2024-11-06 SONNX WG Meeting

SONNX and numerical accuracy

(Elements of discussion)

Reminder

- The MLMD specifies the model to be implemented.
- The MLMD shall be such that any correct implementation preserves some <**property**>

(What is this <property>? ML performance? behaviour? structure?,... But how do you estimate the performance/behaviour of a model without executing it? Does it actually express a relation with the training model?)

Let's consider that <property> = function, i.e., the relation between inputs and ouputs expressed by the MLMD

Facts

- An inference is essentially a huge sequence of numerical computations
- On computers, numerical computations are not done in $\mathbb{R}$ since numbers are represented with a finite number of bits
    - All numbers can't be represented exactly so numbers are rounded to a representable value...
    - Operations may over- / under-flow...
    - (Catastrophic) cancellation effects may occur...
    - Associativity property is lost : $(a + b) + c \neq a + (b + c))$
    - Computation results depend on the implementation of operations
- ***How shall we handle this in SONNX?***

Shall we?

Summary of previous discussions

Premices

- With no formal requirement about accuracy, what does the following requirement mean?

  > "Operator `sin` shall compute $sin(x)$"

- Formally, it means that value returned by the function must be **exactly** the sine of the input.
- But we already no that that this can't be achieved. So it means that theere is an error. If th eerror is not specified,it can be anything in $[-\infty, +\infty]$. Not very useful.
- In practice (computer science) it means that the value returned by the function but be *as close as possible* to the actual sine.

Summary of previous discussions

Conclusions

- The MLMD standard shall specify the minimum expected accuracy of all operators
- The MLMD standard shall provide a means to specify the accuracy of any of its implementation (i.e., an implementation is deemed correct if it complies with the specified accuracy)
- **What is actually really necessary / required?****

Questions

- Is numerical accuracy a problem for us?
- Is it a new problem?
- Is it a particularly serious problem for ML?
- Is it solved by IEEE754?
- How is it handled today?
- Is it more crucial important for ML algorithms?

Is numerical accuracy a problem for us?

Element of context

- We are not targeting DAL-A systems…
- ML algorithms are inherently erroneous. How does numerical error compare to the computation errors due to floating point operations?

Is numerical accuracy a problem for us?

Semantic preservation

- How is it related the capability **to preserve the semantics of the model**?
  - If we specify the operators in $\mathbb{R}$, how can an implementation using `float`, `double` comply with this specification?
    - It can't... there are numerous examples where operations using floats give not only different, but **strange** results...
  - If we specify operators for a given datatype (`float32`, `float64`, etc.), will it be sufficient?
    - Will we be specifying the operation or providing an implementation?
    - Will it be different from providing a reference implementation?

Is numerical accuracy a problem for us?

Reproducibility

- Reproducibility (weak)
  - For an implementation complying with the MLMD, all executions of the same inference (same inputs) shall provide the same output.
- Reproducibility (strong)
  - For all implementations complying with the the MLMD, all executions of the same inference (same inputs) shall provide the same outputs

Is it a new problem?

- No, it is not
- IEEE 754 was created to solve some of those problems (not all)

Is it a particularly serious problem for ML?

- ML involves many computations (esp. deep learning)
  - In a unique inference, errors may accumulate leading to results completely different from those that would be obtained in $\mathbb{R}$.
  - This has to be compared with real-time system for which state is (usually) renewed due to the renewal of inputs.
- On the other side
  - ML algorithms are pretty robust (performance-wise) to quantization errors
  - There are many examples where quantization does not alter performance that much.

Is is solved by IEEE754

- IEE754 defines what a correctly rounded operation is:

  > "[...] each of the computational operations specified by this standard that returns a numeric result shall be performed as if it first produced an intermediate result correct to infinite precision and with unbounded range, and then rounded that intermediate result, if necessary, to fit in the destination’s format"

- But
  - Not all hardware used in ML computations are IEEE754 compliant: BF16, TF32,...
  - IEEE 754 aims at standardizing te way floating point computatoin are done, not to mitigate issues due to rounding (such as catastrophic cancellation)

How is it handled today?

- In industrial applications?
    - Testing? Formal verification (`fluctuat`)?
- In development assurance standards
    - Aero.: nothing specifically said about computation errors. Covered by "normal" verification activities...
    - Other: ???
- In language standards
    - C99:
        - Nothing is said about the accuracy of operators...
        - Rule about associativity: left associativity is required...
    - Ada:
        - Section G.2.4 gives accuracy requiremnts for elementary functions (see here)

- Mathematical libraries
    - `libm`: no requirement about accuracy, not correctly rounded...
    - `BLAS`: no requirement about accuracy, not correctly rounded...
    - ESA's `libmCS` does not give any accuracy requirement...

Should we specify maximal errors on a per operator basis?

- What about the global error?
- How can we give a value to the error?

Over and under flow

- Define conditions on the inputs in which no overflow / underflow will never occur?
    - Easy for an addition, not feasible for a convolution..
- Define the expected behaviour in case of over- under- flow?

Semantic preservation by design

- The description of the operation is prescriptive, i.e., operations shall be done in the way defined by the standard.

"Semantic preservation" by verification

- The implementation is deemed corect if it passes some test set
  - The test set is part of the specification
- The implementation is deemed correct if it is proved (in the formal sense) to be correct...

The question of replication [ARP6983, §6.4.3.6]

- Exact replication

> - [...] the ML Model description should contain sufficient details on the ML Model semantic to fully ***preserve this semantic*** in the implemented ML Model. For example, an exact replication criterion may be the direct and faithful implementation of the ML Model description so that the implemented ML Model meets ***the same performance, generalization, stability, and robustness requirements***.

- Approximated replication

  > - [...] the ML Model description should contain sufficient details on the ML Model semantics ***to approximate this semantic*** in the implemented ML Model with a specified tolerance. For example, an approximation metric may be expressed for a given dataset by **the maximal gap between the trained ML Model outputs and the implemented ML Model outputs_**. The corresponding approximation replication requirement may be that this maximal gap should not exceed a given value epsilon.

Requirements (illustration)

- [**REQ**] The MLMD shall specify the the exact ordering of operations, the representation of numbers, the roundings.
- [**REQ**] The MLMD shall specify the accuracy of all operations
- [**REQ**] The MLMD shall specify the exact input domain in which the model shall provide an output.
- [**REQ**] The SONNX standard shall specify th eexpected behaviour should a runtim error occur (over/under-flow, division by zero,...).
- [**REQ**] The SONNX standard shall provide exactly rounded operators.