Technical Perspective: (Pre-) Semirings Come to the Recursion Party

Atri Rudra
University at Buffalo
atri@buffalo.edu

(This article is an imagined conversation with my U. at Buffalo UG algorithms class students.)

Alright, let's begin. Let me tell you about this great database theory paper that I just read.

Boring! Does this have anything to do with AI/ML?

It does indeed. But first, a little bit of history. As we have seen earlier, traditionally algorithms have been designed for the RAM model where the problem and the input data are equally important. But starting with Codd's discovery of relation algebra, database folks realized data was da bomb.

Wait, so ML is not the only data-first discipline?

Indeed, database researchers pioneered the data-first approach: pre-process the data and store it in a way that the *same* representation can be used to solve *multiple* problems. Specifically, they developed a *query language* where one combines certain atomic operations (e.g. the join operator, which lists all occurrence of a fixed (hyper)graph in a large (hyper)graph) into more generic queries.

You mean a SQL query that Dr. Kennedy goes on about?

Yep! SQL proved to be an immensely powerful data manipulation tool. For decades it reigned by manipulating data that was *carefully curated* to be in nice tabular form.

But ChatGPT deals with data not in a nice tabular form.

Yes, the boom in ML has more to do with less-structured data (using algorithms developed for RAM models). The main reason for database algorithms lagging behind is the class of problems that SQL can model is limited. Consider the *transitive closure* problem where given a graph $G = (V, E)$ we want to decide for every pair of vertices $(u, v) \in V \times V$ whether u and v are connected. We can solve this problem by using BFS but here is another way to re-state the problem. Let $A \in \{0,1\}^{n \times n}$ be the adjacency matrix of G and consider A^{n-1} but with the following twist: addition is Boolean OR and multiplication is Boolean AND. Then u and v are connected iff $A^{n-1}[u, v] = 1$.

I have seem matrix operations before in my ML class but not with the above funky addition and multiplication.

Yeah, the computation above is over the so called *Boolean semiring*. Semiring just means we have 'addition' and 'multiplication' operations but not 'subtraction' or 'division.' But SQL cannot compute A^{n-1} since it requires iteration (or as Dr. Ziarek will say, recursion). Database folks recognized this and developed the Datalog language that allows one to do recursion on top of SQL operation. So the transitive closure property can be presented as $T(u, v) \leftarrow E(u, v) \vee (\vee_w (T(u, w) \wedge E(w, v)))$.

OK, we have talked about the shortest path problem to death in this course. Can Datalog handle that problem?

No: but the new paper can! Consider the All Pairs Shortest Paths (APSP) problem where our input is a edge weighted graph and we want to compute the cost of the shortest path between any two pairs of vertices $u, v \in V$. APSP is similar to the transitive closure in that we can define the problem in the following recursive format: $P(u, v) \leftarrow \min\{E(u, v), \min_w (P(u, w) + E(w, v))\}$.

Isn't the recursive formula for APSP the same as transitive closure if we replace the $\vee$ and $\wedge$ with min and $+$?

Indeed! The two recursive forms are 'basically' the same if we think of 'addition' as min and 'multiplication' as $+$ (over $\mathbb{R}$) (i.e. we use the so-called *Tropical semiring*).

So let's just make Datalog work over semirings?

Not so fast. There are two problems. First, Datalog is *defined* to be over the Boolean semiring so it cannot work over other semirings. Second, there is a technical hurdle. Recall we want to deal with recursion, which means we need some notion of progress of how far we are from the 'base case'. I.e., we need the elements of semirings to be ordered.

OK fine, let's define Datalog over semirings with ordering?

Well, the catch is how do you define the ordering. There is a natural notion of an ordering (called ahem, the *natural order*). There is previous work that looked at variants of Datalog with semirings that have a natural order. But the semiring of real numbers does not have this property.

But does it really matter if we cannot handle $\mathbb{R}$?

Yes, it does! Remember how y'all wanted ML. Well, the backbone of ML are matrix operations but the operations are over the real semiring. This paper achieves this by generalizing Datalog to *Datalog*$^\circ$ so that it can handle recursion over any pre-semiring with a *partial order* on the elements! The main insight of the paper is that this partial order need not be the natural order (if it exists). Thus, making the ordering *independent* of the operations unlocks other problems that can be solved– even a variant of the gradient descent that is the workhorse of ML training.

So when can I use *Datalog*$^\circ$ in PyTorch?

Not so fast: while the paper shows that gradient descent works, it is under the assumption that we need to compute the answer exactly. However, gradient descent in practice does *approximate* computation and it is a great open question to extend the results in this paper to this setting.