

Internship Report

Paul LANDRIER
Supervised by Silviu Maniu

Contents

1	Introduction	1
2	Modeling a Computation	1
2.1	Computation Graphs	1
2.2	Model	2
2.3	Examples of Near-semirings	4
2.4	Properties of Generalized Computation Graphs	4
2.5	Universal Object	6
2.6	Generality of the Complete Near-semiring over a Monoid.	6
3	Instantiation of the Theory	7
3.1	Layer-Wise Relevance Propagation	7
3.2	Quantization	8
4	Conclusion	8
5	Extra work - GNN Stability	8
5.1	Definition of GNNs	8
5.2	Formal Framework for Stability	8
5.3	Theoretical results	9
5.4	Node embedding stability	13
5.5	Instantiation for Databases	14
[TODO: Remove table of contents (it stays for now for better readability)]		

abstract TODO (Mention broad usage of computation graphs (compilers))
Paragraph in model is a good starting poitn

1 Introduction

TODO [Describe the interest of the computation]

2 Modeling a Computation

2.1 Computation Graphs

We try to abstract the notion of *computation graphs*, having in mind the particular example of neural networks.

Computation graphs are a type of graphs that represent a computation. Formally, in this work, we define a computation graph (V, E, Fun, Op) as a directed acyclic graph (V, E) together with a function $Fun : E \rightarrow F$ where F is a set of functions $S \rightarrow S$ (S is a fixed set, the same for all functions) and a function $Op : V \rightarrow Operator$ where $Operator$ is a set containing operators to aggregate several inputs in S . (Remark : usually in a computation graph, the functions are stored in the nodes and the edges represent a data dependency. We choose a different convention here to remain closer to provenance in

graphs as defined in [Ramusat, 2022]. For an other definition of computation graphs, see for instance [Wang and Zhao, 2023].)

For instance, if we want to compute the function $f(x) = e^{x^2}(x^2 + 2x + 2)$ using only the linear, "plus constant" and exponential functions, with $+$ and \times operators, then we can model a possible computation with the computation graph :

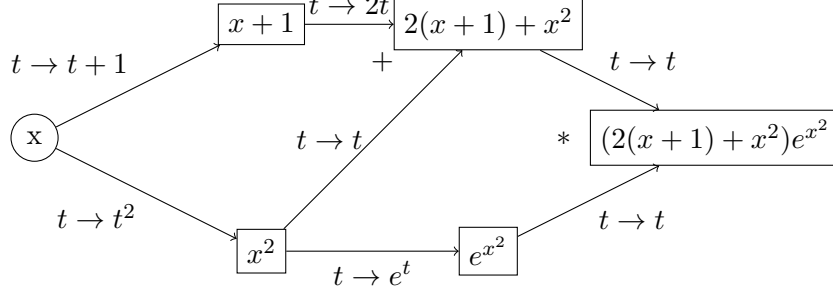


Figure 1: One possible computation graph of f .

Let's take the example of a (non-discretized) perceptron with two inputs. If we want to model its execution using only elementary operations and the activation function f , we get the representation in Figure 2.

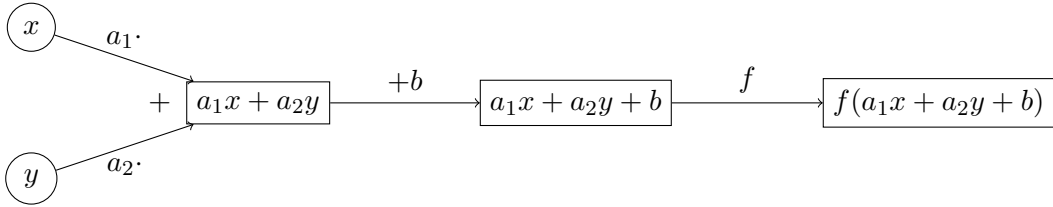


Figure 2: Computation graph of a perceptron.

Since a perceptron only uses one kind of aggregation -the sum-, we shall now only consider computation graphs with one unique operator to aggregate the functions. This operator is denoted $+$ and is implicitly attached to every node in the graph.

In the algorithm 1, we formalize the notion of "function represented by a computation graph" by defining the algorithm to compute this function from the graph. We assume that there is only one node without any outgoing edge, which is the output of the computation graph, noted *out*. We also assume that we have a special function f_v for each input node v . These special functions should be thought of as symbols, such as the " x " in a definition of the form $f(x) = 3x + 2$.

In the section 2.4, we prove that the result does not depend on the topological ordering chosen in the execution of the algorithm.

2.2 Model

[TODO: is modularity a good term ?]

In [Green et al., 2007] and [Ramusat, 2022], the authors defined a set of identities and restrictions on the object of their studies (respectively relational databases and graph databases) and then used this set to identify an algebraic structure to theoretically model them.

We follow the same method to characterize the minimal algebraic structure required to model a computation graph. An example of generalized computation graph is given in Figure 3. In what follows, we describe the different properties of computation graphs and traduce them algebraically. The process is summarized in the table of Figure 4 and specific instances where the described properties are necessary are presented in Figure 5. In the examples of Figure 5, the symbolic functions are omitted for better readability.

Algorithm 1 Retrieving a function from a computation graph

```
1: Input: Computation Graph  $G = (V, E, Fun, Op)$ .
2: Output: The function represented by the graph.
3:
4: Function Retrieve( $G$ ):
5:   Let  $T$  be a topological ordering of  $(V, E)$ .
6:   Initialize:  $Function[v] = \text{None}$  for all  $v \in V$ 
7:   for each  $v$  in  $T$  do
8:     if  $v$  is an input node then
9:        $Function[v] = f_v$ .
10:    else
11:      Initialize:  $f = 0$ 
12:      for each edge  $e = (u, v)$  entering  $v$  do
13:         $f = f + (Fun(e) \circ Function[u])$ 
14:  return  $Function[out]$ 
```

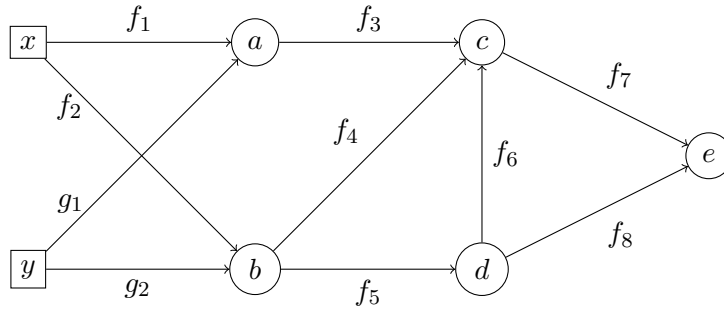


Figure 3: A generalised computation graph.

Structure needed We consider a structure $(F, \circ, +, 0, id)$. The $+$ and \circ are operators $F^2 \rightarrow F$ respectively used to model aggregation and composition in the computation graph.

Firstly, since there is no natural order on incoming edges for a node, $+$ must be associative and commutative so the sum on incoming edges is well-defined. For an example, see the Figure 5a.

We require the operator \circ to be associative to ensure modularity. For an example, see the Figure 5b.

To abstract the concept of zero-weight, we require the existence of a neutral element 0 which is a left absorbing element for the \circ operator and which is a neutral element for $+$. For an example, see the Figure 5c.

We require the left distributivity $(f + g) \circ h = f \circ h + g \circ h$ because it is of theoretical interest (it is necessary for the interpretation of near-semirings as sub-near-semirings of the complete near-semiring over a monoid), because it is a natural identity to interpret the elements as functions and because it allows for a simplification of the computation in some cases. For an example, see the Figure 5d.

Finally, we require the existence of a neutral element id for \circ . This might not be crucial, but it has interesting theoretical properties -it matters for theorem 1- and allows some natural representation. For an example, see the Figure 5e.

The axioms identified are those of a near-semiring, with two differences ($+$ is not assumed to be commutative in general and the “ id ” does not necessarily exist). In our case, we obtain the definition:

Definition 1 (Near-semiring). *A near-semiring $(F, +, \circ, 0_F, id)$ is a set F equipped with two binary operations $+$, \circ and two distinguished elements $0_F, id$ such that :*

- $(F, +, 0_F)$ is a commutative monoid;
- (F, \circ, id) is a monoid;

Property on the computation graph	Algebraic traduction of the property
The edges that enter a node are not ordered	$+$ is associative and commutative
The concatenation of two computation graphs is a computation graph that represent the composition of the functions	\circ is associative
There is a notion of zero-weight that model the fact that there is no interaction between two nodes.	There is a 0_F element that is a neutral element for $+$ and that is left absorbing for \circ .
We can reuse computation previously done in a natural way.	The operator \circ is right distributive over $+$.
We should be able to model the identity function.	There is a neutral element id for \circ .

Figure 4: Motivation of the algebraic structure.

- $\forall(f, g, h) \in F^3, (f + g) \circ h = f \circ h + g \circ h;$
- 0_F is left absorbing for \circ , i.e. for all $f \in F, 0_F \circ f = 0_F$.

Together with this definition comes the notion of morphism of near-semiring and of sub-near-semiring:

A sub-near-semiring of a near-semiring $(F, +, \circ, 0, id)$ is a subset $G \subset F$ such that $0 \in G, id \in G$ and G is stable by $+$ and \circ . If G is a sub-near-semiring of $(F, +, \circ, 0, id)$, then $(G, +, \circ, 0, id)$ (for the restriction of operations $+$ and \circ to G) forms a near-semiring.

If $(F, +, \circ, 0, id)$ and $(G, +', \circ', 0', id')$ are two near-semirings, then $\phi : F \rightarrow G$ is said to be a near-semiring morphism if it satisfies $\phi(0) = 0', \phi(id) = id', \forall(f, g) \in F^2, \phi(f + g) = \phi(f) +' \phi(g)$ and $\phi(f \circ g) = \phi(f) \circ' \phi(g)$. The image $\phi(F)$ is a sub-near-semiring of G .

2.3 Examples of Near-semirings

Before establishing further theoretical results, we give a few examples of near-semirings.

Near-semiring 1 (Complete near-semiring over a monoid). *Let $(M, +, 0_M)$ be a monoid. We define \mathcal{F} the set of all functions $M \rightarrow M$. Then $(\mathcal{F}, +, \circ, 0_F, id)$ where $0_F : x \mapsto 0_M$ is the zero function and $id : x \mapsto x$ is the identity function is a near-semiring, that we will call complete near-semiring over M . A particular sub-near-semiring of \mathcal{F} is the set of monoid endomorphisms of M (it is even a semiring). As we shall see later in this work, this is the most general kind of near-semiring in some sense.*

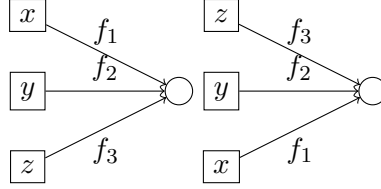
Near-semiring 2. *The \mathcal{C}^∞ functions over the real numbers form a near-semiring with usual addition, composition, zero function and identity.*

Near-semiring 3 (Semirings). *Let $(S, +, \cdot, 0, 1)$ be a semiring. Then $(S, +, \cdot, 0, 1)$ is also a near-semiring. Indeed, the axioms of near-semiring are strictly more general than those of a semiring. The interpretation of semirings as sets of function together with the interpretation of \cdot as a composition can be made through the structure of semimodules, which is itself a sub-near-semiring of the complete near-semiring over a monoid.*

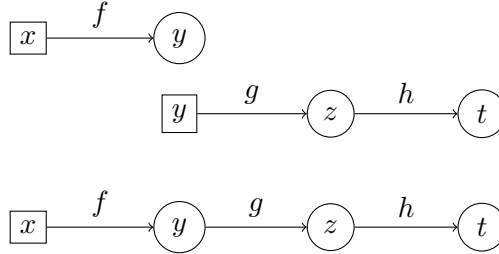
2.4 Properties of Generalized Computation Graphs

We now provide two properties on computation graphs. These two properties are sanity-check on the model, they ensure that w

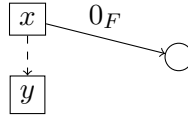
Property 1. *The output of algorithm 1 does not depend on the topological ordering chosen in the execution.*



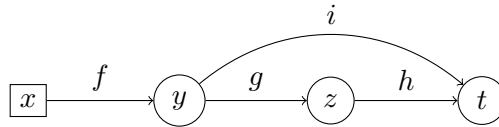
(a) A case that necessitates the “+” operator to be commutative and associative. Indeed, these two graphs are the same represented in two different ways, and the left one represents the element $(f_1 + f_2) + f_3$ whereas the right one represents $(f_3 + f_2) + f_1$. Replacing f_3 by 0 then shows that + must be commutative, and using the commutativity of + shows that the right expression is equal to $f_1 + (f_2 + f_3)$, so the associativity is required too.



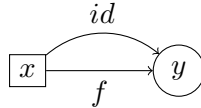
(b) Necessity of the associativity for the modularity. The fact that the graph below, which associated element is $h \circ (g \circ f)$, represents the same element as the composition of the two graphs above, which associated element is $(h \circ g) \circ f$, is equivalent to the associativity of \circ . Then we can say without ambiguity that they both represent $h \circ g \circ f$.



(c) Use-cases of the 0 weight. The plain arrow represents an existing edge with weight 0 (in the context of neural network for instance) and the dashed arrow represents an edge that does not exist and could be represented by the 0 element, which could for instance be used to model computation graphs as adjacency matrices.



(d) Illustration of the modularity allowed by the right distributivity. We can factor the expression $i \circ f + h \circ g \circ f$ in the smaller one $(i + g \circ h) \circ f$.



(e) A use case of the identity: representing $f(x) + x$.

Figure 5: The situations that justify the different properties discussed in Figure 4.

Proof. We consider a computation graph $G = (V, E, Fun, Op)$ and two topological orderings T_1, T_2 . We note $n = |V|$. We use the notations of algorithm 1 with an index 1 or 2 to refer to the execution that is considered.

We proceed by induction on $i \in [0, n]$ with the hypothesis H_i : "For all $1 \leq k \leq i$, if we note v_k the k -th node according to the ordering T_1 , then $Function_1[v_k] = Function_2[v_k]$ ".

In the base case, $i = 0$ and there is nothing to prove. We now consider $i \in [0, n]$, assume H_i is verified and prove that H_{i+1} is verified as well.

We only need to prove that $Function_1[v_{i+1}] = Function_2[v_{i+1}]$.

If v_{i+1} is an input node, then $Function_1[v_{i+1}] = f_{v_{i+1}} = Function_2[v_{i+1}]$.

Else, by definition, $Function_1[v_{i+1}] = \sum_{e=(u, v_{i+1}) \in E} Fun(e) \circ Function_1[u]$. However, since all the vertices u that appear in the sum are before v in the topological ordering, by the induction hypothesis this can be rewritten: $Function_1[v_{i+1}] = \sum_{e=(u, v_{i+1}) \in E} Fun(e) \circ Function_2[u] = Function_2[v_{i+1}]$, which conclude the proof. \square

Note that the only thing required by the proof is that the result of the aggregation operation $+$ (denoted by \sum in the computation) does not depend on the order of its arguments.

Property 2 (Modularity of computation graphs). *We consider two computation graphs G_1, G_2 with G_1 having only one output node and G_2 having only one input node. We define $G_{1 \circ 2}$ as the computation graph that contains the disjoint union of G_1 and G_2 with*

[TODO: the distributivity axiom and the composition axiom are both made so we can use some subpart of a graph as black-boxes. Can we prove that if we have a black-box graph representing a function f , we can indeed plot other graphs at the beginning/ at the end and the concatenation represents $g \circ f \circ h$?]

2.5 Universal Object

TODO write this properly:

In Ramusat we had : The full computation represented by the graph cannot be expressed in a similar fashion as for provenance in graphs ($\sum_{\pi \in P_{x,y}} w(\pi)$) because composition does not distribute over

$+$. In Green : polynomials

Us ?

Given a near-semiring F , we want to characterize the sub-near-semiring generated by a subset $G \subset F$. This notion makes sense for the usual reason: an intersection of sub-near-semiring is a sub-near-semiring itself, and we note $\langle G \rangle$ this sub-near-semiring .

For this purpose we define a sequence of sets (G_n) by $G_0 = \{0, id\} \cup G$ and

$$G_{n+1} = \left\{ g \circ \left(\sum_{i=1}^k g_i \right) \mid k \in \mathbb{N}, g \in G, (g_i)_{1 \leq i \leq n} \in (G_n)^k \right\}.$$

The claim is now that $\langle G \rangle = \bigcup_{n \in \mathbb{N}} G_n$. Indeed, $G \subset G_0 \subset \bigcup_{n \in \mathbb{N}} G_n$, then we show by induction that $\forall n \in \mathbb{N}, G_n \subset \langle G \rangle$ and finally we show that $\bigcup_{n \in \mathbb{N}} G_n$ is indeed a sub-near-semiring of F .

[TODO: Write this proof cleanly or rewrite the section]

2.6 Generality of the Complete Near-semiring over a Monoid.

Theorem 1. *Universal Form of Near-semirings*

It is equivalent to be given:

1. a near-semiring $(F, +, \circ, 0, id)$

2. a commutative monoid $(M, +, 0)$ together with a set of functions $M \rightarrow M$ containing 0 and id and stable by \circ and $+$. (I.e. a sub-near-semiring of the complete near-semiring over M .)

Proof. Firstly, we assume that we are given a monoid $(M, +, 0)$ and a set of functions $M \rightarrow M$ containing 0, id and stable by $+$ and \circ like in two. This precisely means that we are given a sub-near-semiring of the complete near-semiring over $(M, +, 0)$, and hence a near-semiring.

Secondly, if we are given a near-semiring $(F, +, \circ, 0, id)$, we define for an element f the function

$$\bar{f} : \begin{cases} F & \rightarrow & F \\ g & \mapsto & f \circ g \end{cases},$$

then we define the function $\phi : \begin{cases} F & \rightarrow & ((F, +, 0) \rightarrow (F, +, 0)) \\ f & \mapsto & \bar{f} \end{cases}$.

The function ϕ is a morphism of near-semirings. Indeed:

- $\phi(0)$ is the zero function since zero is left absorbing,
- $\phi(id)$ is the identity function because $\forall g \in F, id \circ g = g$,
- $\forall f_1, f_2 \in F^2, \forall g \in F, \phi(f_1 + f_2)(g) = (f_1 + f_2) \circ g = f_1 \circ g + f_2 \circ g = \phi(f_1)(g) + \phi(f_2)(g)$, hence $\phi(f_1 + f_2) = \phi(f_1) + \phi(f_2)$
- $\forall f_1, f_2 \in F^2, \forall g \in F, \phi(f_1 \circ f_2)(g) = (f_1 \circ f_2) \circ g = f_1 \circ (f_2 \circ g) = \phi(f_1)(f_2 \circ g) = (\phi(f_1) \circ \phi(f_2))(g)$, hence $\phi(f_1 \circ f_2) = \phi(f_1) \circ \phi(f_2)$

Therefore, $\phi(F)$ is a sub-near-semiring of the complete near-semiring over the monoid $(F, +, 0)$. \square

This theorem is interesting because it clearly indicates a direction for further investigation -if we want to instantiate the theoretical framework, we should define computation graphs over some monoids- and a limit of the expresiveness of the theory. However, it does not provide any information on the structure of near-semirings.

It might be interesting to note that for the example of computation graphs we restrict ourselves to finitely generated monoids, because the computation only involves a finite number of elements.

3 Instantiation of the Theory

3.1 Layer-Wise Relevance Propagation

[Founded on LRP overview, <https://iphome.hhi.de/samek/pdf/MonXAI19.pdf>]

[Dependance or not on an execution, want to do the computation once or several times]

Layer-Wise Relevance Propagation (LRP) is a technique used to explain the prediction made by a neural network on one of its execution. The idea of the technique is to propagate backwards the contribution (or the *relevance*) of neurons in the network to the final prediction of the model. The general formula is

$$R_j = \sum_k \frac{z_{j,k}}{\sum_j z_{j,k}} R_k.$$

The value R_j is the score of the neuron j , and it is computed by summing over all neurons k in the next layer the term $\frac{z_{j,k}}{\sum_j z_{j,k}} R_k$, where R_k is the relevance of neuron k and $\frac{z_{j,k}}{\sum_j z_{j,k}}$ represents the extent to which neuron j contributed to the relevance of the neuron k .

The normalization term ensures that vanishing phenomenons cannot occur: the sum of the relevances of neurons on a given layer is preserved.

We should now discuss the interpretation of LRP as an instantiation of our model.

We can model LRP with generalized computation graphs. Indeed, the input nodes of the computation graphs are the output neurons of the network (or even the only neuron corresponding to the class predicted, since the value of other neurons is set to zero), the edges of the computation graph are the reversed edges of the neuron and the associated function is the multiplication by the scalar $\frac{z_{j,k}}{\sum_j z_{j,k}}$ (between neurons k and j).

3.2 Quantization

[TODO: Quantization]

4 Conclusion

[TODO]

Computation Graphs Bibliography

[CompGreen et al., 2007] 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*, pages 31–40, Beijing China. ACM.

[CompRamusat, 2022] Ramusat, Y. (2022). *The semiring-based provenance framework for graph databases*. phdthesis, Université Paris sciences et lettres.

[CompWang and Zhao, 2023] Wang, L. and Zhao, J. (2023). *Computation Graph*, pages 149–189. Apress, Berkeley, CA.

5 Extra work - GNN Stability

While investigating the possibility to theoretically model a neural network, I met a Ph.D student that was working on graph neural networks (GNN). GNNs are used to embed graphs in metric spaces and its objective is to extend this technique in order to embed databases.

My contribution in this work was to adapt result on stability to our precise setting, and the hypotheses used in our work even allowed me to improve existing results.

A graph neural network (GNN) is a particular type of neural network that is meant to represent graph-structured data in \mathbb{R}^d while taking into account the relations between the data points. Such representations, or embeddings, are interesting because we can then apply usual machine learning techniques to the transformed data.

However, for such schemes to be meaningful we need to ensure that the embeddings are coherent

5.1 Definition of GNNs

5.2 Formal Framework for Stability

To measure the distance between two graph shift operators, and consequently the distance between the graphs they are derived from, we use $d_{\mathcal{P}}(S, \hat{S})$, noted $\|S - \hat{S}\|_{\mathcal{P}}$ in [Gama et al., 2020], defined as $d_{\mathcal{P}}(S, \hat{S}) = \min_{P \in \mathcal{P}} \|S - P\hat{S}P^T\|$ where \mathcal{P} is the set of permutation matrices. We change the notations to highlight the fact that the quantity depends on the two matrices S, \hat{S} and not on the difference $S - \hat{S}$. The definition of $d_{\mathcal{P}}$ can be extended to general continuous operators $\Psi, \hat{\Psi} : \mathbb{R}^N \rightarrow \mathbb{R}^N$ by $d_{\mathcal{P}}(\Psi, \hat{\Psi}) = \min_{P \in \mathcal{P}} \max_{\|x\|=1} \|P^T \Psi(x) - \hat{\Psi}(P^T x)\|$.

Limitations There are two main challenges to apply the preceding framework. Firstly, computing $d_{\mathcal{P}}(S, \hat{S})$ is more general than determining if the graphs are isomorphic, so it is not feasible for large n in general. (Indeed, if S is the adjacency matrix, then two graphs G and \hat{G} are isomorphic if and only if there exists $P \in \mathcal{P}$ s.t. $S = P\hat{S}P^T$, which is equivalent to $d_{\mathcal{P}}(S, \hat{S}) = 0$.) Secondly, we can only compare matrices that have the same size, i.e. graphs that have the same number of nodes. However, we seek to measure the perturbation induced by the deletion of tuples in the database, which corresponds to node deletion in the graph.

Solution used in this work We consider a database D with associated graph G . Let T be a set of tuples of D , let \hat{D} be the database D without the tuples in T and let \hat{G} be its associated graph. We define their two graph shift operators S and \hat{S} . There are two equivalent points of view for our approach:

- We fill the matrix \hat{S} with rows and columns of 0s so its shape matches the shape of S . Then we consider the permutation σ_0 that matches the remaining nodes in \hat{G} with their original node in G and matches the rows/columns of 0s with the nodes in G that disappeared in \hat{G} . The permutation σ_0 provides us a permutation matrix P_0 and we bound $d_{\mathcal{P}}(S, \hat{S})$ by $\|S - P_0\hat{S}P_0^T\|_{op}$.
- We do not delete the nodes we are supposed to delete, but delete their edges instead. This yields a graph \hat{G}' with exactly the same nodes as G but some missing edges, and we compute $\|S - \hat{S}'\|_{op}$.

5.3 Theoretical results

We take advantage of the fact that we only use filters that are polynomial of order 1 to refine the theorem 4 of [Gama et al., 2020]. The two main improvements that are allowed by this simplification is the fact that the inequality is now a global inequality and there is no dependency in N .

Theorem 2. *Let $\Phi(\cdot, \cdot)$ be a GNN with L layers, less than F features per layer and that only uses filters of the form $H(X) = aX + b$. We assume that the non-linearities σ in Φ are 1-Lipschitz and satisfy $\sigma(0) = 0$. Let $S, \hat{S} \in \mathbb{R}^{N \times N}$ be two symmetric matrices. We note $\|H\|_{\infty} = \max_{H \in \Phi, \lambda \in \sigma(S) \cup \sigma(\hat{S})} |H(\lambda)|$ the*

maximum of the value $|H(\lambda)|$ over all filters H that appear in Φ and all λ that are eigenvalues of S or \hat{S} and we note $A = \max_{H(X)=aX+b \in \Phi} |a|$.

Then, for all $x \in \mathbb{R}^N$, we have:

$$\|\Phi(S, x) - \Phi(\hat{S}, x)\| \leq LA(F\|H\|_{\infty})^{L-1}\|S - \hat{S}\|_{op}\|x\|.$$

There are a few remarks to be done about the theorem. Firstly, the proof provides a more specific bound: we can replace $LA(F\|H\|_{\infty})^{L-1}$ by $B_L = \left(\prod_{i=1}^{L-1} F_i\right) \cdot \sum_{i=1}^L \left[\left(\prod_{j=L-i+1}^{L-1} \|H^{(i)}\|_{\infty}\right) \cdot A^{(L-i)} \cdot \left(\prod_{j=0}^{L-i-1} \|H^{(i)}\|_{\infty}\right) \right]$ where F_i is the number of features at the i -th layer, $\|H^{(i)}\|_{\infty}$ is the maximum of all values $|H^{(i)}(\lambda)|$ for λ ranging over the eigenvalues of S and \hat{S} and $H^{(i)}$ ranging over the filters between the i -th and the $i+1$ -th layer and $A^{(i)}$ is the maximum of the coefficient $|a|$ for all filters taps between the i -th and the $i+1$ -th layer that can be written $H(X) = aX + b$.

This bound is much less readable, but it allows to take into account irregularities between layers. This can make a big difference when the number of layers grows and there is only one layer with a big amount of features or there is only one layer with a big value of $A^{(i)}$.

Secondly, the theorem is wrote for a general couple of matrices (S, \hat{S}) but we can instantiate it with the $d_{\mathcal{P}}$ distance:

Corollary 2.1. Given a GNN Φ satisfying the hypotheses of the theorem 2, the following equality holds:

$$d_{\mathcal{P}}(\Phi(S, \cdot), \Phi(\hat{S}, \cdot)) \leq LA(F\|H\|_{\infty})^{L-1}d_{\mathcal{P}}(S, \hat{S}).$$

Proof. Firstly, we show how to deduce the corollary from the theorem. We consider $P_0 \in \mathcal{P}$ such that $d_{\mathcal{P}}(S, \hat{S}) = \|S - P_0 \hat{S} P_0^T\|_{op}$. The theorem then provides the inequality, for $\|x\| = 1$,

$$\begin{aligned} \|\Phi(S, x) - \Phi(P_0 \hat{S} P_0^T, x)\| &\leq LA(F\|H\|_{\infty})^{L-1} \|S - P_0 \hat{S} P_0^T\|_{op} \\ &= LA(F\|H\|_{\infty})^{L-1} d_{\mathcal{P}}(S, \hat{S}) \end{aligned}$$

and since

$$\begin{aligned} d_{\mathcal{P}}(\Phi(S, \cdot), \Phi(\hat{S}, \cdot)) &= \min_{P \in \mathcal{P}} \max_{\|x\|=1} \|P^T \Phi(S, x) - \Phi(\hat{S}, P^T x)\| \\ &= \min_{P \in \mathcal{P}} \max_{\|x\|=1} \|\Phi(S, x) - P \Phi(\hat{S}, P^T x)\| \\ &= \min_{P \in \mathcal{P}} \max_{\|x\|=1} \|\Phi(S, x) - \Phi(P \hat{S} P^T, x)\| \\ &\leq \max_{\|x\|=1} \|\Phi(S, x) - \Phi(P_0 \hat{S} P_0^T, x)\|, \end{aligned}$$

then we have for the x that reaches the max:

$$\begin{aligned} d_{\mathcal{P}}(\Phi(S, \cdot), \Phi(\hat{S}, \cdot)) &\leq \|\Phi(S, x) - \Phi(P \hat{S} P^T, x)\| \\ &\leq LA(F\|H\|_{\infty})^{L-1} d_{\mathcal{P}}(S, \hat{S}). \end{aligned}$$

We now prove the theorem 2. The method used is very close from what is done in [Gama et al., 2020]. We show a more precise theorem by using the bound

$$B_L = \left(\prod_{i=1}^{L-1} F_i \right) \cdot \sum_{i=1}^L \left[\left(\prod_{j=L-i+1}^{L-1} \|H^{(i)}\|_{\infty} \right) \cdot A^{(L-i)} \cdot \left(\prod_{j=0}^{L-i-1} \|H^{(i)}\|_{\infty} \right) \right]$$

rather than $LA(F\|H\|_{\infty})^{L-1}$. We check that $B_L \leq LA(F\|H\|_{\infty})^{L-1}$ by bounding all the coefficient that depends on a layer by their global counterpart ($A^{(L-i)} \leq A$, $F_i \leq F$ and $\|H^{(i)}\|_{\infty} \leq \|H\|_{\infty}$).

We begin with a lemma:

Lemma 3. We consider a GNN Φ satisfying the hypotheses of 2, S a symmetric matrix and x a vector. Then we have:

$$\|\Phi(S, x)\| \leq \left(\prod_{i=0}^{L-1} F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_{\infty} \right) \|x\|$$

Indeed, we show this lemma by induction on the number of layers of Φ . For a GNN Φ with only one layer, since the input and output layers can only contain one feature, then Φ is of the form $\Phi(S, x) = \sigma(H(S)x)$. Consequently,

$$\begin{aligned} \|\Phi(S, x)\| &\leq \|H(S)x\| \\ &\leq \|H^{(0)}\|_{\infty} \|x\| \end{aligned}$$

where we used $\|H(S)\|_{op} \leq \|H^{(0)}\|$ which directly follows from the definition of $\|H^{(0)}\|$. This proves that the lemma holds for the initial case.

We now consider $L \in \mathbb{N}^*$ and we assume that the lemma is true for all GNNs with L layers satisfying the hypotheses.

Let $\Phi(\cdot, \cdot)$ be a GNN with $L + 1$ layers satisfying the hypotheses, let S be a symmetric matrix and let x be a vector.

By definition of the GNNs in this work, Φ has the form shown in figure 6. We define the function $\Phi_i^{(L)}(S, x)$ that outputs the vector contained in the neuron n_{Li} (the i -th neuron of the L -th layer, see figure 6) during the execution of Φ . The function $\Phi_i^{(L)}$ is a GNN itself, with only L layers.

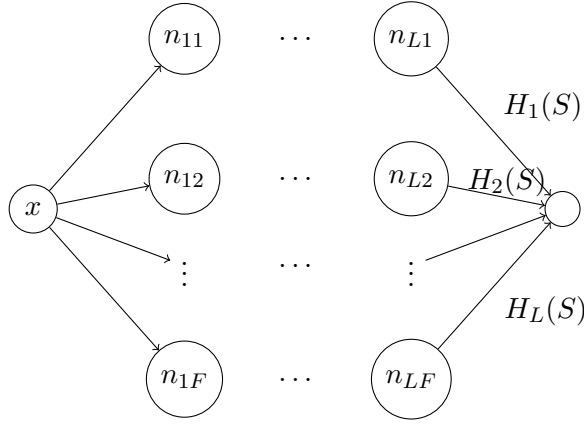


Figure 6: Shape of Φ .

By definition of Φ , we can then write:

$$\Phi(S, x) = \sigma \left(\sum_{i=1}^{F_L} H_i(S) \Phi_i^{(L)}(S, x) \right).$$

As a consequence we have the inequalities:

$$\begin{aligned} \|\Phi(S, x)\| &\leq \sum_{i=1}^{F_L} \|H_i(S) \Phi_i^{(L)}(S, x)\| \\ &\leq F_L \|H^{(L)}\|_{\infty} \|\Phi_i^{(L)}(S, x)\| \\ &\leq F_L \|H^{(L)}\|_{\infty} \left(\prod_{i=0}^{L-1} F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_{\infty} \right) \|x\| \\ &= \left(\prod_{i=0}^L F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_{\infty} \right) \|x\| \end{aligned}$$

where we used the induction hypothesis to bound $\|\Phi_i^{(L)}(S, x)\|$. This concludes the proof of the lemma.

We now prove theorem 2 by induction on the number of layers L as well. Firstly, if we consider a GNN with 1 layer, then $\Phi(S, x)$ is $\sigma(H(S)x)$ with $H(S) = aS + bI_N$. Given two symmetric matrices S and \hat{S} and a vector $x \in \mathbb{R}^N$, the theorem is written: $\|\sigma(H(S)x) - \sigma(H(\hat{S})x)\| \leq B_1 \|S - \hat{S}\|_{op} \|x\|$ with $B_1 = A^{(1)} = |a|$.

This holds since:

$$\begin{aligned} \|\sigma(H(S)x) - \sigma(H(\hat{S})x)\| &\leq \|H(S)x - H(\hat{S})x\| \\ &= \|a(S - \hat{S})x\| \\ &\leq |a| \|S - \hat{S}\|_{op} \|x\|. \end{aligned}$$

We now consider $L \in \mathbb{N}^*$ and we assume that the theorem is true for all GNNs with L layers satisfying the hypotheses. I.e., if Ψ is such a GNN, $\|\Psi(S, x) - \Psi(\hat{S}, x)\| \leq B_L \|S - \hat{S}\|_{op} \|x\|$.

We consider a GNN Φ with $L + 1$ layers and use the exact same notations as in the proof of the lemma.

We begin by bounding $\Phi(S, x) - \Phi(\hat{S}, x)$ with the $\Phi_i^{(L)}$:

$$\begin{aligned}
\|\Phi(S, x) - \Phi(\hat{S}, x)\| &= \left\| \sum_{i=1}^{F_L} H_i(S) \sigma(\Phi_i^{(L)}(S, x)) - H_i(\hat{S}) \sigma(\Phi_i^{(L)}(\hat{S}, x)) \right\| \\
&\leq \sum_{i=1}^{F_L} \|H_i(S) \sigma(\Phi_i^{(L)}(S, x)) - H_i(\hat{S}) \sigma(\Phi_i^{(L)}(\hat{S}, x))\| \\
&\leq \sum_{i=1}^{F_L} \|H_i(S) (\sigma(\Phi_i^{(L)}(S, x)) - \sigma(\Phi_i^{(L)}(\hat{S}, x)))\| \\
&\quad + \sum_{i=1}^{F_L} \|(H_i(S) - H_i(\hat{S})) \sigma(\Phi_i^{(L)}(\hat{S}, x))\|
\end{aligned} \tag{1}$$

We should now bound both terms separately. We begin with $\|H_i(S) (\sigma(\Phi_i^{(L)}(S, x)) - \sigma(\Phi_i^{(L)}(\hat{S}, x)))\|$. Using the definition of $\|H^{(L)}\|_\infty$ and the induction hypothesis, we get:

$$\begin{aligned}
\|H_i(S) (\sigma(\Phi_i^{(L)}(S, x)) - \sigma(\Phi_i^{(L)}(\hat{S}, x)))\| &\tag{2} \\
&\leq \|H^{(L)}\|_\infty \|\sigma(\Phi_i^{(L)}(S, x)) - \sigma(\Phi_i^{(L)}(\hat{S}, x))\| \\
&\leq \|H^{(L)}\|_\infty \|\Phi_i^{(L)}(S, x) - \Phi_i^{(L)}(\hat{S}, x)\| \\
&\leq \|H^{(L)}\|_\infty B_L \|S - \hat{S}\|_{op} \|x\|.
\end{aligned} \tag{3}$$

For the second term, using the definition of $A^{(L)}$ and the lemma, we get:

$$\begin{aligned}
\|(H_i(S) - H_i(\hat{S})) \sigma(\Phi_i^{(L)}(\hat{S}, x))\| &\tag{4} \\
&\leq \|H_i(S) - H_i(\hat{S})\|_{op} \|\sigma(\Phi_i^{(L)}(\hat{S}, x))\| \\
&\leq A^{(L)} \|S - \hat{S}\|_{op} \|\Phi_i^{(L)}(\hat{S}, x)\| \\
&\leq A^{(L)} \|S - \hat{S}\|_{op} \left(\prod_{i=0}^{L-1} F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_\infty \right) \|x\|.
\end{aligned}$$

Combining the bounds given by inequalities 3 and 4 in the inequality 1 yields

$$\begin{aligned}
\|\Phi(S, x) - \Phi(\hat{S}, x)\| &\leq \sum_{i=1}^{F_L} \|H^{(L)}\|_\infty B_L \|S - \hat{S}\|_{op} \|x\| \\
&\quad + \sum_{i=1}^{F_L} A^{(L)} \left(\prod_{i=0}^{L-1} F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_\infty \right) \|S - \hat{S}\|_{op} \|x\|
\end{aligned} \tag{5}$$

$$\leq \left(F_L \|H^{(L)}\|_\infty B_L + F_L A^{(L)} \left(\prod_{i=0}^{L-1} F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_\infty \right) \right) \|S - \hat{S}\|_{op} \|x\|. \tag{6}$$

Given that $F_L \|H^{(L)}\|_\infty B_L + F_L A^{(L)} \left(\prod_{i=0}^{L-1} F_i \right) \left(\prod_{i=0}^{L-1} \|H^{(i)}\|_\infty \right) = B_{L+1}$, this can be rewritten:

$$\|\Phi(S, x) - \Phi(\hat{S}, x)\| \leq B_{L+1} \|S - \hat{S}\|_{op} \|x\|, \tag{7}$$

which shows that the theorem holds for GNN with $L + 1$ layers and concludes the proof. \square

5.4 Node embedding stability

In this section, we discuss two corollaries of the theorem 2 that translate the original result in two stability properties at the node embedding level. Like for the theorem, in the bounds we can replace $LA(F\|H\|_\infty)^{L-1}$ by B_L .

The node embeddings are defined in section ??.

Corollary 3.1. We consider a GNN Φ satisfying the hypotheses of theorem 2, and a couple of graph G, \hat{S} with their associated shift operators S, \hat{S} . We assume that all the columns of the initialisation matrices $X^{(0)}$ are normalized. If we consider the i -th nodes of G and \hat{G} , noted v_i and v'_i and assume that their k -hop are equal, then we have the following property:

$$\begin{aligned} \|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\| \\ \leq \sqrt{d} B_L^{(k)} \|S - \hat{S}\|_{op} \end{aligned} \quad (8)$$

where we define

$$B_L^{(k)} = \left(\prod_{i=1}^{L-1} F_i \right) \cdot \sum_{i=k+1}^L \left[\left(\prod_{j=L-i+1}^{L-1} \|H^{(i)}\|_\infty \right) \cdot A^{(L-i)} \cdot \left(\prod_{j=0}^{L-i-1} \|H^{(i)}\|_\infty \right) \right].$$

We can bound $B_L^{(k)}$ is 0 if $L \geq k$ and it can be bounded by $(L-k)A(F\|H\|_\infty)^{L-1}$ otherwise.

The hypothesis of normalization is not necessary, it is there so the statement is more readable.

Noting C_i the columns of X , we could remove the hypothesis and replace \sqrt{d} by $\sqrt{\sum_{i=1}^d \|C_i\|^2}$.

Corollary 3.2. We consider a GNN Φ satisfying the hypotheses of theorem 2, and a couple of graph G, \hat{S} with their associated shift operators S, \hat{S} . We assume that all the columns of the initialisation matrices $X^{(0)}$ are normalized. We can control the total squared distances between the embeddings:

$$\sum_{i=1}^N \|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\|^2 \leq dM^2 \quad (9)$$

where M is a bound of theorem 2, i.e. $M = LA(F\|H\|_\infty)^{L-1}\|S - \hat{S}\|_{op}$ or $M = B_L\|S - \hat{S}\|_{op}$.

This last property is interesting because it ensures that for most nodes $\|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\|^2 \leq \frac{\sqrt{d}M}{\sqrt{N}}$. Since the bound dM^2 does not depend on N , this means that using a GNN with fixed parameters on large database gives reliable embeddings for the majority.

The worst case distance for two embeddings can still be relatively large, it is bounded by

$$\begin{aligned} \|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\| \\ \leq \sqrt{\sum_{i=1}^N \|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\|^2} \\ \leq \sqrt{d}M, \end{aligned}$$

but this cannot happen for several nodes at the same time.

Proof. We prove the corollary 3.2. We consider Φ, S, \hat{S} and $X^{(0)}$ as in the statement of the corollary.

By definition of the embedding, $\|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\| = \|L_i\|^2$ where L_i is the i -th line of the matrix $\Phi(S, X^{(0)}) - \Phi(\hat{S}, X^{(0)})$. Therefore,

$$\begin{aligned} \sum_{i=1}^N \|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\|^2 &= \sum_{i=1}^N \|L_i\|^2 \\ &= \sum_{j=1}^d \|C_j\|^2 \\ &= \sum_{j=1}^d \|\Phi(S, C_j^{(0)}) - \Phi(\hat{S}, C_j^{(0)})\|^2 \end{aligned}$$

where C_j is the j -th column of the matrix $\Phi(S, X^{(0)}) - \Phi(\hat{S}, X^{(0)})$ and $C_j^{(0)}$ is the j -th column of $X^{(0)}$.

Since theorem 2 ensures that $\|\Phi(S, C_j^{(0)}) - \Phi(\hat{S}, C_j^{(0)})\| \leq M$, we deduce:

$$\sum_{i=1}^N \|\Phi(S, X^{(0)})_{v_i} - \Phi(\hat{S}, X^{(0)})_{v'_i}\|^2 \leq dM^2.$$

□

5.5 Instantiation for Databases

Definition 2. *Error relative to the perturbation* Let \mathbb{D}_1 and $\mathbb{D}_2 = \pi(\mathbb{D}_1)$ be a database and a perturbed database generated from \mathbb{D}_1 . We define the error relative to the perturbation as $\varepsilon = \|S_{\Gamma(\mathbb{D}_1)} - S_{\Gamma(\mathbb{D}_2)}\|_{op}$, where the matrix $S_{\Gamma(\mathbb{D}_2)}$ is built such that the rows and columns that represent an element of \mathbb{D}_2 corresponds to the rows and columns of its original counterpart in \mathbb{D}_1 .

Corollary 3.3. We consider a GNN Φ satisfying the hypotheses of theorem 2, a database \mathbb{D}_1 with a perturbed database \mathbb{D}_2 and a node v in $\Gamma(\mathbb{D}_1)$ that is identified to the node v' in $\Gamma(\mathbb{D}_2)$. Assuming that all the columns of the initialisation matrices $X^{(0)}$ are normalized, we have the following property:

$$\|\Phi(S_{\Gamma(\mathbb{D}_1)}, X^{(0)})_v - \Phi(P_0^T S_{\Gamma(\mathbb{D}_2)} P_0, P_0^T \hat{X}^{(0)})_{v'}\| \quad (10)$$

$$\leq \sqrt{d} B_L \|S_{\Gamma(\mathbb{D}_1)} - S_{\Gamma(\mathbb{D}_2)}\|_{op} \|x\| \quad (11)$$

In equation ??, v and v' being identified means that they represent the same row in $S_{\Gamma(\mathbb{D}_1)}$ and $S_{\Gamma(\mathbb{D}_2)}$ respectively.

Definition 3 (Downstream tasks stability (fig: ??)). *The DST stability translate the capacity of two embeddings of the same element or value, generated from the perturbed and the clean databases, to gives the same result when they are used as input for downstream tasks.*

This is schematised by the last line of the figure, ??. In the last line of the figure, are compared the results of downs stream tasks computed from \mathbb{D} and the ones obtained using $\pi(\mathbb{D})$. In this figure, we are not comparing from which embedding we are retrieving the highest accuracy, but if the results obtained from the two structures are similar.

GNN Bibliography

[GnnGama et al., 2020] Gama, F., Bruna, J., and Ribeiro, A. (2020). Stability properties of graph neural networks. *IEEE Transactions on Signal Processing*, 68:5680–5695.