WIKIPEDIA

Stochastic computing

Stochastic computing is a collection of techniques that represent continuous values by streams of random bits. Complex computations can then be computed by simple bit-wise operations on the streams. Stochastic computing is distinct from the study of randomized algorithms.

Contents

Motivation and a simple example

Suppose that $p, q \in [0,1]$ is given, and we wish to compute $p \times q$. Stochastic computing performs this operation using probability instead of arithmetic.

Specifically, suppose that there are two random, independent bit streams called *stochastic number*s (i.e. Bernoulli processes), where the probability of a one in the first stream is p, and the probability in the second stream is q. We can take the logical AND of the two streams.

a_i	1	0	1	1	0	1	...
b_i	1	1	0	1	1	0	...
$a_i \wedge b_i$	1	0	0	1	0	0	...

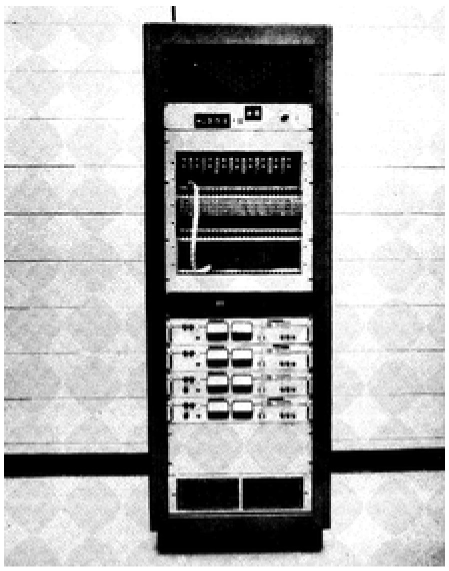
The probability of a one in the output stream is pq. By observing enough output bits and measuring the frequency of ones, it is possible to estimate pq to arbitrary accuracy.

The operation above converts a fairly complicated computation (multiplication of p and q) into a series of very simple operations (evaluation of $a_i \wedge b_i$) on random bits.

More generally speaking, stochastic computing represents numbers as streams of random bits and reconstructs numbers by calculating frequencies. The computations are performed on the streams and translate complicated operations on p and q into simple operations on their stream representations. (Because of the method of reconstruction, devices that perform these operations are sometimes called stochastic averaging processors.) In modern terms, stochastic computing can be viewed as an interpretation of calculations in probabilistic terms, which are then evaluated with a Gibbs sampler. It can also be interpreted as a hybrid analog/digital computer.

History

Stochastic computing was first introduced in a pioneering paper by John von Neumann in 1953.[1] However, the theory could not be fully developed until advances in computing of the 1960s,[2] [3] mostly through a series of simultaneous and parallel efforts in the US[4] and the UK.[5] By the late 1960s, attention turned to the design of special-purpose hardware to perform stochastic computation. A host[6] of these machines were constructed between 1969 and 1974; RASCEL[7] is pictured in this article.


The RASCEL stochastic computer, circa 1969

Despite the intense interest in the 1960s and 1970s, stochastic computing ultimately failed to compete with more traditional digital logic, for reasons outlined below. The first (and last) International Symposium on Stochastic Computing[8] took place in 1978; active research in the area dwindled over the next few years.

Although stochastic computing declined as a general method of computing, it has shown promise in several applications. Research has traditionally focused on certain tasks in machine learning and control.[9] [10] Somewhat recently, interest has turned towards stochastic decoding, which applies stochastic computing to the decoding of error correcting codes.[11] More recently, stochastic circuits have been successfully used in image processing tasks such as edge detection [12] and image thresholding.[13]

Strengths and weaknesses

Although stochastic computing was a historical failure, it may still remain relevant for solving certain problems. To understand when it remains relevant, it is useful to compare stochastic computing with more traditional methods of digital computing.

Strengths

Suppose we wish to multiply two numbers each with n bits of precision. Using the typical long multiplication method, we need to perform n^2 operations. With stochastic computing, we can AND together any number of bits and the expected value will always be correct. (However, with a small number of samples the variance will render the actual result highly inaccurate).

Moreover, the underlying operations in a digital multiplier are full adders, whereas a stochastic computer only requires an AND gate. Additionally, a digital multiplier would naively require $2n$ input wires, whereas a stochastic multiplier would only require 2 input wires. (If the digital multiplier serialized its output, however, it would also require only 2 input wires.)

Additionally, stochastic computing is robust against noise; if a few bits in a stream are flipped, those errors will have no significant impact on the solution.

Furthermore, stochastic computing elements can tolerate skew in the arrival time of the inputs. Circuits work properly even when the inputs are misaligned temporally. As a result, stochastic systems can be designed to work with inexpensive locally generated clocks instead of using a global clock and an expensive clock distribution network.[14]

Finally, stochastic computing provides an estimate of the solution that grows more accurate as we extend the bit stream. In particular, it provides a rough estimate very rapidly. This property is usually referred to as *progressive precision*, which suggests that the precision of stochastic numbers (bit streams) increases as computation proceeds. [15] It is as if the most significant bits of the number arrive before its least significant bits; unlike the conventional arithmetic circuits where the most significant bits usually arrive last. In some iterative systems the partial solutions obtained through progressive precision can provide faster feedback than through traditional computing methods, leading to faster convergence.

Weaknesses

Stochastic computing is, by its very nature, random. When we examine a random bit stream and try to reconstruct the underlying value, the effective precision can be measured by the variance of our sample. In the example above, the digital multiplier computes a number to $2n$ bits of accuracy, so the precision is 2^{-2n}. If we are using a random bit stream to estimate a number and want the standard deviation of our estimate of the solution to be at least 2^{-2n}, we would need $O(2^{4n})$ samples. This represents an exponential increase in work. In certain applications, however, the progressive precision property of stochastic computing can be exploited to compensate this exponential loss.

Second, stochastic computing requires a method of generating random biased bit streams. In practice, these streams are generated with pseudo-random number generators. Unfortunately, generating (pseudo-)random bits is fairly costly (compared to the expense of, e.g., a full adder). Therefore, the gate-level advantage of stochastic computing is typically lost.

Third, the analysis of stochastic computing assumes that the bit streams are independent (uncorrelated). If this assumption does not hold, stochastic computing can fail dramatically. For instance, if we try to compute p^2 by multiplying a bit stream for p by itself, the process fails: since $a_i \wedge a_i = a_i$, the stochastic computation would yield $p \times p = p$, which is not generally true (unless $p =$0 or 1). In systems with feedback, the problem of decorrelation can manifest in more complicated ways. Systems of stochastic processors are prone to *latching*, where feedback between different components can achieve a deadlocked state.[16] A great deal of effort must be spent decorrelating the system to attempt to remediate latching.

Fourth, although some digital functions have very simple stochastic counterparts (such as the translation between multiplication and the AND gate), many do not. Trying to express these functions stochastically may cause various pathologies. For instance, stochastic decoding requires the computation of the function $f(p,q) \to pq/(pq + (1-p)(1-q))$. There is no single bit operation that can compute this function; the usual solution involves producing correlated output bits, which, as we have seen above, can cause a host of problems.

Other functions (such as the averaging operator $f(p,q) \to (p+q)/2$) require either stream decimation or inflation. Tradeoffs between precision and memory can be challenging.

Stochastic decoding

Although stochastic computing has a number of defects when considered as a method of general computation, there are certain applications that highlight its strengths. One notable case occurs in the decoding of certain error correcting codes.

In developments unrelated to stochastic computing, highly effective methods of decoding LDPC codes using the belief propagation algorithm were developed. Belief propagation in this context involves iteratively reestimating certain parameters using two basic operations (essentially, a probabilistic XOR operation and an averaging operation).

In 2003, researchers realized that these two operations could be modeled very simply with stochastic computing.[17] Moreover, since the belief propagation algorithm is iterative, stochastic computing provides partial solutions that may lead to faster convergence. Hardware implementations of stochastic decoders have been built on FPGAs. [18] The proponents of these methods argue that the performance of stochastic decoding is competitive with digital alternatives.

Variants of stochastic computing

There are a number of variants of the basic stochastic computing paradigm. Further information can be found in the referenced book by Mars and Poppelbaum.

Bundle Processing involves sending a fixed number of bits instead of a stream. One of the advantages of this approach is that the precision is improved. To see why, suppose we transmit s bits. In regular stochastic computing, we can represent a precision of roughly $O(1/\sqrt{s})$ different values, because of the variance of the estimate. In bundle processing, we can represent a precision of $1/s$. However, bundle processing retains the same robustness to error of regular stochastic processing.

Ergodic Processing involves sending a stream of bundles, which captures the benefits of regular stochastic and bundle processing.

Burst Processing encodes a number by a higher base increasing stream. For instance, we would encode 4.3 with ten decimal digits as

$$4444444555$$

since the average value of the preceding stream is 4.3. This representation offers various advantages: there is no randomization since the numbers appear in increasing order, so the PRNG issues are avoided, but many of the advantages of stochastic computing are retained (such as partial estimates of the solution). Additionally, it retains the linear precision of bundle and ergodic processing.

References

1. von Neumann, J. (1963). "Probabilistic logics and the synthesis of reliable organisms from unreliable components" (https://archive.org/details/incompletenesspr00gold). *The Collected Works of John von Neumann*. Macmillan. ISBN 978-0-393-05169-8.
2. Petrovic, R.; Siljak, D. (1962). "Multiplication by means of coincidence". *ACTES Proc. of 3rd Int. Analog Comp. Meeting*.
3. Afuso, C. (1964), *Quart. Tech. Prog. Rept.*, Department of Computer Science, University of Illinois, Urbana, Illinois
4. Poppelbaum, W.; Afuso, C.; Esch, J. (1967). "Stochastic computing elements and systems". *Afips Fjcc*. **31**: 635–644.
5. Gaines, B. (1967). "Stochastic Computing". *Afips Sjcc*. **30**: 149–156.
6. Mars, P.; Poppelbaum, W. (1981). *Stochastic and deterministic averaging processors*.
7. Esch, J. (1969). *RASCEL, a programmable analog computer based on a regular array of stochastic computing element logic* (Thesis). University of Illinois, Urbana, Illinois.
8. *Proceedings of the first International Symposium on Stochastic Computing and its Applications*. Toulouse, France. 1978.
9. Gaines, B. R. (1969). "Stochastic Computing Systems". In Tou, Julius (ed.). *Advances in Information Systems Science*. **2**. Plenum Press.
10. van Daalen, M. R.; et al. (1993). "A stochastic neural architecture that exploits dynamically reconfigurable FPGAs". *FPGAs for Custom Computing Machines, Proceedings IEEE, NAPA*.

11. Gaudet, Vincent; Rapley, Anthony (February 2003). "Iterative decoding using stochastic computation". *Electronics Letters*. **39** (3): 299–301. doi:10.1049/el:20030217 (https://doi.org/10.1049%2Fel%3A20030217).
12. Alaghi, A.; Li, C.; Hayes, J. P. (2013). "Stochastic circuits for real-time image-processing applications". *Proceedings of the 50th Annual Design Automation Conference on - DAC '13*. p. 1. doi:10.1145/2463209.2488901 (https://doi.org/10.1145%2F2463209.2488901). ISBN 9781450320719.
13. Najafi, M. H.; Salehi, M. E. (2016). "A Fast Fault-Tolerant Architecture for Sauvola Local Image Thresholding Algorithm Using Stochastic Computing". *IEEE Transactions on Very Large Scale Integration (VLSI) Systems - TVLSI '16*. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*. **24**. pp. 808–812. doi:10.1109/TVLSI.2015.2415932 (https://doi.org/10.1109%2FTVLSI.2015.2415932).
14. Najafi, M. H.; Lilja, D. J.; Riedel, M. D.; Bazargan, K. (2016). *Polysynchronous stochastic circuits*. *2016 21st Asia and South Pacific Design Automation Conference (ASP-DAC)*. pp. 492–498. doi:10.1109/ASPDAC.2016.7428060 (https://doi.org/10.1109%2FASPDAC.2016.7428060). ISBN 978-1-4673-9569-4.
15. Alaghi, A.; Hayes, J. P. (2013). "Survey of Stochastic Computing". *ACM Transactions on Embedded Computing Systems*. **12** (2s): 1. CiteSeerX 10.1.1.296.4448 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.296.4448). doi:10.1145/2465787.2465794 (https://doi.org/10.1145%2F2465787.2465794).
16. Winstead, C.; Rapley, A.; Gaudet, V.; Schlegel, C. (September 2005). "Stochastic iterative decoders". *IEEE International Symposium on Information Theory*. Adelaide Australia. arXiv:cs/0501090 (https://arxiv.org/abs/cs/0501090).
17. Gaudet, Vincent; Rapley, Anthony (February 2003). "Iterative decoding using stochastic computation". *Electronics Letters*. **39** (3): 299–301. doi:10.1049/el:20030217 (https://doi.org/10.1049%2Fel%3A20030217).
18. Gross, W.; Gaudet, V.; Milner, A. (2006). "Stochastic implementation of LDPC decoders". *Conference Record of the Thirty-Ninth Asilomar Conference on Signals, Systems and Computers*.

Further reading

- Gaines, Brian R. (1967). "Techniques of Identification with the Stochastic Computer" (http://pages.cpsc.ucalgary.ca/~gaines/reports/COMP/IdentSC/IdentSC.pdf) (PDF). *Proceedings IFAC Symposium on "The Problems of Identification in Automatic Control Systems", Section 6 Special Identification Instruments, Prague June 12–19, 1967*. Retrieved 2013-11-11.
- Alaghi, Armin; Hayes, John P. (2013). "Survey of Stochastic Computing" (http://homes.cs.washington.edu/~armin/ACM_TECS_2013.pdf) (PDF). *ACM Transactions on Embedded Computing Systems*. **12** (2s): 1–19. CiteSeerX 10.1.1.296.4448 (https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.296.4448). doi:10.1145/2465787.2465794 (https://doi.org/10.1145%2F2465787.2465794). Retrieved 2013-11-11.