Study of random rounding scheme applied to gradient descent method and Deep Neural Networks

Christophe Pont[1], David Defour[1], and Bijan Mohammadi[2]

[1]*LAMPS, Univ. de Perpignan, 52 Av. Paul Alduy, 66860 Perpignan, France*
{christophe.pont,david.defour}@univ-perp.fr
[2]*IMAG, Université de Montpellier, Place Eugène Bataillon, Montpellier, France.*
bijan.mohammadi@umontpellier.fr

January 10, 2018

Abstract

In this paper, we propose to explore the vast literature adressing the concerns raised by floating point arithmetic computations. We also introduce the reader to Monte Carlo Arithmetic (MCA)[1], a promising extension of floating point arithmetic. MCA exploit randomness in floating point computations in order to analyze and estimate the quality of numerical results.

Lastly, we extend this work in applying the `round_random_nearness` rounding method, giving an insightful trail on its use and performance on various gradient descent methods such as Krylov's conjugate gradient methods. We also experiment its properties during stochastic gradient descent method, in the context of Deep Neural Network (DNN) training.

Contents

1 Introduction

In order to fit real numbers into a finite number of bits, it is necessary to *round* them. As one may already know though, this rounding can cause a loss of information, through loss of significant digits. Plus, each floating-point operation is followed by a rounding at the current precision, meaning potentially loosing information again. These undesired side-effects are called *roundoff errors*[2]. Numerical analysis aim at estimate this kind of errors in computations[3]. As the number of operation grows, the cumulated errors eventually get worse; hence the issue of numerical reliability in large simulations.

Among the various error sources, we propose to focus on roundoff errors, and more specifically to dig into the different rounding modes and their impact on the computations accuracy.

We study a rounding mode potentially able to *balance* the precision loss of floating point computations, and displaying interesting properties: the *stochastic rounding mode*. Parker previously called it the `round_random_nearness`[1] rounding method. It rely on Monte Carlo Arithmetic to introduce randomness in roundings.

- We propose to explain it and apply it to precision dependent contexts, and more precisely during gradient descent methods, particularly because they are vastly used during Deep Learning processes. Indeed, Deep Learning has seen a rise of interest since the applications of Deep Neural Networks (DNN). One of the learning method used is gradient descent method, thus it seems reasonnable to apply our findings to this field, as we'll see in the last part.
- We expect to see the minimum precision required to converge increase as we use the stochastic rounding mode, hence having a reduced precision requirement for some memory expensive iterative process.

2 State of the art

2.1 Floating point arithmetic

- Floating point arithmetic can be seen as a way for a physical computer to deal with theoretically infinitely many digits real numbers,

given their finite physical memory capacity. To accomplish that, a real number is necessarily rounded to the closest floating point value (hence loosing part of the significant digits of the real number).

- Let's examine the basis of floating point numbers representation.
  As Goldberg[4] describes it, a floating point number of precision p, exponent e and in base β can be represented as such : $\pm d.dd\cdots d \times \beta^e$. Here, $d.dd\cdots d$ is the significand of the floating point number, each d being a distinct digit in base β; in other words

$$\forall t \in \{0, 1 \cdots p-1\}\,,\ 0 < d_t < \beta.$$

- The represented number is

$$\pm \left(d_0 + d_1\beta^{-1} + ... + d_{p-1}\beta^{-(p-1)} \right) \beta^e. \tag{1}$$

- The vast litterature[1] about the different kind of errors and their consequences tells us a lot about the gravity of these matters.

- Cancellation errors arise when two close floating point number are substracted to one another, resulting a loss of significant digits. *example*

- The different errors can accumulate and progressively drift the resulting value away from the real value it is supposed to describe

- More recent approaches study the interesting properties of the application of Monte Carlo techniques to floating point arithmetic[1]. MCA give us the ability to study the distribution of errors, through statistical analysis. It provides insightful quantification about roundoff errors and program instability. It also circumvent some anomalies of floating point arithmetic, as for instance the non associative summation (which is, with MCA, 'statistically associative'.

- MCA approach has been the basis of numereous promising works, such as CADNA[5] and Verificarlo[6].

2.1.1 Rounding methods

- As seen earlier, when a real value isn't exactly representable as a floating point number, it is necessary to round it in order to store it in the computer's memory.

[1] For the reader who'd like to dive deeper into the concepts stated here, Goldberg's[4] tutorial form a great starting point.

- IEE754-1985 norm is a standard for floating-point arithmetic which adresses many problems found in previous implementations. It defines formats, operations and rounding rules in order to make the implementations more reliable and portable. The IEEE-754 norm suggest four rounding modes[7] :
  - Round toward 0
  - Round toward $-\infty$
  - Round toward $+\infty$
  - Round to nearest (ties to even)
- The most commonly used is round to nearest. However, it is does not come exempt of error[2]. MCA gives us two additionnal rounding modes :
  - `round_random_up_or_down` rounding method. It simply consists in rounding up or down with probability $\frac{1}{2}$
  - `round_random_nearness` rounding method. In this case, the probability of rounding is related to the proximity of the corresponding floating point value. This is the rounding mode that give us the most promising results in our studied case.

2.2 Gradient descent methods

- We propose to apply this rounding mode to a specific iterative process : the conjuguate gradient method[3].
- The goal of the algorithm is to solve particular systems of linear equations (with a symmetric and positive-definite associated matrix).
- The conjuguate gradient method aim at solving linear equations systems, where the matrix is symmetric and positive defined. It iteratively minimize the function

$$f : x \mapsto \frac{1}{2}(Ax, x) - (b, x)$$

- As we're about to see, its convergence is particularly sensitive to precision in floating point computations; hence also sensitive to the rounding mode used.

[2] *ref needed*

[3] A good introduction to this method can be found in the great related Shewchuk paper[8].

2.3 Deep Neural Networks, Learning Methods

- Deep Learning appear to benefit from a rise of interest, and many stunning industrial applications has emerged (especially from Google with DeepMind or Waymo[4]).
- Deep Neural Networks use different learning methods during inference phase; one of them is gradient descent method.
- In this context, studying the sensitivity of this method can give insightful perspectives.
- Other works can give a good insight of the progress of this subject[9][10].

3 Monte Carlo Arithmetic and stochastic rounding

3.1 Method

The method, why we chose it, which of its properties interest us...
We decided to develop and validate the `round_random_nearness` method, and to apply it to a program of conjuguate gradient method (in C) with different parameters.

3.2 Tests

Present the tests made to validate and prove our stochastic rounding mode. Why we are certain that it's working and some screenshots or tests results, with some examples showing its efficiency on simple cases

4 Results

4.1 Sensibility to rounding mode changes

- We applied this methodology to a conjuguate gradient descent method, by changing the precision used for each run, from 2 bits to 100. We then gathered the last value of the gradient after 60 iterations, and did it for each computed precision.
- Below you'll see a plot showing the difference between the stochastic rounding and classic RNDN rounding. For now though, it looks like for lower precisions (<20 bits), both stochastic and RNDN diverge, and stochastic rounding seem to diverge a little bit more. They tend to converge from a precision of 25 bits, apparently in the same way.

[4] *ref needed*

more tests will be made in order to test other data and to assert this result. Indeed, the randomness of our stochastic rounding is yet to be proven

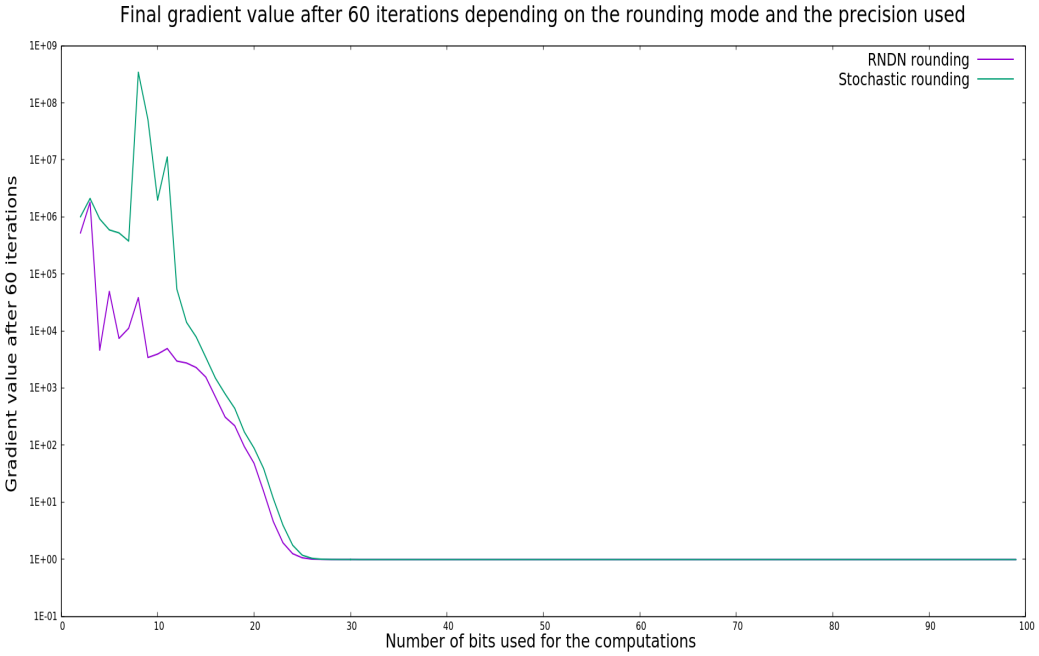

Figure 1: First results

- *What is the impact of changing the rounding mode on* ***performance*** *and* ***reliability*** *(in what way this change implies a deviation on* ***convergence****)*
- *resulting tables and charts of precision level for each configuration*

4.2 Other experimented sensibilities

Results of varying:

- *the matrix size,*
- *the matrix type,*
- *the number of iterations,*
- *the random repartition function of the stochastic rounding*
- *and the floating point numbers precision*

with results associated if relevant

5 Conclusion and future works

- *Conclude about our approach, our results, and if needed the other thing that could need to be done*
- *Cite other similar initiatives, such as Precimonious[11] or Verrou[12].*

References

[1] Douglass Stott Parker. *Monte Carlo Arithmetic: exploiting randomness in floating-point arithmetic*. University of California (Los Angeles). Computer Science Department, 1997. URL: http://web.cs.ucla.edu/~stott/mca/.

[2] Toyohisa Kaneko and Bede Liu. "On local roundoff errors in floating-point arithmetic". In: *Journal of the ACM (JACM)* 20.3 (1973), pp. 391–398.

[3] Michiel Hazewinkel. *Encyclopedia of Mathematics. Numerical analysis.* 2001st ed. Springer Science+Business Media B.V. / Kluwer Academic Publishers, 1994. ISBN: 978-1-55608-010-4. URL: http://www.encyclopediaofmath.org/index.php?title=Numerical_analysis&oldid=36393.

[4] David Goldberg. "What Every Computer Scientist Should Know About Floating-Point Arithmetic". In: *ACM Computing Surveys* 23.1 (Mar. 1991), pp. 5–48. ISSN: 0360-0300. DOI: 10.1145/103162.103163. URL: http://www.lsi.upc.edu/~robert/teaching/master/material/p5-goldberg.pdf.

[5] Stef Graillat et al. "Stochastic arithmetic in multiprecision". In: *Mathematics in Computer Science* 5.4 (2011), pp. 359–375. DOI: 10.1007/s11786-011-0103-4.

[6] Christophe Denis, Pablo de Oliveira Castro, and Eric Petit. "Verificarlo: checking floating point accuracy through Monte Carlo Arithmetic". working paper or preprint. Sept. 2015. URL: https://hal.archives-ouvertes.fr/hal-01192668.

[7] Dan Zuras et al. "IEEE standard for floating-point arithmetic". In: *IEEE Std 754-2008* (2008), pp. 1–70. DOI: 10.1109/IEEESTD.2008.4610935.

[8] Jonathan Richard Shewchuk et al. *An introduction to the conjugate gradient method without the agonizing pain*. 1994.

[9] Suyog Gupta et al. "Deep learning with limited numerical precision". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*. 2015, pp. 1737–1746. URL: http://proceedings.mlr.press/v37/gupta15.pdf.

[10] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. "Training deep neural networks with low precision multiplications". In: *arXiv preprint arXiv:1412.7024* (Dec. 22, 2014). arXiv: 1412.7024v5 [cs.LG].

[11] Cindy Rubio-González et al. “Precimonious”. In: *Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis on - SC ’13*. ACM Press, 2013. DOI: 10.1145/2503210.2503296.

[12] François Févotte and Bruno Lathuilière. “VERROU: Assessing Floating-Point Accuracy Without Recompiling”. In: (2016).