The Gradient of Algebraic Model Counting

Anonymous submission

Abstract

Algebraic model counting unifies many inference tasks on logic formulas by exploiting semirings. Rather than focusing on inference, we consider learning, especially in statistical-relational and neurosymbolic AI, which combine logical, probabilistic and neural representations. Concretely, we show that the very same semiring perspective of algebraic model counting also applies to learning. This allows us to unify various learning algorithms by generalizing gradients and backpropagation to different semirings. Furthermore, we show how cancellation and ordering properties of a semiring can be exploited for more memory-efficient backpropagation. This allows us to obtain some interesting variations of state-of-the-art gradient-based optimisation methods for probabilistic logical models. We also discuss why algebraic model counting on tractable circuits does not lead to more efficient second-order optimization. Empirically, our algebraic backpropagation exhibits considerable speed-ups as compared to existing approaches.

1 Introduction

Algebraic model counting (AMC) generalizes the well-known satisfiability task on propositional logic formulas to semirings (Kimmig, Van den Broeck, and De Raedt 2017). Using AMC various probabilistic inference tasks can be solved using the same unifying algorithm, including calculating marginals with evidence, entropy, and the most probable explanation (MPE). The principles of AMC are reminiscent of those underlying the sum- and max-product algorithms for probabilistic graphical models (Friesen and Domingos 2016).

In the current machine learning age, inference is often combined with learning. This is the focus of statistical-relational learning (De Raedt et al. 2016) and neurosymbolic AI (Garcez and Lamb 2023), which aim to integrate the inference and learning paradigms of logic, probability and neural networks. Our goal is to *extend the unifying algebraic perspective that AMC brought to inference to the learning setting*. To this end, we show how gradients can be generalized over semirings, by taking inspiration from differentiable algebra. This algebraic gradient provides a new toolkit of tractable operations, which can be used to realize a wide variety of learning algorithms, including gradient descent, expectation-maximization, entropy maximization, and low variance gradient estimation.

From a theoretical perspective, the algebraic viewpoint provides a unifying framework and solver for computing many concepts that are used in learning. This is timely, as Marra et al. (2024) relate that "[t]he situation in neurosymbolic computation today is very much like that of the early days in statistical relational learning, in which there were many competing formalisms, sometimes characterized as the statistical relational learning alphabet soup". Ott et al. (2023) even state that "the field of neurosymbolic AI exhibits a progress-hampering level of fragmentation".

From a practical perspective, we create a single optimized algorithm that subsumes many existing ones as special cases. For example, the forward-backward algorithm of Darwiche (2003) and the gradient estimator of De Smet, Sansone, and Zuidberg Dos Martires (2023) have a quadratic worst-case time complexity. However, these are special cases of our algebraic backpropagation algorithm which has linear time complexity. We provide an efficient implementation for the algebraic backpropagation algorithm which takes the semiring properties into account. In our experiments, our implementation outperforms PyTorch and Jax, which are the de facto standard for neurosymbolic learning, by several orders of magnitude.

Finally, we consider algebraic learning algorithms that rely on second-order information. Indeed, empirical evidence of e.g. Liu, Zhang, and Van den Broeck (2022) suggests that the existing first-order optimization methods for circuits might be suboptimal. Second-order optimization has quadratic complexity in general, which explains its lack of popularity in machine learning. However, specialized tractable circuit representations such as sd-DNNF have been developed which support many operations in linear time which are otherwise NP-hard. Unfortunately, we show that this tractability does not carry over to second-order derivatives and Newton's method is unlikely to be feasible in linear time on these tractable circuits.

In summary, we make the following contributions.

1. We introduce the ∇AMC for computing algebraic gradients within a logical semiring framework and show that many tasks related to gradients and learning can be implemented as such an algebraic gradient.
2. We provide an optimized ∇AMC algorithm for the alge-

braic gradient by exploiting dynamic programming and semiring properties.

3. The ∇AMC algorithm is implemented in a user-friendly Rust library called *Kompyle*, and empirically outperforms state-of-the-art algorithms used in neurosymbolic learning
4. We prove that a second-order algebraic gradient cannot be computed by a circuit in linear time.

2 Preliminaries

We first review some relevant background on abstract algebra and propositional logic, before turning to algebraic model counting and algebraic circuits.

2.1 Algebra

Definition 2.1 (Commutative Monoid)**.** A commutative monoid $(A, \odot, e)$ is a set A with a binary operation $\odot : A \times A \to A$ and an identity element $e \in A$ such that the following properties hold for all a, b, and c in A.

- *Associativity*: $(a \odot b) \odot c = a \odot (b \odot c)$
- *Commutativity*: $a \odot b = b \odot a$
- *Neutrality of Identity*: $e \odot a = a$

An element $a \in A$ is *idempotent* when $a \odot a = a$. A monoid is idempotent when all its elements are idempontent. Note that the identity e is always idempotent ($e \odot e = e$).

Definition 2.2 (Commutative Semiring)**.** A commutative semiring $(A, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ combines two commutative monoids $(A, \oplus, e^{\oplus})$ and $(A, \otimes, e^{\otimes})$ where the following properties hold for all a, b, and c in A.

- *Distributivity*. $(a \oplus b) \otimes c = (a \otimes c) \oplus (b \otimes c)$
- *Absorption*. $e^{\oplus} \otimes a = e^{\oplus}$

We will from on now write monoid or semiring for brevity, and leave the commutativity implied. A *ring* is a semiring with additive inverses, meaning that for every a in A, there is an inverse element $-a$ such that $a \oplus -a = e^{\oplus}$.

A simple example of a semiring is the Boolean semiring, which has true ($\top$) and false ($\bot$) as the domain and uses the logical OR and AND operations as sum and product, respectively. Table 1 gives an overview of relevant semirings along with the shorthand name we employ for them. For example, we denote the Boolean semiring as BOOL.

2.2 Propositional Logic

Syntax We write $\mathcal{V}$ for a set of propositional variables. The infinite set of logical formulas $\mathcal{F}_{\mathcal{V}}$ over the variables $\mathcal{V}$ is defined inductively as follows. A formula $\phi \in \mathcal{F}_{\mathcal{V}}$ is either true $\top$, false $\bot$, a propositional variable $v \in \mathcal{V}$, a negation of a formula $\neg\phi_1$, a conjunction of formulas $\phi_1 \wedge \phi_2$, or a disjunction of formulas $\phi_1 \vee \phi_2$. *Literals* are variables or negated variables, and the set of all literals is denoted as $\mathcal{L} = \mathcal{V} \cup \{\neg v \mid v \in \mathcal{V}\}$.

Definition 2.3. Given a formula ϕ, the *conditioned* formula $\phi|x$ with $x \in \mathcal{L}$ equals the formula ϕ where every occurrence of the literal x is replaced with $\top$ and every occurrence of $\neg x$ is replaced with $\bot$. When $x \notin \mathcal{L}$, $\phi|x$ is defined to be $\bot$.

Semantics An *interpretation* $I \subset \mathcal{L}$ is a set of literals which denotes a truth assignment. This means that for each variable $v \in \mathcal{V}$, either $v \in I$ or $\neg v \in I$. When a formula ϕ is satisfied in the interpretation I according to the usual semantics, we say that I is a *model* of ϕ, written as $I \models \phi$. The set of all models of a formula is denoted $\mathcal{M}(\phi) = \{I \mid I \models \phi\}$.

From the algebraic view, the propositional formulas also form a semiring $(\mathcal{F}_{\mathcal{V}}, \vee, \wedge, \bot, \top)$. As opposed to the BOOL semiring, the operations $\vee$ and $\wedge$ are structural here and create new formulas from their arguments. This is also known as the *free* commutative semiring generated by the literals $\mathcal{L}$.

2.3 Algebraic Model Counting

The task of *algebraic model counting* (AMC) consists of evaluating the models of a formula in a given semiring (Kimmig, Van den Broeck, and De Raedt 2017).

Definition 2.4 (Algebraic Model Counting)**.** Given a semiring $(A, \oplus, \otimes, e^{\oplus}, e^{\otimes})$ and a labelling function $\alpha : \mathcal{L} \to A$ which maps literals into the semiring domain, the algebraic model count is a mapping from formulas into the semiring domain.

$$\text{AMC}(\phi; \alpha) = \bigoplus_{I \in \mathcal{M}(\phi)} \bigotimes_{x \in I} \alpha(x)$$

AMC generalizes many existing inference tasks. For example, the satisfiability (SAT) task, which asks whether a formula has a model, can be solved by AMC in the BOOL semiring. Model counting (#SAT), which asks how many models a formula has, is solved using the NAT semiring, and weighted model counting (WMC) using the PROB semiring. WMC is of particular significance, as probabilistic inference in e.g. Bayesian networks can be reduced to WMC (Chavira and Darwiche 2008).

Example 1. Consider the formula $\phi = (x \vee y) \wedge z$ over the set of variables $\mathcal{V} = \{x, y, z\}$. This formula has three models: $\mathcal{M}(\phi) = \{\{x, y, z\}, \{\neg x, y, z\}, \{x, \neg y, z\}\}$. So for the AMC, we get

$$\begin{aligned}\text{AMC}(\phi; \alpha) = \; & (\alpha(x) \otimes \alpha(y) \otimes \alpha(z)) \\ & \oplus (\alpha(\neg x) \otimes \alpha(y) \otimes \alpha(z)) \\ & \oplus (\alpha(x) \otimes \alpha(\neg y) \otimes \alpha(z))\end{aligned}$$

To compute the model count, we can evaluate the above in the NAT semiring with a constant labelling function $\forall x \in \mathcal{L} : \alpha(x) = 1$, giving $\text{AMC}_{\text{NAT}}(\phi; \alpha) = 3$. If we take some more complex labelling function, e.g.

$$\alpha(x) = 0.5,\ \alpha(\neg x) = 0.5,\ \alpha(y) = 0.1,$$
$$\alpha(\neg y) = 0.9,\ \alpha(z) = 0.8,\ \alpha(\neg z) = 0.2$$

we get $\text{AMC}_{\text{PROB}}(\phi; \alpha) = 0.44$ for the weighted model count.

The algebra of propositional logic (the *Boolean algebra*, not to be confused with the Boolean semiring BOOL) observes additional properties on top of the semiring such as idempotency. This difference between the Boolean algebra and the free semiring is precisely what makes AMC hard; two equivalent formulas might not be equivalent under a specific semiring. For example, $\phi \wedge \phi$ equals ϕ in the Boolean algebra but not in the free semiring.

Semiring	Domain A	$\oplus$	$\otimes$	$e^{\oplus}$	$e^{\otimes}$	Labels $\alpha(x)$	AMC Task	∇AMC Task
BOOL	$\{\top, \bot\}$	$\vee$	$\wedge$	$\bot$	$\top$	$\top$	SAT	Cond. SAT
						$\sim p(x)$	Sampling	IndeCateR
NAT	$\mathbb{N}$	$+$	$\times$	0	1	1	#SAT	Cond. #SAT
PROB	$\mathbb{R}_{\geq 0}$	$+$	$\times$	0	1	$p(x)$	WMC	Gradient
LOG	$\{-\infty\} \cup \mathbb{R}_{\leq 0}$	logaddexp	$+$	$-\infty$	0	$\log p(x)$	Log WMC	Log Gradient
VITERBI	$\mathbb{R}_{\geq 0}$	max	$\times$	0	1	$p(x)$	MPE	MGE
TROPICAL	$\mathbb{R}_{\geq 0}$	max	$+$	$-\infty$	0	$\log p(x)$	Log MPE	Log MGE
FUZZY	$[0, 1]$	max	min	0	1	$p(x)$	Fuzzy	/
GRAD$_y$	$\mathbb{R}_{\geq 0} \times \mathbb{R}$	Eq. 3	Eq. 4	$(0, 0)$	$(0, 1)$	$(p(x), \frac{\partial p(x)}{\partial y})$	Gradient	Hessian
SENS	$\mathbb{R}[\mathcal{V}]$	$+$	$\times$	0	1	x	Sensitivity	Cond. Sensitivity
OBDD	OBDD($\mathcal{V}$)	$\vee$	$\wedge$	OBDD(0)	OBDD(1)	OBDD(x)	OBDD	Cond. OBDD

Table 1: Overview of relevant commutative semirings with their corresponding interpretation in AMC and ∇AMC.

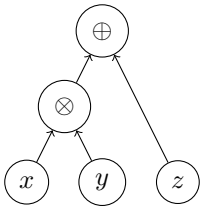
2.4 Circuits

By reusing subformulas, circuits are a more compact representation for Boolean formulas (Vollmer 1999).

Definition 2.5 (Boolean Circuit)**.** A Boolean circuit is a directed acyclic graph representing a propositional formula. This means that every leaf node contains a literal, and all other nodes are either $\vee$-nodes or $\wedge$-nodes.

Algebraic circuits generalize Boolean circuits to semiring operations (Derkinderen and De Raedt 2020). Algebraic circuits in the PROB semiring are better known as *arithmetic circuits*.

Example 2. The formula $\phi = (x \wedge y) \vee z$ can be represented by the algebraic circuit below.



The tractability of queries on a circuit are related to the structural properties of that circuit (Darwiche and Marquis 2002; Vergari et al. 2021). For example, *determinism* is required to compute the model count in polynomial time. A circuit is deterministic if all the children of an $\vee$-node are mutually exclusive, meaning they do not share any model. The circuit in Example 2 is not deterministic for example.

Transforming circuits to achieve structural properties such as determinism is achieved with *knowledge compilation* (Darwiche and Marquis 2002). Kimmig, Van den Broeck, and De Raedt (2017) proved that the properties of the semiring determine are linked to structural properties of the algebraic circuit. For example, determinism is needed to evaluate in a semiring that is not additively idempotent (of which model counting in the NAT semiring is an example). From an algebraic viewpoint, knowledge compilation tries to generate the smallest circuit representing the models $\mathcal{M}(\phi)$ within a specific semiring.

3 Conditionals as Gradients

Due to the inclusion of neural networks, neurosymbolic methods are typically trained using gradient descent. As probabilistic inference is differentiable, these neurosymbolic models are end-to-end trainable with off-the-shelf optimizers such as Adam (Kingma and Ba 2017). Some examples of this approach include the semantic loss (Xu et al. 2018) and DeepProbLog (Manhaeve et al. 2018). Other statistical-relational techniques are frequently trained by expectation-maximization (EM), which is also strongly linked to gradients (Xu and Jordan 1996). For this reason, we focus on the computation of gradients.

It is known that the gradient of probabilistic inference is is the same as conditional inference (Darwiche 2003). In AMC with the PROB semiring, better known as weighted model counting, we have that

$$\frac{\partial \text{AMC}_{\text{PROB}}(\phi; \alpha)}{\partial \alpha(x)} = \text{AMC}_{\text{PROB}}(\phi | x; \alpha)$$

We hence propose to generalize the notion of gradients to algebraic model counting as follows.

Definition 3.1. The AMC gradient is defined as the vector of AMC conditionals to each literal $x_i \in \mathcal{L}$.

$$\nabla\text{AMC}(\phi; \alpha) = [\text{AMC}(\phi | x_1; \alpha), \ldots, \text{AMC}(\phi | x_n; \alpha)]^\top$$

Observe that ∇AMC is well-defined in any semiring, even when the semiring domain is discrete or otherwise non-differentiable such as in the BOOL or NAT semirings.

Literal vs Variable Gradients Our definition of ∇AMC is the gradient towards a literal and not a variable. However, when maximizing the labels of the positive and negative literal separately, the global optimum is trivial. In practice, the positive and negative labels of a variable v are often linked, e.g. as $\alpha(v) \oplus \alpha(\neg v) = e^{\otimes}$. In this case, it follows that the AMC derivative to a variable v is

$$\text{AMC}(\phi | v; \alpha) \ominus \text{AMC}(\phi | \neg v; \alpha)$$

We will not further make this distinction, as the above can be computed straightforwardly from the AMC gradient from Definition 2.4 and is only well-defined on rings.

Example 3. Consider again $\phi = (x \vee y) \wedge z$, the formula of Example 1. Then $\nabla\text{AMC}(\phi; \alpha)$ is

$$\begin{aligned}
&[\text{AMC}(\phi|x; \alpha), \ldots, \text{AMC}(\phi|\neg z; \alpha)]^\top \\
&= [\alpha(y)\otimes\alpha(z) \oplus \alpha(\neg y)\otimes\alpha(z), \\
&\quad \alpha(x)\otimes\alpha(z) \oplus \alpha(\neg x)\otimes\alpha(z), \\
&\quad \alpha(x)\otimes\alpha(y) \oplus \alpha(\neg x)\otimes\alpha(y) \oplus \alpha(x)\otimes\alpha(\neg y), \\
&\quad \alpha(y)\otimes\alpha(z), \\
&\quad \alpha(x)\otimes\alpha(z), \\
&\quad \alpha(e^\oplus)]
\end{aligned}$$

3.1 Conditionals as Semiring Derivations

We further motivate the definition of ∇AMC by its relation to differentiable algebra (Kolchin 1973). Differentiable algebra studies generalizations of derivatives, through functions which observe the same linearity and product rule as conventional derivatives.

Definition 3.2 (Semiring Derivation)**.** A derivation δ on a semiring $(A, \oplus, \otimes, e^\oplus, e^\otimes)$ is a map $\delta : A \to A$ on itself which satisfies the following properties for all a and b in A.

- *Linearity*: $\delta(a \oplus b) = \delta(a) \oplus \delta(b)$
- *Product rule*: $\delta(a \otimes b) = (a \otimes \delta(b)) \oplus (b \otimes \delta(a))$

Many properties that hold for conventional derivation are retained for semiring derivations. For example, $\delta(e^\oplus) = \delta(e^\otimes) = e^\oplus$ in any derivation. We refer to e.g. Dimitrov (2017) for a summary of existing results on semiring derivations. Interestingly, semiring derivations themselves induce a monoid structure (Chajda and Länger 2021). Here, we denote the set of all possible derivations on A as $\mathcal{D}(A)$.

Theorem 1. *$(\mathcal{D}(A), \oplus, \delta_0)$ is a commutative monoid, with δ_0 the derivation that maps all elements to $e^\oplus$.*

Proof. By simple verification of the monoid axioms. □

We can now connect this to our definition of ∇AMC, as the elements of ∇AMC are exactly what generates this derivation monoid.

Theorem 2. ∇AMC *is the minimal generating set for the monoid of derivations $(\mathcal{D}(A), \oplus, \delta_0)$.*

Proof. See Appendix B. □

In other words, all possible semiring derivations in $\mathcal{D}(A)$ are a linear combination of the elements in ∇AMC. So in this sense, ∇AMC can be seen as a basis for semiring derivations.

4 Computing ∇AMC

Conditioning a formula is straightforward, and hence computing ∇AMC can be done with any off-the-shelf AMC solver. This naive approach closely equals forward mode differentiation and was already proposed for WMC by Sang, Bearne, and Kautz (2005) to compute the conditionals of Bayesian inference.

Forward mode differentiation is well-known for scaling linearly in the number of input variables. Reverse mode differentiation, better known as *backpropagation*, is a dynamic programming algorithm that scales linearly in the number of output variables, which is usually constant. The backpropagation algorithm can easily be extended to work over semirings (Du et al. 2023), and can hence compute ∇AMC.

As a dynamic programming algorithm, the downside of backpropagation is its memory use. More precisely, the naive backpropagation algorithm on a circuit has linear memory complexity in the number of edges the circuit. Shih et al. (2019) already demonstrated that when the semiring is a semifield, i.e. there is a division operation, this memory complexity reduces to linear in the number of nodes of the circuit. As the number of edges in a circuit can be up to the square of the number of nodes, this forms a substantial improvement. We show that this semifield requirement can be dropped while retaining the same memory complexity.

Theorem 3. *The backward pass on an algebraic circuit C has $O(e)$ time and $O(n)$ memory complexity, where e and n are the number of edges and nodes in C respectively.*

Theorem 3 is realised by Algorithm 1. This algorithm assumes that the forward pass already computed the values of sum nodes as $\alpha(n) = \bigoplus_{c\in\text{children}(n)} \alpha(c)$ and product nodes as $\alpha(n) = \bigotimes_{c\in\text{children}(n)} \alpha(c)$. Algorithm 1 then performs backpropagation in the usual way, going over the circuit from the root to the leaves and calculating the gradients of the children of each node using the chain rule. Concretely, the derivative towards a node n is

$$\gamma(n) = \bigoplus_{p\in\text{parents}(n)} \gamma(n) \tag{1}$$

when n has sum nodes as parents (lines 5-7), or

$$\gamma(n) = \bigoplus_{p\in\text{parents}(n)} \bigotimes_{c\in\text{children}(p)\setminus\{n\}} \alpha(c) \tag{2}$$

when n has product nodes as parents (lines 9-19).

Algorithm 1 relies on some dynamic programming to avoid recomputing the inner product in Equation 2 for every parent. For example, taking the gradient of $c_1 \otimes c_2 \otimes c_3$ requires us to compute $[c_2\otimes c_3, c_1\otimes c_3, c_1\otimes c_2]$. Naively, computing these individually will result in quadratic time complexity. We avoid this using two cumulative products: the forward $[e^\otimes, c_1, c_1 \otimes c_2]$ and the backward $[c_2 \otimes c_3, c_2, e^\otimes]$. Both of these cumulative products can be computed in linear time, and their element-wise product results in the desired gradient.

Further optimizations on Algorithm 1 are possible depending on the semiring structure to avoid the need to store the cumulative products in the backpropagation of the product nodes (lines 9-14). A first property that can achieve this is cancellation.

Definition 4.1. A semiring element $a \in A$ is (multiplicatively) *cancellative* if, for each b and c in A, $a \otimes b = a \otimes c$ implies $b = c$.

Cancellation can be seen as a generalization of inverses. So when $c = a \otimes b$ and a is cancellative we can write $b = c \oslash a$. Indeed, the inverse $c \oslash a$ must exist as $c = a \otimes b$, and the cancellation property furthermore assures that it is unique. The semiring of the natural numbers NAT form an

Algorithm 1: Algebraic Backpropagation

Input: A circuit C over the literals $\mathcal{L}$, a labelling of the nodes α (computed in the forward pass), and a semiring $(A, \oplus, \otimes, e^{\oplus}, e^{\otimes})$.
Output: The algebraic gradient $\nabla\text{AMC}(C; \alpha)$.

$\gamma \leftarrow [e^{\oplus}, \ldots, e^{\oplus}]$, a vector with $e^{\oplus}$ for each node in C
$\gamma[\text{root of } C] \leftarrow e^{\otimes}$
for nodes n in the circuit C, parents before children **do**
  if n is a sum node **then**
    for c in children(n) **do**
      $\gamma[c] \leftarrow \gamma[c] \oplus \gamma[n]$
    end for
  else if n is a product node **then**
    $r \leftarrow [e^{\otimes}, \ldots, e^{\otimes}]$, with length $|\text{children}(n)|$
    $t \leftarrow e^{\otimes}$
    for c in children(n) **do**
      $r[c] \leftarrow t$
      $t \leftarrow t \otimes \alpha[c]$
    end for
    $t \leftarrow e^{\otimes}$
    for c in children(n) in reverse order **do**
      $\gamma[c] \leftarrow \gamma[c] \oplus \gamma[n] \otimes t \otimes r[c]$
      $t \leftarrow t \otimes \alpha[c]$
    end for
  end if
end for
return γ

example where we can use cancellation even though there are no inverse elements.

If a product node n is multiplicatively cancellative, we can simply compute the product gradient as

$$\gamma(n) = \bigoplus_{p \in \text{parents}(n)} \alpha(p) \oslash \alpha(n)$$

A second property we can exploit is ordering.

Definition 4.2. We call $a, b \in A$ (multiplicatively) *ordered* when $a \otimes b = a$ or $a \otimes b = b$.

When the children of a node n are all ordered, the product derivatives are simply $\gamma[n] \otimes \alpha[n]$ for all children except the largest child. For the largest child, the derivative is $\gamma[n]$ times the one-but-largest child.

The FUZZY semiring is an example of a semiring where all elements are multiplicatively ordered. Note that barring the identity, ordering and cancellation are mutually exclusive, meaning they can be used complementarily. For example, in the PROB and NAT semirings, all elements are cancellative except zero which is ordered. So we can always exploit either cancellation or ordering, depending on the value of the node. We present a variation of Algorithm 1 in Appendix C which exploits both these cancellation and ordering properties.

5 Semirings for ∇AMC

We now explore the use of ∇AMC for first-order optimization and related applications. In the probabilistic setting, we assume that our formula ϕ has weights $\alpha(x) \in [0, 1]$ with $\alpha(\neg x) = 1 - \alpha(x)$. In other words, the variables correspond to Bernoulli distributions, and we have a probability distribution over interpretations $p(I; \alpha) = \prod_{l \in I} \alpha(l)$. The weighted model count can then also be seen as a probability, which we denote $p(\phi; \alpha) = \text{AMC}_{\text{PROB}}(\phi; \alpha)$.

PROB semiring By construction, the algebraic gradient ∇AMC in the PROB semiring is just the usual gradient.

$$\nabla\text{AMC}_{\text{PROB}}(\phi; \alpha) = \nabla_\alpha\, p(\phi; \alpha)$$

LOG semiring Next, if we take the LOG semiring, we instead get the logarithm of the gradient. Note that this is different from $\nabla_\alpha \log p(\phi; \alpha)$, the gradient of the log probability.

$$\nabla\text{AMC}_{\text{LOG}}(\phi; \alpha) = \log \nabla_\alpha\, p(\phi; \alpha)$$

Backpropagation with the LOG semiring provides a more numerically stable way to compute gradients. It can also be applied for Expectation-Maximization (EM), as the expectation step on logic formulas reduces to computing the conditionals $p(x \mid \phi)$ (Peharz et al. 2020).

$$p(x \mid \phi) = \exp\left(\nabla\text{AMC}_{\text{LOG}}(\phi; \alpha) + \alpha - \text{AMC}_{\text{LOG}}(\phi; \alpha)\right)$$

The above also closely relates to the PC flow as defined by Choi, Dang, and Van den Broeck (2021).

VITERBI semiring In the VITERBI semiring, the AMC gradient computes the maximum gradient over all models. Here, $\max$ computes the element-wise maximum of the gradient vectors of each model of ϕ.

$$\nabla\text{AMC}_{\text{VITERBI}}(\phi; \alpha) = \max_{I \in \mathcal{M}(\phi)} \nabla_\alpha\, p(I; \alpha)$$

Use-cases for this include greedily approximating the true gradient or gradient-based interpretability. Similarly, the **TROPICAL semiring** gives the log-space equivalent of the VITERBI semiring.

$$\nabla\text{AMC}_{\text{TROPICAL}}(\phi; \alpha) = \log \max_{I \in \mathcal{M}(\phi)} \nabla_\alpha\, p(I; \alpha)$$

GRAD semiring AMC with the GRAD semiring computes the Shannon entropy (Li and Eisner 2009).

$$H(\phi) = -\sum_{I \in \mathcal{M}(\phi)} p(I; \alpha) \log p(I; \alpha)$$

Taking the gradient of the AMC in GRAD hence results in the conditional Shannon entropy towards each literal.

$$\nabla\text{AMC}_{\text{GRAD}}(\phi) = [H(\phi \mid x_1), \cdots H(\phi \mid x_n)]^\top$$

This conditional entropy is used as the information gain for learning, or for interpretability or regularization purposes. Several other statistical quantities such as the KL divergence can be framed as the difference between the entropy and conditional entropy, and can hence easily be computed from the GRAD semiring.

BOOL semiring with sampled labels By sampling from the labels α, we can combine the Boolean semiring BOOL with a stochastic labelling. This implements a naive Monte Carlo approximation, which equals the true probability in expectation.

$$\mathbb{E}_{w\sim\alpha}[\text{AMC}_{\text{BOOL}}(\phi; w)] = p(\phi; \alpha)$$

Interestingly, we can immediately take the algebraic gradient here to get an unbiased gradient estimator.

$$\mathbb{E}_{w\sim\alpha}[\nabla\text{AMC}_{\text{BOOL}}(\phi; w)] = \nabla_\alpha\, p(\phi; \alpha)$$

Upon closer inspection, this equals IndeCateR (De Smet, Sansone, and Zuidberg Dos Martires 2023), a state-of-the-art unbiased estimator for gradients of discrete distributions that Roa-Blackwellizes REINFORCE. Moreover, by using backpropagation we estimate the gradient in linear time in the size of the formula, as opposed to the quadratic time required in the original formulation by De Smet, Sansone, and Zuidberg Dos Martires (2023).

Other semirings Some remaining semirings have less clear gradient-based interpretations, but might still be of note. The gradient in the **SENS semiring** computes sensitivity polynomials for the conditioned formulas. The gradient of the **FUZZY semiring** computes the highest minimal membership of a literal in each conditioned formulas. Taking the gradient in the **OBDD semiring** creates a multi-rooted circuit for the gradient using bottom-compilation (i.e. an OBDD circuit which explicitly computes the forward and backward pass).

6 Second-Order Derivations

So far, we only considered first-order methods such as gradient descent. Second-order methods such as Newton's method take curvature into account by preconditioning the gradient using the inverse Hessian. For a function f parameterized by θ, Newton's method updates the parameters as

$$\theta \leftarrow \theta + [\nabla^2 f(\theta)]^{-1} \nabla f(\theta)$$

Computing the full Hessian matrix has quadratic complexity, making the cost of second-order methods for machine learning often prohibitive. However, tractable circuits support a wide range of inference operations in polytime that are otherwise NP-hard (Vergari et al. 2021). The question hence poses itself whether tractable circuits could improve upon this quadratic complexity.

Similar to the gradient, we first generalize the Hessian matrix to AMC as follows.

$$\nabla^2\text{AMC}(\phi) = \begin{bmatrix} \text{AMC}(\phi|x_1, x_1) & \dots & \text{AMC}(\phi|x_1, x_n) \\ \vdots & & \vdots \\ \text{AMC}(\phi|x_n, x_1) & \dots & \text{AMC}(\phi|x_n, x_n) \end{bmatrix}$$

We ignore the labels α here to simplify notation. By construction, ∇^2AMC is the conventional Hessian in the PROB semiring. Note that although the algebraic Hessian is well-defined in any semiring, applying Newton's method requires that the semiring is a field as the Hessian needs to be inverted.

By using ∇AMC in the GRAD semiring, we get a straightforward way to calculate ∇^2AMC for the PROB semiring. Indeed, the GRAD semiring calculates partial derivatives using dual numbers, so in algebraic backpropagation this gives a row of the Hessian.

$$\nabla\text{AMC}_{\text{GRAD}_x}(\phi) = \left[\frac{\partial^2 \text{AMC}_{\text{PROB}}(\phi)}{\partial\alpha(x)\partial\alpha(y_1)}, \dots, \frac{\partial^2 \text{AMC}_{\text{PROB}}(\phi)}{\partial\alpha(x)\partial\alpha(y_n)}\right]$$

Next, we prove that ∇^2AMC on tractable circuits still lacks structure, meaning that the expensive quadratic memory cost cannot avoided.

Theorem 4. *Representing $\nabla^2\text{AMC}(C)$ of a circuit C over n variables has a $O(n^2)$ memory complexity, even when C is smooth, decomposable, and deterministic.*

Proof sketch. Take for example the binary Galois field as the semiring. Given a sequence of n^2 bits, we can now construct a formula ϕ such that the (flattened) Hessian equals this bit sequence. This means that, in general, the Hessian has no structure and a sub-quadratic memory complexity would imply lossless compression. □

One might give two counter-arguments to the theorem above. First, gradient descent takes linear memory in the circuit size, but this circuit size might be up to exponential in the number of variables. Indeed, Theorem 4 does not rule out that the Hessian can be stored in linear memory complexity in the size of the circuit. Second, calculating the Hessian is typically not the true goal. The above result does not rule out the tractability of a matrix-free approach, where the preconditioned gradient $[\nabla^2\text{AMC}(\phi)]^{-1}\nabla\text{AMC}(\phi)$ is computed directly without constructing the Hessian explicitly. However, even when considering these arguments, the existence of an algorithm for Newton's method which runs in linear time complexity in the circuit size remains unlikely.

Theorem 5. *Given a circuit C with n nodes, there cannot exist a circuit C' with size $O(n)$ that computes the preconditioned gradient $[\nabla^2 AMC(C)]^{-1}\nabla AMC(C)$, even when C is deterministic, decomposable and smooth.*

Proof. See Appendix B. □

In Appendix B.3, we further discuss that relaxing the computational model of arithmetic circuits does not improve the situation much. More precisely, if there would exist any kind of algorithm that computes the preconditioned gradient in linear time in the size of the circuit, a quadratic matrix multiplication algorithm would be implied. The above results indicate that, much like for deep learning, first-order optimization might still be preferable. Our theorems do not rule out that there might be specific semirings which are not fields where the situation is better. However, the practical relevance of Hessians in these semirings is more limited.

7 Related Work

Semirings & Inference The semiring perspective in artificial intelligence has its roots in weighted automata (Schützenberger 1961), as the weights of an automata can be defined over a semiring. These ideas have been carried over

Time (s)	PROB	BOOL	LOG	FUZZY
Kompyle	**0.0082 ± 0.0124**	**0.0042 ± 0.0080s**	**0.0142 ± 0.0256**	**0.0167 ± 0.0259**
- *dynamic* (Algorithm 1)	0.0190 ± 0.0298	0.0156 ± 0.0243	0.0241 ± 0.0432	0.0271 ± 0.0441
- *naive* + *cancel.* (Shih et al. 2019)	0.2035 ± 0.8075	0.0108 ± 0.0151	0.1682 ± 0.6326	/
- *naive* (Du et al. 2023)	0.2767 ± 0.8518	0.0161 ± 0.0250	0.2332 ± 0.6562	0.3075 ± 0.8296
Pytorch	7.4660 ± 15.7834	/	7.3753 ± 15.6997	/
Jax	Out of Memory	/	Out of Memory	/

Table 2: Runtime to compute the algebraic gradient ∇AMC averaged over 100 instances from the MC2021 competition (track 2). "*cancel.*" denotes the cancellation property. See Section 8 for an explanation of the ablations of Kompyle. Methods that did not run on all circuits within 16GB of memory are denoted "Out of Memory". PyTorch and Jax cannot compute ∇AMC in the BOOL and FUZZY semirings. All timings are in seconds, lower is better.

to various other paradigms such as constraint satisfaction problems (Bistarelli, Montanari, and Rossi 1997), parsing (Goodman 1999), dynamic programs (Eisner, Goldlust, and Smith 2005), database provenance (Green, Karvounarakis, and Tannen 2007), logic programming (Kimmig, Van den Broeck, and De Raedt 2011), propositional logic (Kimmig, Van den Broeck, and De Raedt 2017), answer set programming (Eiter and Kiesel 2020), Turing machines (Eiter and Kiesel 2023), and tensor networks (Goral et al. 2024).

Semirings & Learning While semirings have mostly been applied to inference problems, some works also investigated learning in semirings. Li and Eisner (2009) introduced the expectation semiring, which can be used to compute gradients and perform expectation-maximization. Pavan et al. (2023) studied the complexity of constraint optimization in some semirings. On the other hand, our goal is to provide a more general framework, where we can compute derivations of any semiring.

Darwiche (2001) already described a forward-backward algorithm for computing conditionals of a formula. Backpropagation on semirings has been described recently by (Du et al. 2023). Most similar to our work, Shih et al. (2019) already included an algorithm for computing the conditionals on algebraic circuits, which they call All-Marginals instead of ∇AMC. This algorithm can be seen as a special case of our cancellation optimization, as all cancellative commutative monoids can be embedded in a group using the Grothendieck construction.

Neurosymbolic Learning Several neurosymbolic methods rely on (probabilistic) circuits and hence could apply the algebraic learning framework we outlined. Some examples include DeepProbLog (Manhaeve et al. 2018), the semantic loss (Xu et al. 2018), and probabilistic semantic layers (Ahmed et al. 2022). Dickens, Pryor, and Getoor (2024) proposed another overarching view of neurosymbolic learning by framing it as energy-based models. On the other hand, our work focuses on algebraic circuits where inference and learning can be performed exactly.

8 Experiments

We implemented the algebraic backpropagation in a Rust library called *Kompyle*, and empirically demonstrate the runtime performance of the algebraic backpropagation algorithm on several semirings.

Setup As a benchmark, we take 100 formulas of the 2021 Model Counting Competition (Fichte, Hecher, and Hamiti 2021) and compile them to d-DNNF circuits using the d4 knowledge compiler (Lagniez and Marquis 2017). We randomly generate weights for the formulas, with on average 1% of the weights being set to zero. All experiments were performed on the CPU of a 2022 M2 MacBook Pro. We include PyTorch (Paszke et al. 2019) and Jax (Frostig, Johnson, and Leary 2018) as a baseline for the regular gradient, as they are frequently used for neurosymbolic learning in practice. Furthermore, we include ablations for Kompyle. Firstly, a naive variant, as was used by Du et al. (2023). Secondly, a variant which also exploits cancellation, as was proposed by Shih et al. (2019). Thirdly, a variant which applies the dynamic programming described in Algorithm 1. Finally, the full version of Kompyle with all optimizations (using ordering and cancellation).

Results Table 2 contains the results. PyTorch and Jax perform poorly, as these frameworks are not optimized for very large computational graphs. Jax does not run within the 16GB of RAM, even on comparatively small circuits (ca. 100MB). Even though real-world circuits are far from the worst-case quadratic complexity, the dynamic programming considerably outperforms the naive variants of Kompyle. The semiring optimizations yield smaller but still considerable speed-ups, depending on the semiring.

9 Conclusion

We proposed a notion of gradients for algebraic model counting as conditional inference. We showed that many quantities of interest in learning, such as gradients, Hessians, conditional entropy, etc. can be seen as this algebraic gradient in different semirings. Furthermore, we introduced an optimized backpropagation algorithm for a broad class of semirings. Finally, we gave an indication that second-order optimization is still expensive on tractable circuits.

References

Ahmed, K.; Teso, S.; Chang, K.-W.; Van den Broeck, G.; and Vergari, A. 2022. Semantic Probabilistic Layers for Neuro-

Symbolic Learning. In *Advances in Neural Information Processing Systems*.

Bistarelli, S.; Montanari, U.; and Rossi, F. 1997. Semiring-based constraint satisfaction and optimization. *Journal of the ACM*, 44(2): 201–236.

Chajda, I.; and Länger, H. 2021. Integration in semirings. ArXiv:2110.00245 [math].

Chavira, M.; and Darwiche, A. 2008. On probabilistic inference by weighted model counting. *Artificial Intelligence*, 172(6): 772–799.

Choi, Y.; Dang, M.; and Van den Broeck, G. 2021. Group Fairness by Probabilistic Modeling with Latent Fair Decisions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(13): 12051–12059. Number: 13.

Darwiche, A. 2001. On the Tractable Counting of Theory Models and its Application to Truth Maintenance and Belief Revision. *Journal of Applied Non-Classical Logics*, 11(1-2): 11–34. Publisher: Taylor & Francis _eprint: https://doi.org/10.3166/jancl.11.11-34.

Darwiche, A. 2003. A differential approach to inference in Bayesian networks. *Journal of the ACM*, 50(3): 280–305.

Darwiche, A.; and Marquis, P. 2002. A Knowledge Compilation Map. *Journal of Artificial Intelligence Research*, 17: 229–264.

De Raedt, L.; Kersting, K.; Natarajan, S.; and Poole, D. 2016. *Statistical Relational Artificial Intelligence: Logic, Probability, and Computation*, volume 10 of *Synthesis lectures on artificial intelligence and machine learning*. Morgan & Claypool Publishers. ISBN 978-1-62705-841-4.

De Smet, L.; Sansone, E.; and Zuidberg Dos Martires, P. 2023. Differentiable Sampling of Categorical Distributions Using the CatLog-Derivative Trick. In *Advances in Neural Information Processing Systems*.

Derkinderen, V.; and De Raedt, L. 2020. Algebraic Circuits for Decision Theoretic Inference and Learning. In *ECAI 2020*, 2569–2576. IOS Press.

Dickens, C.; Pryor, C.; and Getoor, L. 2024. Modeling Patterns for Neural-Symbolic Reasoning Using Energy-based Models. *Proceedings of the AAAI Symposium Series*, 3(1): 90–99. Number: 1.

Dimitrov, S. 2017. Derivations on semirings. *AIP Conference Proceedings*, 1910(1): 060011.

Du, K.; Torroba Hennigen, L.; Stoehr, N.; Warstadt, A.; and Cotterell, R. 2023. Generalizing Backpropagation for Gradient-Based Interpretability. In Rogers, A.; Boyd-Graber, J.; and Okazaki, N., eds., *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 11979–11995. Toronto, Canada: Association for Computational Linguistics.

Eisner, J.; Goldlust, E.; and Smith, N. A. 2005. Compiling Comp Ling: practical weighted dynamic programming and the Dyna language. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT '05, 281–290. USA: Association for Computational Linguistics.

Eiter, T.; and Kiesel, R. 2020. Weighted LARS for Quantitative Stream Reasoning. In *ECAI 2020*, 729–736. IOS Press.

Eiter, T.; and Kiesel, R. 2023. Semiring Reasoning Frameworks in AI and Their Computational Complexity. *Journal of Artificial Intelligence Research*, 77: 207–293.

Fichte, J. K.; Hecher, M.; and Hamiti, F. 2021. The Model Counting Competition 2020. *ACM Journal of Experimental Algorithmics*, 26: 13:1–13:26.

Friesen, A.; and Domingos, P. 2016. The Sum-Product Theorem: A Foundation for Learning Tractable Models. In *Proceedings of The 33rd International Conference on Machine Learning*, 1909–1918. PMLR. ISSN: 1938-7228.

Frostig, R.; Johnson, M. J.; and Leary, C. 2018. Compiling machine learning programs via high-level tracing. *Systems for Machine Learning*, 4.

Garcez, A. d.; and Lamb, L. C. 2023. Neurosymbolic AI: the 3rd wave. *Artificial Intelligence Review*.

Goodman, J. 1999. Semiring parsing. *Comput. Linguist.*, 25(4): 573–605.

Goral, A.; Giesen, J.; Blacher, M.; Staudt, C.; and Klaus, J. 2024. Model Counting and Sampling via Semiring Extensions. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18): 20395–20403. Number: 18.

Green, T. J.; Karvounarakis, G.; and Tannen, V. 2007. Provenance semirings. In *Proceedings of the twenty-sixth ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems*, PODS '07, 31–40. New York, NY, USA: Association for Computing Machinery. ISBN 978-1-59593-685-1.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2011. An Algebraic Prolog for Reasoning about Possible Worlds. *Proceedings of the AAAI Conference on Artificial Intelligence*, 25(1): 209–214.

Kimmig, A.; Van den Broeck, G.; and De Raedt, L. 2017. Algebraic model counting. *Journal of Applied Logic*, 22: 46–62.

Kingma, D. P.; and Ba, J. 2017. Adam: A Method for Stochastic Optimization. ArXiv:1412.6980 [cs].

Kolchin, E. R. 1973. *Differential Algebra & Algebraic Groups*. Academic Press. ISBN 978-0-08-087369-5.

Lagniez, J.-M.; and Marquis, P. 2017. An improved decision-DNNF compiler. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, 667–673. Melbourne, Australia: AAAI Press. ISBN 978-0-9992411-0-3.

Li, Z.; and Eisner, J. 2009. First- and second-order expectation semirings with applications to minimum-risk training on translation forests. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 1 - Volume 1*, EMNLP '09, 40–51. USA: Association for Computational Linguistics. ISBN 978-1-932432-59-6.

Liu, A.; Zhang, H.; and Van den Broeck, G. 2022. Scaling Up Probabilistic Circuits by Latent Variable Distillation.

Manhaeve, R.; Dumancic, S.; Kimmig, A.; Demeester, T.; and De Raedt, L. 2018. DeepProbLog: Neural Probabilistic Logic Programming. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc.

Marra, G.; Dumančić, S.; Manhaeve, R.; and De Raedt, L. 2024. From Statistical Relational to Neurosymbolic Artificial Intelligence: a Survey. *Artificial Intelligence*, 104062.

Ott, J.; Ledaguenel, A.; Hudelot, C.; and Hartwig, M. 2023. How to Think About Benchmarking Neurosymbolic AI? In *17th International Workshop on Neural-Symbolic Learning and Reasoning -NESY 2023*. Sienne, Italy.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; Desmaison, A.; Kopf, A.; Yang, E.; DeVito, Z.; Raison, M.; Tejani, A.; Chilamkurthy, S.; Steiner, B.; Fang, L.; Bai, J.; and Chintala, S. 2019. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Pavan, A.; Meel, K. S.; Vinodchandran, N. V.; and Bhattacharyya, A. 2023. Constraint Optimization over Semirings. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(4): 4070–4077. Number: 4.

Peharz, R.; Lang, S.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; Broeck, G. V. D.; Kersting, K.; and Ghahramani, Z. 2020. Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits. In *Proceedings of the 37th International Conference on Machine Learning*, 7563–7574. PMLR. ISSN: 2640-3498.

Sang, T.; Bearne, P.; and Kautz, H. 2005. Performing Bayesian inference by weighted model counting. In *Proceedings of the 20th national conference on Artificial intelligence - Volume 1*, AAAI'05, 475–481. Pittsburgh, Pennsylvania: AAAI Press. ISBN 978-1-57735-236-5.

Schützenberger, M. P. 1961. On the definition of a family of automata. *Information and Control*, 4(2): 245–270.

Shih, A.; Van den Broeck, G.; Beame, P.; and Amarilli, A. 2019. Smoothing Structured Decomposable Circuits. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Vergari, A.; Choi, Y.; Liu, A.; Teso, S.; and Van den Broeck, G. 2021. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference. In *Advances in Neural Information Processing Systems*, volume 34, 13189–13201. Curran Associates, Inc.

Vollmer, H. 1999. *Introduction to Circuit Complexity: A Uniform Approach*. Springer Science & Business Media. ISBN 978-3-540-64310-4. Google-Books-ID: 55ZTgOJs8bsC.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and Van den Broeck, G. 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *Proceedings of the 35th International Conference on Machine Learning*, 5502–5511. PMLR. ISSN: 2640-3498.

Xu, L.; and Jordan, M. I. 1996. On Convergence Properties of the EM Algorithm for Gaussian Mixtures. *Neural Computation*, 8(1): 129–151.

Reproducibility Checklist

This paper:

- Includes a conceptual outline and/or pseudocode description of AI methods introduced. *yes, see e.g. Algorithm 1 and Section 4*.
- Clearly delineates statements that are opinions, hypothesis, and speculation from objective facts and results. *yes*
- Provides well marked pedagogical references for less-familiare readers to gain background necessary to replicate the paper. *yes*

Does this paper make theoretical contributions? *yes*

- All assumptions and restrictions are stated clearly and formally. *yes*
- All novel claims are stated formally (e.g., in theorem statements). *yes*
- Proofs of all novel claims are included. yes, see Appendix B
- Proof sketches or intuitions are given for complex and/or novel results. *partial*
- Appropriate citations to theoretical tools used are given. *yes*
- All theoretical claims are demonstrated empirically to hold. *partial, see Section 8*
- All experimental code used to eliminate or disprove claims is included. *yes*

Does this paper rely on one or more datasets? *yes*

- A motivation is given for why the experiments are conducted on the selected datasets. *yes*
- All novel datasets introduced in this paper are included in a data appendix. *NA*
- All novel datasets introduced in this paper will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. *NA*
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are accompanied by appropriate citations. *yes*
- All datasets drawn from the existing literature (potentially including authors' own previously published work) are publicly available. *yes*
- All datasets that are not publicly available are described in detail, with explanation why publicly available alternatives are not scientifically satisficing. *NA*

Does this paper include computational experiments? *yes*

- Any code required for pre-processing data is included in the appendix. *yes*
- All source code required for conducting and analyzing the experiments is included in a code appendix. *yes*
- All source code required for conducting and analyzing the experiments will be made publicly available upon publication of the paper with a license that allows free usage for research purposes. *yes*

- All source code implementing new methods have comments detailing the implementation, with references to the paper where each step comes from. *partial*
- If an algorithm depends on randomness, then the method used for setting seeds is described in a way sufficient to allow replication of results. *NA*
- This paper specifies the computing infrastructure used for running experiments (hardware and software), including GPU/CPU models; amount of memory; operating system; names and versions of relevant software libraries and frameworks. *yes*
- This paper formally describes evaluation metrics used and explains the motivation for choosing these metrics. *partial*
- This paper states the number of algorithm runs used to compute each reported result. *yes*
- Analysis of experiments goes beyond single-dimensional summaries of performance (e.g., average; median) to include measures of variation, confidence, or other distributional information. *yes*
- The significance of any improvement or decrease in performance is judged using appropriate statistical tests (e.g., Wilcoxon signed-rank). *no*
- This paper lists all final (hyper-)parameters used for each model/algorithm in the paper's experiments. *NA*
- This paper states the number and range of values tried per (hyper-) parameter during development of the paper, along with the criterion used for selecting the final parameter setting. *NA*