r having about one-thousandth of N'. As an imprecise cue η_x shares hard locations with the target bitstring η, it is possible to retrieve η correctly. (Actually, probably more than one read is necessary to retrieve exactly η.). Moreover, if some neurons are lost, only a fraction of the datum is lost and it is possible that the memory can still retrieve the right datum.

A random bitstring is generated with equal probability of 0's and 1's for each bit. One can readily see that the average distance between two random bitstrings follows the binomial distribution with mean $n/2$ and standard deviation $\sqrt{n/4}$. For a large n, most of the space lies close to the mean and has fewer shared hard locations. As two bitstrings with distance far from $n/2$ are very improbable, **(author?)** [6] defined that two bitstrings are orthogonal when their distance is $n/2$.

The write operation needs to store, for each dimension bit which happened more (0's or 1's). This way, each hard location has n counters, one for each dimension. The counter is incremented for each bit 1 and decremented for each bit 0. Thus, if the counter is positive, there have been more 1's than 0's, if the counter is negative, there have been more 0's than 1's, and if the counter is zero, there have been an equal number of 1's and 0's.

The read is performed polling each activated hard location and statistically choosing the most written bit for each dimension. It consists of adding all n counters from the activated hard locations and, for each bit, choosing bit 1 if the counter is positive, choosing bit 0 if the counter is negative, and randomly choose bit 0 or 1 if the counter is zero.

II. Neurons as pointers

One interesting view is that neurons in SDM work like pointers. As we write bitstrings in memory, the hard locations' counters are updated and some bits are flipped. Thus, the activated hard locations do not necessarily point individually to the bitstring that activated it, but together they point correctly. In other words, the read operation depends on many hard locations to be successful. This effect is represented in Figure 6: where all hard locations inside the circle are activated and they, individually, do not point to η. But, like vectors, adding them up points to η. If another datum ν is written into the memory near η, the shared hard locations will have information from both of them and would not point to either. All hard locations outside of the circle are also pointing somewhere (possibly other data points). This is not shown, however, in order to keep the picture clean and easily understandable.

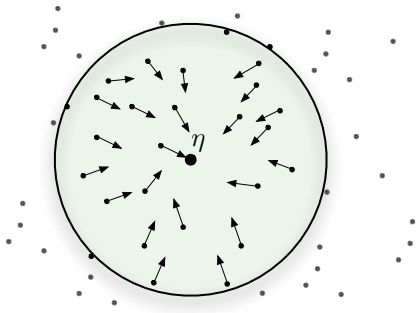


Fig. 6. Hard-locations pointing, approximately, to the target bitstring.

III. Read operation

In his work, Kanerva proposed and analyzed a read algorithm called here Kanerva's read. His read takes all activated hard locations counters and sum them. The resulting bitstring has bit 1 where the result is positive, bit 0 where the result is negative, and a random bit where the result is zero. In a word, each bit is chosen according to all written bitstrings in all hard locations, being equal to the bit more appeared. Table **??** shows an example of Kanerva's read result bitstring.

In this other paragraph we explain what the equator distance is.

Then, this introductory section breaks and moves on to the main findings of this paper.

IV. A DEVIATION FROM THE EQUATOR DISTANCE?

(author?) [7]

Kanerva writes[1]:

> You have done an incredibly thorough analysis of SDM. I like the puzzle in your message and believe that your simulations are correct and to be learned from. So what to make of the difference compared to my Figure 7.3 (and your Figure **??**)? I think the difference comes from my not having accounted fully for the effect of the other 9,999 vectors that are stored in the memory. You say in it
>
> "Our results show that the theoretical prediction is not accurate. There are interaction effects from one or more of the attractors created by the 10,000 writes, and these attractors seem to raise the distance beyond 500 bits (Figure **??**)."
>
> I think that is correct. It also brings to mind a comment Louis Jaeckel made when we worked at NASA Ames. He pointed out that autoassociative storage (each vector is stored with itself as the address) introduces autocorrelation that my formula for Figure 7.2 did not take into account. When we read from memory, each stored vector exerts a pull toward itself, which also means that each bit of a retrieved vector is slightly biased toward the same bit of the read address, regardless of the read address. We never worked out the math.

This is an important observation. A hard location is activated because it shares many dimensions with the items read from or written onto it. Imagine the 'counter's eye view': each individual counter 'likes' to write on its own corresponding bit-address value more than it likes the opposite; as each hard-location has a say in its own area — and nowhere else.

Let x and y be random bitstrings and n be the number of dimensions in the memory; let x_i and y_i be the i-th bit of x and y, respectively; and $d(x, y)$ be the Hamming distance. Whilst the probability of a shared bit-value between same dimension-bits in two random addresses is $1/2$, an address only activates hard-locations close to it. Let us call these shared bitvalues a *bitmatch in dimension* i.

So, what is the probability of bitmatches given that we know the access radius r between the address and a hard-location?

Theorem 1. Each dimension i has a small pull bias, which can be measured by $P(x_i = y_i | d(x, y) \leq r) = \frac{\sum_{k=0}^{r} \binom{n-1}{k}}{\sum_{k=0}^{r} \binom{n}{k}}$.

[1]Email thread 'SDM: A puzzling issue and an invitation', started March 16th 2018, in which we discussed the aforementioned discrepancy. To think that some centuries ago, all scientific publishing was the exchange of such letters.

Proof. The left-hand expression $P(x_i = y_i | d(x, y) \leq r)$ computes the probability of a bitmatch in i, given that we know that x and y are in the access radius defined by r, i.e., $d(x, y) \leq r$.

Applying the law of total probability to the left-hand expression we obtain

$$\sum_{k=0}^{r} P(x_i = y_i | d(x, y) = k \leq r) P(d(x, y) = k | d(x, y) \leq r) \tag{1}$$

We also know that

$$P(x_i = y_i | d(x, y) = k) = \frac{n - k}{n} \tag{2}$$

$$P(d(x, y) = k | d(x, y) \leq r) = \frac{\binom{n}{k}}{\sum_{j=0}^{r} \binom{n}{j}} \tag{3}$$

Hence,

$$P(x_i = y_i | d(x, y) \leq r) = \frac{\sum_{k=0}^{r} \frac{n-k}{n} \binom{n}{k}}{\sum_{j=0}^{r} \binom{n}{j}} \tag{4}$$

Finally, the combinatorial identity

$$\frac{n - k}{n} \binom{n}{k} = \binom{n - 1}{k} \tag{5}$$

closes the theorem.

□

Theorem 1 is valid for both "x written at x" (autoassociative memory) and "random written at x" (heteroassociative memory). When $n = 1,000$ and $r = 451$, $P(x_i = y_i | d(x, y) \leq r) = p = 0.552905$. Each bit of a hard location does indeed have a small pull bias. What is meant by this is that each particular dimension has a small preference toward positive values if its address bit is set to 1, and negative values if set to 0—an intuition developed in Figure 7.

Lemma 2. *Let r be the access radius given that f percent of the hard locations are activated. Then,* $\lim_{n \to \infty} r/n = 1/2$.

Proof. As the bits of the hard locations' addresses are randomly chosen, the distance between two hard locations follow a Binomial distribution with n samples and probability 0.5, $B(n, 0.5)$. For n sufficiently large, the Binomial distribution can be approximated by a Normal distribution, i.e., $B(n, 0.5) \to \mathcal{N}(\mu = n/2, \sigma^2 = n/4)$.

Let $\Phi(x)$ be the cdf of the standard normal distribution. Let $z = \frac{r - n/2}{\sqrt{n}/2}$. Thus, $P(d(x, y) \leq r) = \Phi(z)$. As $f = P(d(x, y) \leq r)$, then, $f = \Phi(z)$.

Calculating the inverse, $z = \Phi^{-1}(f)$. Then,

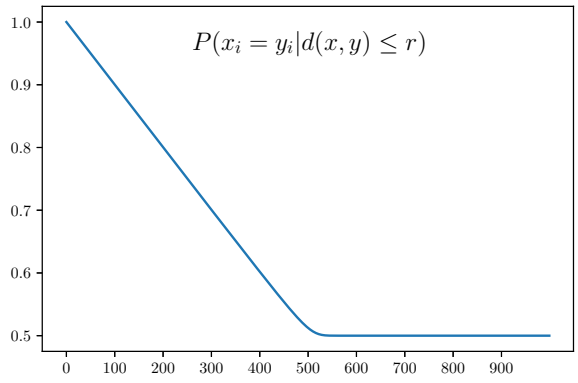


Fig. 7. The threshold size r of hard-locations bring about the autocorrelation of Theorem 1. Consider, for instance, the extremes $\{0, n\}$. Let us start with n (in our case, $n = 1000$ dimensions): given the information that $d(x, y) \leq n$, the probability of a bitmatch in dimension i is $1/2$; as $\forall x, y \in \{0, 1\}^N, d(x, y) \leq n$. At the other extreme, consider that we have the information that $d(x, y) \leq 0$: in this case $x = y$ and the probability of a bitmatch, in any dimension, is 1. The autocorrelation hence drops monotonically until convergence at $1/2$ as the distance grows. Numerically, our results converge to precisely $1/2$ only after $d(x, y) \geq 607$ for $n = 1000$; $d(x, y) \geq 5332$ for $n = 10000$, and $d(x, y) \geq 83$ for $n = 100$. This will be seen in Lemma 2.

$$z = \Phi^{-1}(f) \tag{6}$$

$$\frac{r - n/2}{\sqrt{n}/2} = \Phi^{-1}(f) \tag{7}$$

$$r = \frac{n}{2} + \Phi^{-1}(f)\frac{\sqrt{n}}{2} \tag{8}$$

$$\frac{r}{n} = \frac{1}{2} + \Phi^{-1}(f)\frac{1}{2\sqrt{n}} \tag{9}$$

Therefore, $n \to \infty \Rightarrow r/n \to 1/2$. □

Lemma 3. *Let $\Phi(x)$ be the cdf of the standard normal distribution. Then, $n \to \infty \Rightarrow P(x_i = y_i | d(x, y) \leq r) = \frac{1}{2}\frac{\Phi(z_1)}{\Phi(z_2)}$, where $z_1 = \frac{2r-n+1}{\sqrt{n-1}}$ and $z_2 = \frac{2r-n}{\sqrt{n}}$.*

Proof. From the approximation of the Binomial distribution $B(a, 0.5)$ by the Normal distribution $\mathcal{N}(\mu = a/2, \sigma^2 = a/4)$, we conclude that, for a sufficiently large, the cdf of the Binomial is approximately equal to the cdf of the Normal distribution. Thus,

$$\frac{1}{2^a}\sum_{k=0}^{b}\binom{a}{k} = \Phi\left(\frac{b - a/2}{\sqrt{a}/2}\right) = \Phi\left(\frac{2b - a}{\sqrt{a}}\right)$$

Thus,

$$\sum_{k=0}^{b}\binom{a}{k} = 2^a\Phi\left(\frac{2b - a}{\sqrt{a}}\right)$$

The result comes directly from applying the equation above in $P(x_i = y_i | d(x, y) \leq r)$.

□

Theorem 4. *The autocorrelation vanishes when $n \to \infty$, i.e.,* $\lim_{n\to\infty} P(x_i = y_i | d(x, y) \leq r) = 1/2$.

Proof. From Lemma 2, we know that $r = n/2$ for n sufficiently large. Thus, replacing $r = n/2$ in Lemma 3, $P(x_i = y_i | d(x, y) \leq r) = \frac{\Phi(z_1)}{2\Phi(z_2)}$, where $z_1 = \frac{1}{\sqrt{n-1}}$ and $z_2 = 0$.

As $n \to \infty$, $z_1 \to 0$, and $P(x_i = y_i | d(x, y) \leq r) = \frac{\Phi(0)}{2\Phi(0)} = 1/2$. □

Another way to prove is to divide into two cases:

Suppose that n is an even integer, then,

$$\sum_{k=0}^{r}\binom{n}{k} = \sum_{k=0}^{n/2}\binom{n}{k} = \frac{1}{2}\sum_{k=0}^{n}\binom{n}{k} = \frac{2^n}{2} = 2^{n-1}$$

And, also,

$$\begin{aligned}\sum_{k=0}^{r}\binom{n-1}{k} &= \sum_{k=0}^{n/2}\binom{n-1}{k} \\ &= \frac{1}{2}\left[\sum_{k=0}^{n-1}\binom{n-1}{k} - \binom{n-1}{n/2}\right] \\ &= \frac{1}{2}\left[2^{n-1} - \binom{n-1}{n/2}\right] \\ &= 2^{n-2} - \frac{1}{2}\binom{n-1}{n/2}\end{aligned}$$

Finally,

$$P(x_i = y_i | d(x, y) \leq r) = \frac{\sum_{k=0}^{r}\binom{n-1}{k}}{\sum_{k=0}^{r}\binom{n}{k}} \tag{10}$$

$$= \frac{2^{n-2} - \frac{1}{2}\binom{n-1}{n/2}}{2^{n-1}} \tag{11}$$

$$= \frac{2^{n-2}}{2^{n-1}} - \frac{1}{2^n}\binom{n-1}{n/2} \tag{12}$$

$$= \frac{1}{2} - \frac{1}{2^n}\binom{n-1}{n/2} \tag{13}$$

Stirling's approximation yields that, for n sufficiently large, $\binom{n}{n/2} \sim \frac{2^n}{\sqrt{\pi n/2}}$. Thus, $\frac{1}{2^n}\binom{n}{n/2} \sim \frac{\sqrt{2}}{\sqrt{\pi n}}$, which yields $\lim_{n\to\infty}\frac{1}{2^n}\binom{n}{n/2} = 0$. Finally, as $\binom{n-1}{n/2} \leq \binom{n}{n/2}$, by the squeeze theorem, $\lim_{n\to\infty}\frac{1}{2^n}\binom{n-1}{n/2} = 0$, which closes the proof for n even.

When n is an odd integer, the steps of the proof are similar. Therefore, the proof is complete.

□

In Figure 8, with $n = 10,000$ and $r = 4,845$, we can notice that the autocorrelation has reduced significantly as predicted by Theorem 4. In fact, in this case, using Lemma 3, $P(x_i = y_i | d(x, y) \leq r) = 0.516876$.

So far we have looked only at a single pair of bitstrings, the probability of a single bitmatch between bitstrings within the access radius distance. Now let us consider the number of activated hard locations exhibiting this bitmatch.

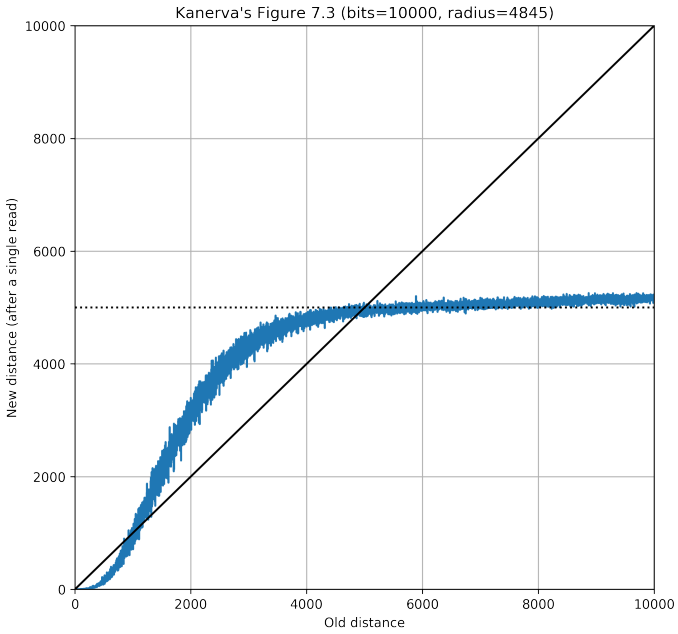


Fig. 8. The same setup as in Figure **??**, but for $n = 10,000$. It shows that the interference has almost gone away when n is sufficiently large.

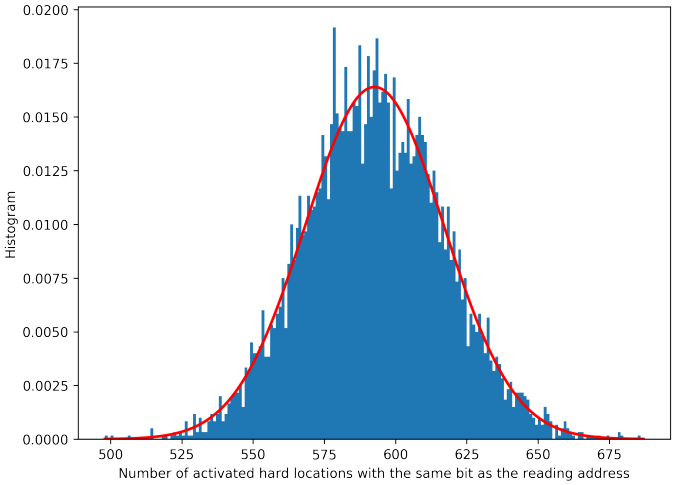


Fig. 9. Given an address x and a dimension i, how many hard locations with bitmatches in i are activated by reading at x? The histogram was obtained through numerical simulation. The red curve is the theoretical normal distribution found in Theorem 5.

Let h be the number of activated hard locations. As the probability of activating a specific hard location is a constant $h \sim \text{Binomial}(H, p_1)$. Thus, $\mathbf{E}[h] = \mu_h = Hp_1$ and $\mathbf{V}[h] = \sigma_h^2 = Hp_1(1-p_1)$, where $p_1 = 2^{-n}\sum_{k=0}^{r}\binom{n}{k}$.

Let Z be the number of activated hard locations with the same bit as the reading address. Then, $Z = \sum_{i=1}^{h} X_i$, where $X_i \sim \text{Bernoulli}(p)$, where $p = P(x_i = y_i | d(x,y) \leq r)$.

Theorem 5. *Given a reading address x and a dimension i, the number of activated hard-locations with bitmatches at i follows a normal distribution with $\mathbf{E}[Z] = \mu_Z = p\mu_h$ and $\mathbf{V}[Z] = \sigma_Z^2 = p(1-p)\mu_h + p^2\sigma_h^2$.*

Proof. By the central limit theorem, Z is normally distributed.

Applying the law of total averages and the law of total variance, $\mathbf{E}[Z] = \mathbf{E}[\mathbf{E}[Z|h]] = \mathbf{E}[ph] = p\mathbf{E}[h] = ph$, and $\mathbf{V}[Z] = \mathbf{E}[\mathbf{V}[Z|h]] + \mathbf{V}[\mathbf{E}[Z|h]] = \mathbf{E}[hp(1-p)] + \mathbf{V}[ph] = p(1-p)\mathbf{E}[h] + p^2\mathbf{V}[h] = hp(1-p) + p^2Hp_1(1-p_1)$.

Applying the law of total variance, $\mathbf{V}[Z] = \mathbf{E}[\mathbf{V}[Z|h]] + \mathbf{V}[\mathbf{E}[Z|h]] = \mathbf{E}[hp(1-p)] + \mathbf{V}[ph] = p(1-p)\mathbf{E}[h] + p^2\mathbf{V}[h] = p(1-p)\mu_h + p^2\sigma_h^2$. □

As, in our case, $P(973 < h < 1170) = 0.997$, by the central limit theorem, Z may be approximated by a normal distribution.

See Figure 9 for a comparison between the theoretical model and a simulation.

V. Counter bias

The previous theorems show that there is bias in the counters. In this section we show that there is a small counter bias, as a function of the bitmatch in dimension i. If the Hard location's address bit is set to 1, the mean will be above zero. And the mean will be below zero if the address bit is set to 0.

Let us analyze the ith counter of a hard location. Let s be the number of bitstrings written into memory (in our case, $s = 10,000$) and addr_i be the ith bit of the hard location's address.

Let θ be the average number of bitstrings written in each hard location. As there are s bitstrings written into the memory, and the probability of activating a specific hard location is constant, $\theta \sim \text{Binomial}(s, p_1)$. Thus, $\mathbf{E}[\theta] = \mu_\theta = sp_1$ and $\mathbf{V}[\theta] = \sigma_\theta^2 = sp_1(1-p_1)$.

Let us introduce another random variable. Let Y_i be the number of bitmatches in the ith dimension of a hard location's address after s written bitstrings. Then, $Y_i = \sum_{k=1}^{\theta} X_k$.

Theorem 6. *Given the number of written bitstrings s, $\mathbf{E}[Y_i] = \mu_Y = p\mu_\theta$ and $\mathbf{V}[Y_i] = \sigma_Y^2 = p(1-p)\mu_\theta + p^2\sigma_\theta^2$.*

Proof. Applying the law of total expectation, $\mathbf{E}[Y] = \mathbf{E}[\mathbf{E}[Y|\theta]] = \mathbf{E}[p\theta] = p\mathbf{E}[\theta] = p\mu_\theta$.

Applying the law of total variance, $\mathbf{V}[Y] = \mathbf{E}[\mathbf{V}[Y|\theta]] + \mathbf{V}[\mathbf{E}[Y|\theta]] = \mathbf{E}[\theta p(1-p)] + \mathbf{V}[p\theta] = p(1-p)\mathbf{E}[\theta] + p^2\mathbf{V}[\theta] = p(1-p)\mu_\theta + p^2\sigma_\theta^2$. □

During a write operation, the counters are incremented for every bit 1 and decremented for every bit 0. So, after s writes, there will be θ bitstrings written in each hard location with Y_i bitmatches and $\theta - Y_i$ non-bitmatches. Thus, $[\text{cnt}_i|\text{addr}_i = 1] = (Y_i) - (\theta - Y_i) = 2Y_i - \theta$ and $[\text{cnt}_i|\text{addr}_i = 0] = \theta - 2Y_i$.

Theorem 7. $\mathbf{E}[cnt_i|addr_i = 1] = \mu_{cnt} = (2p-1)\mu_\theta$ *and* $\mathbf{V}[cnt_i|addr_i = 1] = \sigma_{cnt}^2 = 4p(1-p)\mu_\theta + (2p-1)^2\sigma_\theta^2$.

Proof. $\mathbf{E}[\text{cnt}_i|\text{addr}_i = 1] = \mathbf{E}[2Y_i - \theta] = \mathbf{E}[2Y_i] - \mathbf{E}[\theta] = 2\mathbf{E}[Y_i] - \mu_\theta = 2p\mu_\theta - \mu_\theta = (2p-1)\mu_\theta$.

Applying the law of total variance, $\mathbf{V}[\text{cnt}_i|\text{addr}_i = 1] = \mathbf{V}[2Y_i - \theta] = \mathbf{E}[\mathbf{V}[2Y_i - \theta|\theta]] + \mathbf{V}[\mathbf{E}[2Y_i - \theta|\theta]]$.

Let us solve each part independently. Thus,

$\mathbf{V}[2Y_i - \theta|\theta] = \mathbf{V}[2Y_i|\theta] = 4\mathbf{V}[Y_i|\theta] = 4\mathbf{V}[\sum_{k=1}^{\theta} X_k] = 4\theta p(1-p)$.

$\mathbf{E}[\mathbf{V}[2Y_i-\theta|\theta]] = \mathbf{E}[4\theta p(1-p)] = 4p(1-p)\mathbf{E}[\theta] = 4p(1-p)\mu_\theta$.

Finally,

$\mathbf{E}[2Y_i - \theta|\theta] = 2\mathbf{E}[Y_i|\theta] - \mathbf{E}[\theta|\theta] = 2p\theta - \theta = (2p-1)\theta$.

$\mathbf{V}[\mathbf{E}[2Y_i - \theta|\theta]] = \mathbf{V}[(2p-1)\theta] = (2p-1)^2\mathbf{V}[\theta] = (2p-1)^2\sigma_\theta^2$. □

Theorem 8. $\mathbf{E}[cnt_i|addr_i = 0] = -\mu_{cnt}$ *and* $\mathbf{V}[cnt_i|addr_i = 1] = \sigma_{cnt}^2$.

Proof. Notice that $[\text{cnt}_i|\text{addr}_i = 0] = -[\text{cnt}_i|\text{addr}_i = 1]$. Thus, $\mathbf{E}[\text{cnt}_i|\text{addr}_i = 0] = -\mathbf{E}[\text{cnt}_i|\text{addr}_i = 1]$ and $\mathbf{V}[\text{cnt}_i|\text{addr}_i = 0] = \mathbf{V}[\text{cnt}_i|\text{addr}_i = 1]$. □

In summary, what we learn is that

$$[\text{cnt}_i|\text{addr}_i = 1] \sim \mathcal{N}(\mu_{\text{cnt}}, \sigma_{\text{cnt}}^2) \tag{14}$$

$$[\text{cnt}_i|\text{addr}_i = 0] \sim \mathcal{N}(-\mu_{\text{cnt}}, \sigma_{\text{cnt}}^2) \tag{15}$$

In our case, $p = 0.5529$, $s = 10,000$, and $H = 1,000,000$, so $[\text{cnt}_i|\text{addr}_i = 1] \sim \mathcal{N}(\mu = 1.1341, \sigma^2 = 10.7184)$. For "random at x", $p = 0.5$, so $\mu = 0$ and $\sigma^2 = 10.7185$. The slight bias above or below 0 can be seen in Figure 10.

Finally,

$$P(\text{cnt}_i \geq 0|\text{addr}_i = 1) = P(\text{cnt}_i \leq 0|\text{addr}_i = 0) \tag{16}$$

$$= 1 - \mathcal{N}.\text{cdf}(0) \tag{17}$$

For "random written at x", $p = 0.5$ implies $\mu_{\text{cnt}} = 0$, which implies $P(\text{cnt}_i \geq 0|\text{addr}_i = 1) = P(\text{cnt}_i \leq 0|\text{addr}_i = 0) = 0.5$, independently of the parameters because they will only affect the variance and the normal distribution is symmetrical around the average.

However, for "x written at x", $p = 0.5529$ and the probabilities depend on s. For $s = 10,000$, they are equal to 0.6354. For $s = 20,000$, they are equal to 0.6867. For $s = 30,000$, they are equal to 0.7232. The more random bitstrings are written into the memory, the more the hard locations point to themselves.

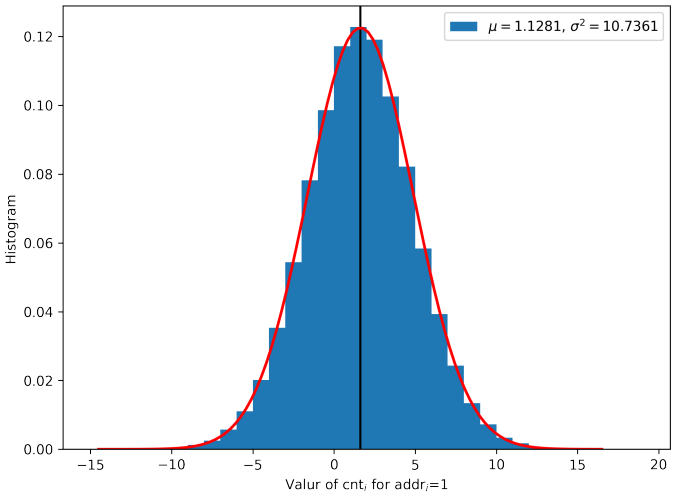
Let D be the number of counters aligned with addr_i. The standard deviation was calculated using the fact that $[D|\theta] \sim \text{Binomial}(1000, q = P(\text{cnt}_i > 0|\text{addr}_i = 1, \theta))$.

Applying the law of total variance, $\mathbf{V}[D] = \mathbf{E}[\mathbf{V}[D|\theta]] + \mathbf{V}[\mathbf{E}[D|\theta]] = \mathbf{E}[1000q(1-q)] + \mathbf{V}[1000q] = 1000\mathbf{E}[q - q^2] + 1000^2\mathbf{V}[q] = 1000\mathbf{E}[q](1-\mathbf{E}[q]) + 1000(1000-1)\mathbf{V}[q]$, where $\mathbf{E}[q] = \sum_\theta P(\text{cnt}_i > 0|\text{addr}_i = 1, \theta)P(\theta)$ and $\mathbf{E}[q^2] = \sum_\theta [P(\text{cnt}_i > 0|\text{addr}_i = 1, \theta)]^2 P(\theta)$.

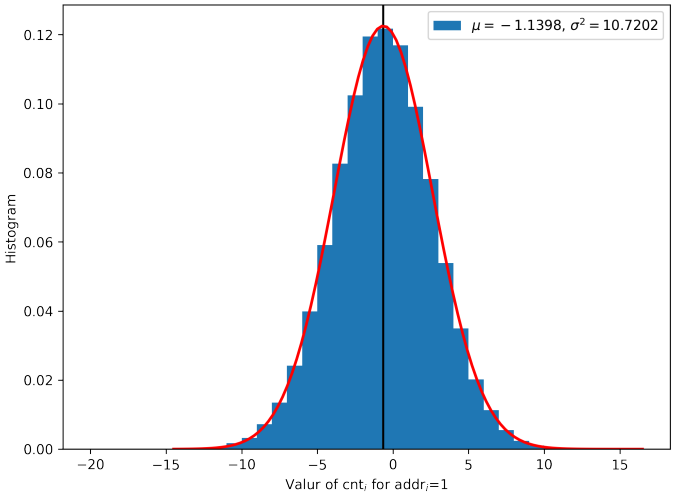
Doing the math, $\mathbf{E}[q] = 0.402922$ and $\mathbf{E}[q^2] = 0.634433$. Thus, $\mathbf{V}[q] = \mathbf{E}[q^2] - (\mathbf{E}[q])^2 = 0.0004166$. Hence, $\mathbf{V}[D] = 648.2041$. See Figure 11 and notice that I still have to figure out why the mean is correct, but the standard deviation is not.

VI. Read Bias

Now that we know the distribution of $\text{cnt}_i|\text{addr}_i$, we may go to the read operation. During the read operation, on average, h hard locations are activated and their counters are summed up. So, for the ith bit,



(a) $\text{addr}_i = 1$



(b) $\text{addr}_i = 0$

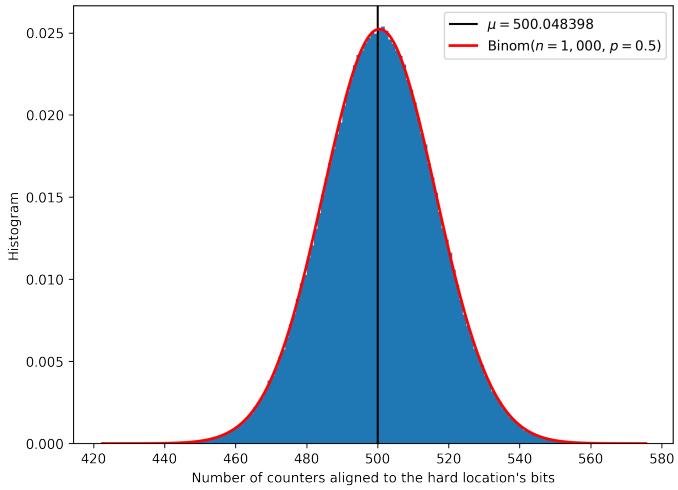
Fig. 10. The value of the counters after $s = 10,000$ writes shows the autocorrelation in the counters in autoassociative memories ("x at x"). The histogram was obtained through simulation. The red curve is the theoretical normal distribution found in equations (14) and (15).

$$\text{acc}_i = \sum_{k=1}^{h} \text{cnt}_k \tag{18}$$

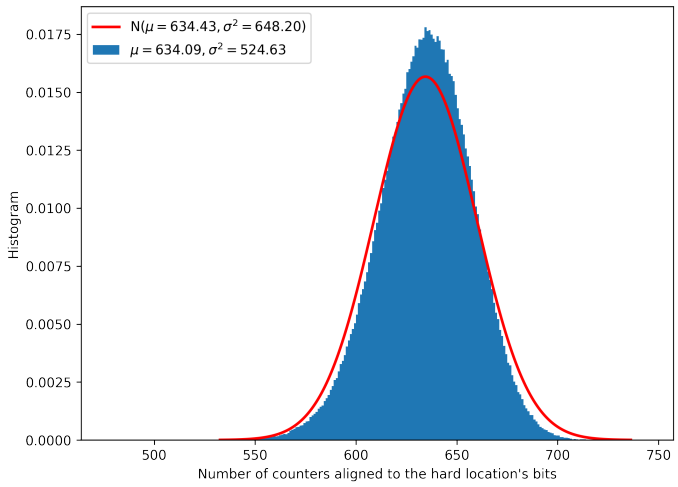
Let η be the reading address and η_i the ith bit of it. Then, let's split the h activated hard locations into two groups: (i) the ones with the same bit as η_i with Z hard locations, and (ii) the ones with the opposite bit as η_i with $h - Z$ hard locations.

$$[\text{acc}_i|\eta_i] = \sum_{k=1}^{Z} [\text{cnt}_k|\text{addr}_k = \eta_i] + \sum_{k=1}^{h-Z} [\text{cnt}_k|\text{addr}_k \neq \eta_i] \tag{19}$$

Each sum is a sum of normally distributed random variables, so

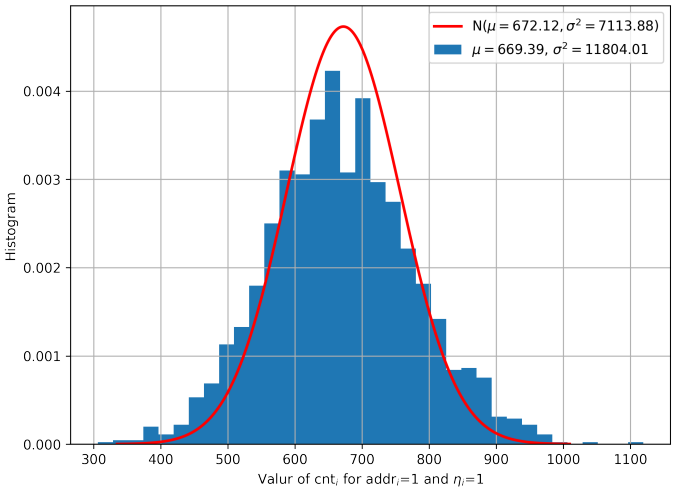


(a) "random at x"

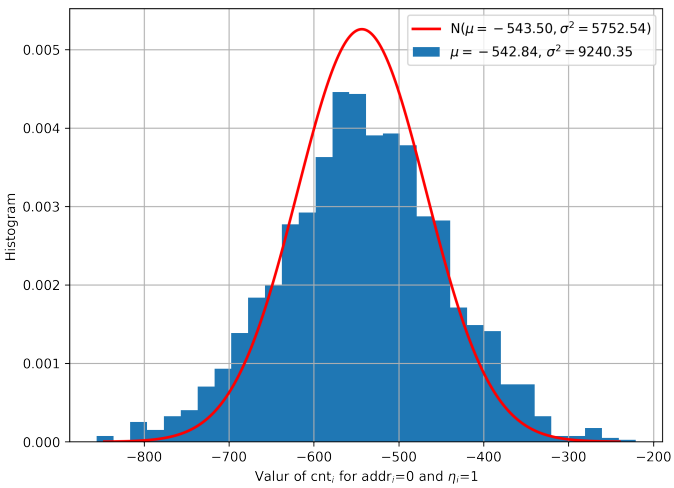


(b) "x at x"

Fig. 11. Autocorrelation in the counters in autoassociative memories ("x written at x"). The histogram was obtained through simulation. The red curve is the theoretical distribution.



(a) Equation 20 ($\text{addr}_k = 1$)



(b) Equation 21 ($\text{addr}_k = 0$)

Fig. 12. The histogram was obtained through simulation. The red curve is the theoretical normal distribution.

$$\sum_{k=1}^{Z} [\text{cnt}_k | \text{addr}_k = \eta_1] \sim \mathcal{N}(\mu = \mu_{\text{cnt}}\mu_Z, \sigma^2 = \sigma^2_{\text{cnt}}\mu_Z + \mu^2_{\text{cnt}}\sigma^2_Z) \tag{20}$$

$$\sum_{k=1}^{h-Z} [\text{cnt}_k | \text{addr}_k \neq \eta_1] \sim \mathcal{N}(\mu = -\mu_{\text{cnt}}(1-p)\mu_h, \sigma^2 = \sigma^2_{\text{cnt}}(1-p)\mu_h + \mu^2_{\text{cnt}}\sigma^2_{h-Z}) \tag{21}$$

In our case, $\sum_{k=1}^{Z} [\text{cnt}_k | \text{addr}_k = 1] \sim \mathcal{N}(\mu = 672.12, \sigma^2 = 7113.87)$, and $\sum_{k=1}^{Z} [\text{cnt}_k | \text{addr}_k = 1] \sim \mathcal{N}(\mu = -543.49, \sigma^2 = 5752.54)$. See Figure 12 — we can notice that the average is correct but the variance is too small.

Hence,

$$[\text{acc}_i | \eta_i = 1] \sim \mathcal{N}(\mu = (2p-1)^2\mu_\theta\mu_h, \sigma^2 = \sigma^2_{\text{cnt}}\mu_h + 2\mu^2_{\text{cnt}}\sigma^2_h) \tag{22}$$

$$[\text{acc}_i | \eta_i = 0] \sim \mathcal{N}(\mu = -(2p-1)^2\mu_\theta\mu_h, \sigma^2 = \sigma^2_{\text{cnt}}\mu_h + 2\mu^2_{\text{cnt}}\sigma^2_h) \tag{23}$$

In our case, $[\text{acc}_i | \eta_i = 1] \sim \mathcal{N}(\mu = 128.62, \sigma^2 = 12865.69)$, and $[\text{acc}_i | \eta_i = 0] \sim \mathcal{N}(\mu = -128.62, \sigma^2 = 12865.69)$. See Figure 13 — we can notice that the variance issue from Figure 12 has propagated to these images.

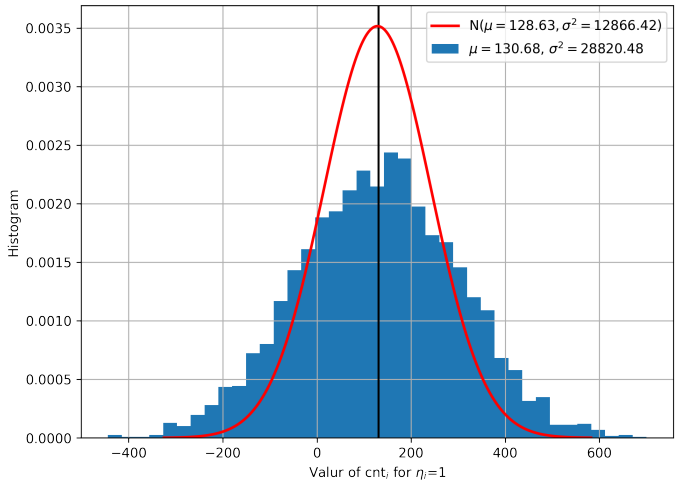
Finally,

$$P(wrong) = P(\text{acc}_i < 0 | \eta_i = 1) \cdot P(\eta_i = 1) + P(\text{acc}_i > 0 | \eta_i = 0) \cdot P(\eta_i = 0) \tag{24}$$

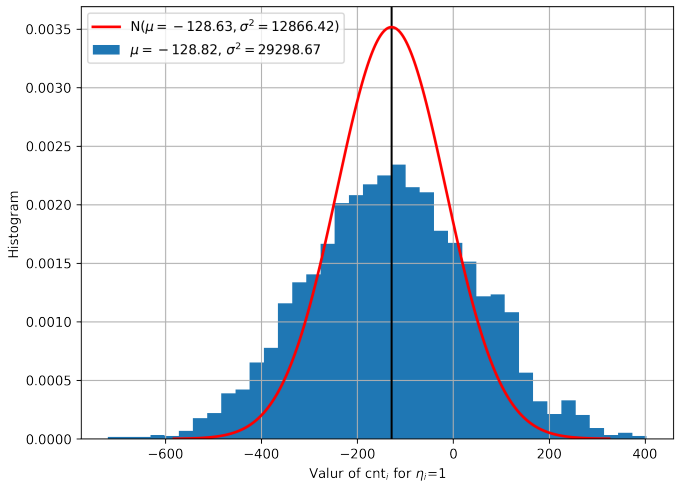
$$= \frac{\mathcal{N}_{\eta_i=1}.\text{cdf}(0)}{2} + \frac{1 - \mathcal{N}_{\eta_i=0}.\text{cdf}(0)}{2} \tag{25}$$

$$= \frac{\mathcal{N}_{\eta_i=1}.\text{cdf}(0)}{2} + \frac{\mathcal{N}_{\eta_i=1}.\text{cdf}(0)}{2} \tag{26}$$

$$= \mathcal{N}_{\eta_i=1}.\text{cdf}(0) \tag{27}$$



(a) Equation 22 ($\eta_k = 1$)



(b) Equation 23 ($\eta_k = 0$)

Fig. 13. The histogram was obtained through simulation. The red curve is the theoretical normal distribution.

Using the empirical variance of $\sigma^2 = 27838.3029$, we calculate $P(wrong) = 0.220377$.

In order to check this probability, I have run a simulation reading from 1,000 random bitstrings (which have never been written into memory) and calculate the distance from the result of a single read. As the $P(wrong) = 0.22037$, I expected to get an average distance of 220.37 with a standard deviation of 13.10. See Figure 14 for the comparison between the simulated and the theoretical outcomes.

Figure 15 shows the new distance between η_d and read(η_d), where η_d is d bits away from η. As for $d \geq 520$ there is no intersection between η and η_d, our models applies and explains the horizontal line around distance 220.

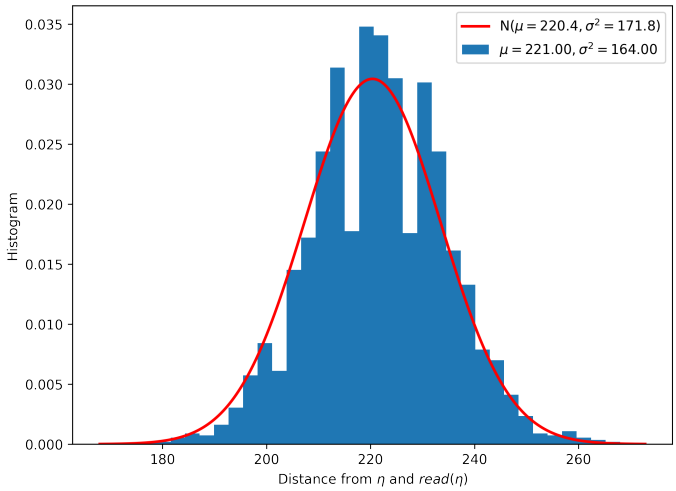


Fig. 14. The histogram was obtained through simulation. The red curve is the theoretical normal distribution.

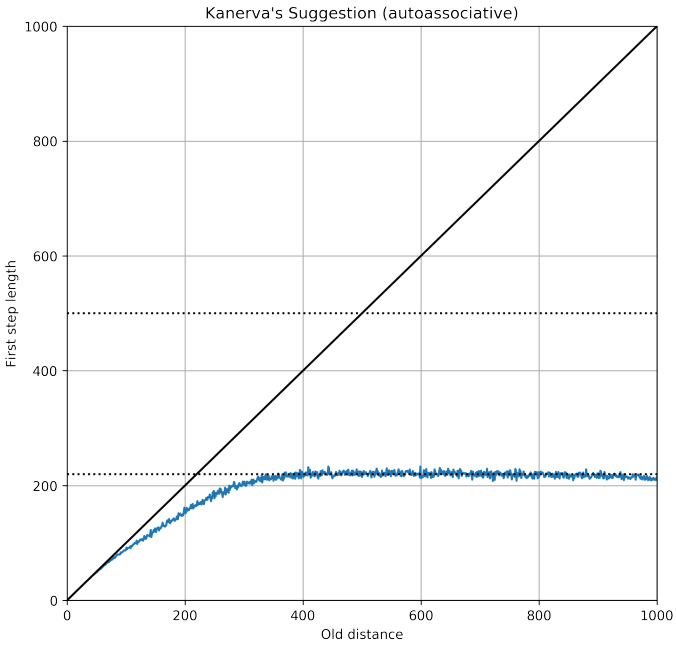


Fig. 15. New distance after a single read operation in a bitstring η_d, which is d bits away from η. The new distance was calculated between η_d and read(η_d). Notice that when $d \geq 520$, the intersection between η and η_d is zero, which means there is only random bitstrings written into the activated hard locations. The distance 220 equals $1000 \cdot 0.220$ which is the probability find in Figure 14.

VII. Conclusion

A. *Effects on system-wide properties*

The skeptical reader may ask: 'Beyond theory; what are the consequences of this work? How does it apply, if at all, to building systems that incorporate SDM?'

Our conclusion is that this work is of special relevance as it brings about some *systemwide properties* that models need to take into account. The first decision choice involved in building a system with SDM regards the size of n, the number of dimensions of the memory. Here, complementing the work of XXXXXXXX, we show that as n grows, autocorrelation tends to 0. But recall that as n grows, the size of the critical distance also tends to 0. It is therefore crucial to select a value for n in which the model will retain its convergence properties, with the understanding of the possible effects brought about by autocorrelation. Systemwide properties such as these affect

the entire behavior of the model and, as such, should always be taken into account by theoreticians and modelers alike.

B. Future Research

BPs, System1 vs System2, HRRs, etc.

C. Availability of the Code

OpenCL for the partial binomail sum (cite 'Inference-Based Similarity Search in Randomized Montgomery Domains for Privacy-Preserving Biometric Identification') python (show example with figure) Technical Documentation

Open-source Book

another article with a larger overview of the developed system so far (Reinforcement learning, critical distance, noise filtering and prototype theory, and pattern recognition.) This –> transparency –> replicability -> incremental improvement –> parallel explorations by different teams around the world.

ACKNOWLEGMENT

The authors would like to thank Pentti Kanerva, Eric Paul Nichols, José Ricardo de Almeida Torreão, Moacyr Alvim Horta Barbosa da Silva, Flavio Codeço Coelho and Paulo Murilo Castro de Oliveira for their careful, comprehensive, reviews of the first author's theses.

REFERENCES

[1] Peter J. Denning. Sparse Distributed Memory. *American Scientist*, 77:333–335, July 1989.

[2] M J Flynn, P Kanerva, and N Bhadkamkar. Sparse Distributed Memory: Principles and Operation. Technical Report CSL-TR-89-400, NASA Ames Research Center, 1989.

[3] S Foldes. A characterization of hypercubes. *Discrete Mathematics*, 17(1):155–159, 1977.

[4] Frank Harary, John P Hayes, and Horng-Jyh Wu. A survey of the theory of hypercube graphs. *Computers & Mathematics with Applications*, 15(4):277–289, 1988.

[5] S. Jockel, F. Lindner, and Jianwei Zhang. Sparse distributed memory for experience-based robot manipulation. In *Proceedings of the 2008 IEEE International Conference on Robotics and Biomimetics*, pages 1298–1303. IEEE, February 2009.

[6] P. Kanerva. *Sparse Distributed Memory*. MIT Press, 1988.

[7] P. Kanerva. The spatter code for encoding concepts at many levels. In M. Marinaro and P.G. Morasso, editors, *ICANN '94, Proceedings of International Conference on Artificial Neural Networks*, volume 1, pages 226–229, London, 1994. Springer-Verlag.

[8] Pentti Kanerva. Parallel Structures in Human and Computer Memory. *Cognitiva*, 85, June 1985.

[9] Pentti Kanerva. Sparse Distributed Memory and Related Models. In *Associative Neural Memories: Theory and Implementation*, page 41. Oxford University Press, 1993.

[10] Pentti Kanerva. The Spatter Code for Encoding Concepts at Many Levels. In Maria Marinaro and Pietro G. Morasso, editors, *ICANN '94*, pages 226–229. Springer London, London, 1994.

[11] Pentti Kanerva. Binary spatter-coding of ordered K-tuples. In Gerhard Goos, Juris Hartmanis, Jan Leeuwen, Christoph Malsburg, Werner Seelen, Jan C. Vorbrüggen, and Bernhard Sendhoff, editors, *Artificial Neural Networks — ICANN 96*, volume 1112, pages 869–873. Springer Berlin Heidelberg, Berlin, Heidelberg, 1996.

[12] Pentti Kanerva. Fully Distributed Representation. In *In Proceedings RWC Symposium*, pages 358–365, 1997.

[13] Pentti Kanerva. Hyperdimensional computing: An introduction to computing in distributed representation with high-dimensional random vectors. *Cognitive Computation*, 1(2):139–159, 2009.

[14] James D. Keeler. Comparison between Kanerva's SDM and Hopfield-type neural networks. *Cognitive Science*, 12(3):299–329, 1988.

[15] Hee-Yong Kwon. ATM call admission control using sparse distributed memory. In *Neural Networks, 1997., International Conference on*, volume 2, pages 1321–1325. IEEE, 1997.

[16] Hee-Yong Kwon, Dong-Keyu Kim, Seung-Jun Song, Je-U. CHoi, In-Heang Lee, and Hee-Yeung Hwang. ATM call admission control using sparse distributed memory. II. In *Neural Networks Proceedings, 1998. IEEE World Congress on Computational Intelligence. The 1998 IEEE International Joint Conference on*, volume 3, pages 1799–1803. IEEE, 1998.

[17] Jean-Marie Laborde and Surya Prakash Rao Hebbare. Another characterization of hypercubes. *Discrete Mathematics*, 39(2):161–166, January 1982.

[18] Alexandre Linhares, Daniel M. Chada, and Christian N. Aranha. The emergence of Miller's Magic Number on a Sparse Distributed Memory. *PLOS One*, 6(1):e15592, Jan 2011.

[19] Mateus Mendes, Manuel Crisóstomo, and A Paulo Coimbra. Robot navigation using a sparse distributed memory. In *Robotics and automation, 2008. ICRA 2008. IEEE international conference on*, pages 53–58. IEEE, 2008.

[20] Hongying Meng, Kofi Appiah, Andrew Hunter, Shigang Yue, Mervyn Hobden, Nigel Priestley, Peter Hobden, and Cy Pettit. A modified sparse distributed memory model for extracting clean patterns from noisy inputs. In *Proceedings of International Joint Conference on Neural Networks*, pages 2084–2089. IEEE, June 2009.

[21] Rajesh Rao and Olac Fuentes. Hierarchical learning of navigational behaviors in an autonomous robot using a predictive sparse distributed memory. *Autonomous Robots*, pages 297–316, 1998.

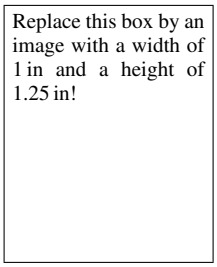
[22] Frank Ruskey. *Combinatorial Generation*. University of Victoria, working version (1j-csc 425/520) edition, 2003.

[23] Magnus Sahlgren and Pentti Kanerva. Permutations as a Means to Encode Order in Word Space. page 6.

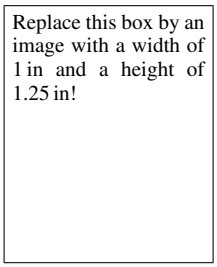
[24] Yoshinori Uesaka, Pentti Kanerva, and Hideki Asoh, editors. *Foundations of real-world intelligence*. Number no. 125 in CSLI lecture notes. CSLI Publications, Stanford, Calif, 2001.

[25] A. Wagner and D. Corneil. Embedding Trees in a Hy-

percube is NP-Complete. *SIAM Journal on Computing*, 19(3):570–590, June 1990.



Marcelo Salhab Brogliato received the Ph.D. degree from Getulio Vargas Foundation in 2018. Vialink, sdm.ai,



Alexandre Linhares received the Ph.D. degree from the National Institute of Space Research, 2001. He is an Associate Professor at the Center of Behavioral Research, EBAPE / Getulio Vargas Foundation. He has published circa 30 papers in journals such as *Artificial Intelligence, Biological Cybernetics, Cognitive Science, IEEE Transactions on Evolutionary Computation, Information Sciences, Frontiers in Psychology,* etc.