



SELF STUDY  
BASED ON MY UNDERSTANDING OF THE SUBJECT

---

# Graph Neural Network

---

Compilation of my study materials

**Author:** Dr. Md Arafat Hossain Khan

**Last updated :** December 6, 2022

---

# Contents

<b>1</b>	<b>Fundamentals</b>	<b>1</b>
1.1	Graph Neural Network Model Mathematics . . . . .	1
1.1.1	Labels of a graph . . . . .	2
1.1.2	Local transition and output functions . . . . .	2
1.1.3	Model Variation . . . . .	3
1.1.4	Global transition and output functions . . . . .	5
1.2	Graph Neural Network Model Implementation . . . . .	6
1.2.1	Computation of the State . . . . .	6
1.2.2	The Learning Algorithm . . . . .	8
1.2.3	Transition and Output Function Implementations . . . . .	9
<b>A</b>	<b>Contraction mapping theorem</b>	<b>10</b>
A.1	Contraction mapping . . . . .	10
A.2	Contraction mapping theorem . . . . .	12
<b>B</b>	<b>Classic iterative methods</b>	<b>15</b>
B.1	Jacobi and Gauss-Seidel iterative method . . . . .	15
<b>C</b>	<b>Important Algorithms</b>	<b>16</b>
C.1	Almeida-Pineda algorithm . . . . .	16
	<b>Index</b>	<b>17</b>
	<b>Bibliography</b>	<b>18</b>

---

# Fundamentals

---

*Artificial intelligence is a tool, not a threat. Artificial intelligence is a tool, not a threat. Artificial intelligence is a tool, not a threat.*

*Rodney Brooks*

---

1.1	Graph Neural Network Model Mathematics . . . . .	1
1.1.1	Labels of a graph . . . . .	2
1.1.2	Local transition and output functions . . . . .	2
1.1.3	Model Variation . . . . .	3
1.1.4	Global transition and output functions . . . . .	5
1.2	Graph Neural Network Model Implementation . . . . .	6
1.2.1	Computation of the State . . . . .	6
1.2.2	The Learning Algorithm . . . . .	8
1.2.3	Transition and Output Function Implementations . . . . .	9
A.1	Contraction mapping . . . . .	10
A.2	Contraction mapping theorem . . . . .	12
B.1	Jacobi and Gauss-Seidel iterative method . . . . .	15
C.1	Almeida-Pineda algorithm . . . . .	16

---

## 1.1 Graph Neural Network Model Mathematics

This note is fundametally inspired by the original paper of graph neural network [1] and based on my undertsanding of the subject.

### 1.1.1 Labels of a graph

A graph  $G$  is a pair  $(N, E)$ , where  $N = \{n_1, n_2, \dots, n_{|N|-1}, n_{|N|}\}$  is the set of nodes and  $E = \{e_1, e_2, \dots, e_{|E|-1}, e_{|E|}\}$  is the set of edges.  $\mathcal{N}(n)$  stands for the neighbors of  $n \in N$ . If an *undirected* edge  $e$  connects node  $x$  and  $y$ , we can also denote  $e$  as  $(x, y)$  or  $(y, x)$ . All nodes are embedded into some euclidean space of dimension  $d_N \in \mathbb{N}$  and all edges are embedded into some euclidean space of dimension  $d_E \in \mathbb{N}$ , e.g.

label (embedding) for node  $n$  is defined as  $\ell_n : n \mapsto \mathbb{R}^{d_N}$ , where  $n \in N$

label (embedding) for edge  $e$  is defined as  $\ell_e : e \mapsto \mathbb{R}^{d_E}$ , where  $e \in E$

The notion of labels can be extended to multiple nodes and edges. Suppose, we take a node set,  $\mathfrak{N} = \{\mathbf{n}_1, \mathbf{n}_2, \dots, \mathbf{n}_{|\mathfrak{N}|-1}, \mathbf{n}_{|\mathfrak{N}|}\} \subseteq N$ . Then we define,

$$\ell_{\mathfrak{N}} : \mathbb{R}^{d_N \times |\mathfrak{N}|} \mapsto \mathbb{R}^{d_N} \text{ as } (\ell_{\mathbf{n}_1}, \ell_{\mathbf{n}_2}, \dots, \ell_{\mathbf{n}_{|\mathfrak{N}|-1}}, \ell_{\mathbf{n}_{|\mathfrak{N}|}}) \mapsto \mathbb{R}^{d_N}$$

Here, we want  $\ell_{\mathfrak{N}}$  to be permutation invariant. Similarly, we take an edge set,  $\mathfrak{E} = \{\mathbf{e}_1, \mathbf{e}_2, \dots, \mathbf{e}_{|\mathfrak{E}|-1}, \mathbf{e}_{|\mathfrak{E}|}\} \subseteq E$ . Then we define,

$$\ell_{\mathfrak{E}} : \mathbb{R}^{d_E \times |\mathfrak{E}|} \mapsto \mathbb{R}^{d_E} \text{ as } (\ell_{\mathbf{e}_1}, \ell_{\mathbf{e}_2}, \dots, \ell_{\mathbf{e}_{|\mathfrak{E}|-1}}, \ell_{\mathbf{e}_{|\mathfrak{E}|}}) \mapsto \mathbb{R}^{d_E}$$

Here, we want  $\ell_{\mathfrak{E}}$  to be permutation invariant. We extend the notion of the labels even further for the entire graph by the following permutation invariant map,

$$\ell : \mathbb{R}^{d_N \times |N|} \times \mathbb{R}^{d_E \times |E|} \mapsto \mathbb{R}^D \text{ as } (\ell_{n_1}, \ell_{n_2}, \dots, \ell_{n_{|N|-1}}, \ell_{n_{|N|}}, \ell_{e_1}, \ell_{e_2}, \dots, \ell_{e_{|E|-1}}, \ell_{e_{|E|}}) \mapsto \mathbb{R}^D$$

where  $D \in \mathbb{N}$ .

### 1.1.2 Local transition and output functions

Let us take  $n \in N$  from the graph  $G = (N, E)$  and define the following,

- $\ell_n \in \mathbb{R}^{d_N}$ : node label of  $n \in N$ .
- $\ell_{\mathcal{N}(n)} \in \mathbb{R}^{d_N}$ : label of the neighborhood  $\mathcal{N}(n) = \{\nu_1, \nu_2, \dots, \nu_{|\mathcal{N}(n)|-1}, \nu_{|\mathcal{N}(n)|}\} \subseteq N$ .
- $\mathcal{E}(n)$ : edges connected to  $n$  i.e  $\mathcal{E}(n) = \{(\nu_1, n), (\nu_2, n), \dots, (\nu_{|\mathcal{N}(n)|-1}, n), (\nu_{|\mathcal{N}(n)|}, n)\}$ .
- $\ell_{\mathcal{E}(n)}$ : edge label of  $\mathcal{E}(n)$ .

Now we can define something called the *state* of  $n$ , denoted as  $x_n \in \mathbb{R}^s$ , which captures the the local information of  $n$  using its own node label  $\ell_n \in \mathbb{R}^{d_N}$ , node label of the neighbors  $\ell_{\mathcal{N}(n)} \in \mathbb{R}^{d_N}$ , connected edges label  $\ell_{\mathcal{E}(n)}$ , and the *state* of the neighbors  $x_{\mathcal{N}(n)} \in \mathbb{R}^s$ . Note that,  $x_{\mathcal{N}(n)}$  is a permutation invariant map, defined by

$$x_{\mathcal{N}(n)} : \mathbb{R}^{s \times |\mathcal{N}(n)|} \mapsto \mathbb{R}^s \text{ as } (x_{\nu_1}, x_{\nu_2}, \dots, x_{\nu_{|\mathcal{N}(n)|-1}}, x_{\nu_{|\mathcal{N}(n)|}}) \mapsto \mathbb{R}^s.$$

Then we define the *local transition function*, that expresses the dependence of a node on its neighborhood as

$$f_w : \mathbb{R}^{d_N} \times \mathbb{R}^{d_E} \times \mathbb{R}^s \times \mathbb{R}^{d_N} \mapsto \mathbb{R}^s \text{ as } (\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}, \ell_{\mathcal{N}(n)}) \mapsto \mathbb{R}^s.$$

$$\begin{aligned} \text{where } & (\ell_{(\nu_1, n)}, \ell_{(\nu_2, n)}, \dots, \ell_{(\nu_{|\mathcal{N}(n)|-1}, n)}, \ell_{(\nu_{|\mathcal{N}(n)|}, n)}) := \ell_{\mathcal{E}(n)} \\ & (x_{\nu_1}, x_{\nu_2}, \dots, x_{\nu_{|\mathcal{N}(n)|-1}}, x_{\nu_{|\mathcal{N}(n)|}}) := x_{\mathcal{N}(n)} \\ & (\ell_{\nu_1}, \ell_{\nu_2}, \dots, \ell_{\nu_{|\mathcal{N}(n)|-1}}, \ell_{\nu_{|\mathcal{N}(n)|}}) := \ell_{\mathcal{N}(n)} \end{aligned}$$

And we also express *the local output function* that describes how the output is produced as follows,

$$g_w : \mathbb{R}^s \times \mathbb{R}^{d_N} \mapsto \mathbb{R}^r \quad \text{as} \quad (x_n, \ell_n) \mapsto \mathbb{R}^r$$

Hence, we have the following model,

$$\begin{aligned} \text{local transition function, } x_n &= f_w(\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}, \ell_{\mathcal{N}(n)}) \\ \text{local output function, } o_n &= g_w(x_n, \ell_n) \end{aligned}$$

### 1.1.3 Model Variation

#### Minimal model

##### Definition 1: Minimal Model

A model is said to be minimal if it has the smallest number of variables while retaining the same computational power.

Note that, one may wish to remove the labels  $\ell_{\mathcal{N}(n)}$ , since they include information that is implicitly contained in  $x_{\mathcal{N}(n)}$ . Thus one can hunt for a minimal model. It can be shown that the minimal model exists but not unique.

#### Extension of neighborhood

The neighborhood could contain nodes that are two or more links away from  $n$  capturing a bigger local region.

#### Directed and mixed graph

For directed or mixed graph we may find a variation of model definition by incorporating the following map into the model definition,

$$\begin{aligned} \mathfrak{d}_{\mathcal{E}(n)} : \mathcal{E}(n) &\mapsto \{0, 1\} \quad \text{as} \\ \mathfrak{d}_{\mathcal{E}(n)}(e) = \mathfrak{d}_e &= \begin{cases} 1 & \text{if } e \in \mathcal{E}(n) \text{ directs towards } n \\ 0 & \text{if } e \in \mathcal{E}(n) \text{ comes out of } n \end{cases} \end{aligned}$$

Thus the definition of *local transition function* changes as follows,

$$f_w : \mathbb{R}^{d_N} \times \mathbb{R}^{d_E} \times \mathbb{R}^s \times \mathbb{R}^{d_N} \times \{0, 1\}^{|\mathcal{N}(n)|} \mapsto \mathbb{R}^s \quad \text{as} \quad (\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}, \ell_{\mathcal{N}(n)}, \mathfrak{d}_{\mathcal{E}(n)}) \mapsto \mathbb{R}^s.$$

where

$$\begin{aligned} (\ell_{(\nu_1, n)}, \ell_{(\nu_2, n)}, \dots, \ell_{(\nu_{|\mathcal{N}(n)|-1}, n)}, \ell_{(\nu_{|\mathcal{N}(n)|}, n)}) &:= \ell_{\mathcal{E}(n)} \\ (x_{\nu_1}, x_{\nu_2}, \dots, x_{\nu_{|\mathcal{N}(n)|-1}}, x_{\nu_{|\mathcal{N}(n)|}}) &:= x_{\mathcal{N}(n)} \\ (\ell_{\nu_1}, \ell_{\nu_2}, \dots, \ell_{\nu_{|\mathcal{N}(n)|-1}}, \ell_{\nu_{|\mathcal{N}(n)|}}) &:= \ell_{\mathcal{N}(n)} \\ (\mathfrak{d}_{(\nu_1, n)}, \mathfrak{d}_{(\nu_2, n)}, \dots, \mathfrak{d}_{(\nu_{|\mathcal{N}(n)|-1}, n)}, \mathfrak{d}_{(\nu_{|\mathcal{N}(n)|}, n)}) &:= \mathfrak{d}_{\mathcal{E}(n)} \end{aligned}$$

However, unless explicitly stated, all the results proposed in this paper hold also for directed graphs and for graphs with mixed directed and undirected links.

## Node specific local transition and output functions

Moreover, each node can have its own *local transition function* and *the local output function* definition,

$$\begin{aligned} \text{local transition function, } x_n &= f_{w_n}^{(n)}(\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}, \ell_{\mathcal{N}(n)}) \\ \text{local output function, } o_n &= g_{w_n}^{(n)}(x_n, \ell_n) \end{aligned}$$

However, for the sake of simplicity, our analysis will consider a particular model where all the nodes share the same implementation.

## Positional and nonpositional graphs

Nonpositional graphs are those described so far. Positional graphs differ since a unique integer identifier is assigned to each neighbors of a node to indicate its logical position.

### Definition 2: Positional graphs

If for a graph  $G = (N, E)$  and for each  $n \in N$ , there exists,

$$\rho_n : \mathcal{N}(n) \hookrightarrow [|N|]$$

then  $G$  is called a positional graph. Note that the symbol  $\hookrightarrow$  means an injection and  $|N|$  means the set  $\{1, 2, \dots, |N|\}$ .

For  $n \in N$ ,

$$\rho_n(u) = i \implies u \text{ is the } i\text{-th neighbor of } n.$$

An example of this assignment can be such that  $\rho_n$  might enumerate the neighbors of a node following a clockwise ordering convention. Notice that for nodes  $n_j$  and  $n_k$  we may have  $\rho_{n_j} = \rho_{n_k}$  which implies that this is only a relative positional assignment. Note that, for a complete graph,  $\rho_n$  is a bijection for all  $n$ , on the other hand for a path graph with at least 3 nodes  $\rho_n$  is an injection not a surjection for any  $n$ .

Now,  $f_w$  will take this positional information. Here is how it is done. Suppose, for a graph  $G = (N, E)$ ,

$$M := \max_{n,u} \rho_n(u)$$

Then  $\forall n \in N$ , we make  $x_{\mathcal{N}(n)} = (y_1, y_2, \dots, y_M)$  where,

$$y_i = \begin{cases} x_u & \text{if } \rho_n(u) = i \implies u \text{ is the } i\text{-th neighbor of } n \\ x_0 & \text{if there is no } i\text{-th neighbor of } n \end{cases}$$

Here  $x_0$  is some predefined null state. In the same way  $\ell_{\mathcal{E}(n)}$  and  $\ell_{\mathcal{N}(n)}$  are modified as well.

## Positional and non-positional form

Note that in general,

$$\text{local transition function, } x_n = f_w(\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}, \ell_{\mathcal{N}(n)})$$

For non-positional graph we assume a significant Simplification of the model which turned out to be useful,

$$\text{local transition function, } x_n = \sum_{\nu_i \in \mathcal{N}(n)} h_w(\ell_n, \ell_{(\nu_i, n)}, x_{\nu_i}, \ell_{\nu_i})$$

These two representations are called *positional form* and *non-positional form* respectively.

### 1.1.4 Global transition and output functions

Note that we have *local transition functions* for each node of the graph  $G = (N, E)$ , where  $N = \{n_1, n_2, \dots, n_{|N|-1}, n_{|N|}\}$  is the set of nodes and  $E = \{e_1, e_2, \dots, e_{|E|-1}, e_{|E|}\}$  is the set of edges.

$$\begin{aligned} x_{n_1} &= f_w(\ell_{n_1}, \ell_{\mathcal{E}(n_1)}, x_{\mathcal{N}(n_1)}, \ell_{\mathcal{N}(n_1)}) \\ x_{n_2} &= f_w(\ell_{n_2}, \ell_{\mathcal{E}(n_2)}, x_{\mathcal{N}(n_2)}, \ell_{\mathcal{N}(n_2)}) \\ &\vdots \\ x_{n_{|N|}} &= f_w(\ell_{n_{|N|}}, \ell_{\mathcal{E}(n_{|N|})}, x_{\mathcal{N}(n_{|N|})}, \ell_{\mathcal{N}(n_{|N|})}) \end{aligned}$$

We define a *global transition function*,  $F_w$  that takes the graph  $G$  as input and returns the state  $x_n$  for each  $n \in N$ . We represent the system of above equations in the following compact fashion,

$$x = F_w(x, \ell)$$

Now, observe the *local output functions*,

$$\begin{aligned} o_{n_1} &= g_w(x_{n_1}, \ell_{n_1}) \\ o_{n_2} &= g_w(x_{n_2}, \ell_{n_2}) \\ &\vdots \\ o_{n_{|N|}} &= g_w(x_{n_{|N|}}, \ell_{n_{|N|}}) \end{aligned}$$

We define a *global output function*,  $G_w$  that takes the graph  $G$  as input and returns the output  $o_n$  for each  $n \in N$ . We represent the system of above equations in the following compact fashion,

$$o = F_w(x, \ell_N)$$

At this point we have to make sure that such *global transition function* and *global output function* ensures the existence and uniqueness of the solution. Moreover, we also need to figure out a method of computation for the solution.

## Existence and uniqueness of global transition and output functions

Let us observe that under standard topology every euclidean space is a complete metric space. Thus if we can ensure that  $F_w$  is a contraction mapping with respect to the states then using *contraction mapping theorem*, Appendix A, we conclude that there exists unique solution to the states of the nodes. We will enforce the contraction mapping property by an appropriate implementation of the transition function.

## 1.2 Graph Neural Network Model Implementation

In order to implement the GNN model, the following items must be provided:

1. Computation of the State. That means a method to solve,

$$\begin{aligned} \text{local transition function, } x_n &= f_w(\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}, \ell_{\mathcal{N}(n)}) \\ \text{local output function, } o_n &= g_w(x_n, \ell_n) \end{aligned}$$

2. The Learning Algorithm. That means a learning algorithm to adapt  $f_w$  and  $g_w$  using examples from the training data set.
3. Transition and Output Function Implementations. That means an implementation of  $f_w$  and  $g_w$ .

### 1.2.1 Computation of the State

*Contraction mapping theorem*, Theorem A.1, does not only ensure the existence and the uniqueness of the solution of  $x = F_w(x, \ell)$  but it also suggests the following classic iterative scheme for computing the state:

$$x(t+1) = F_w(x(t), \ell).$$

We see the dynamical system converges exponentially fast to the solution of

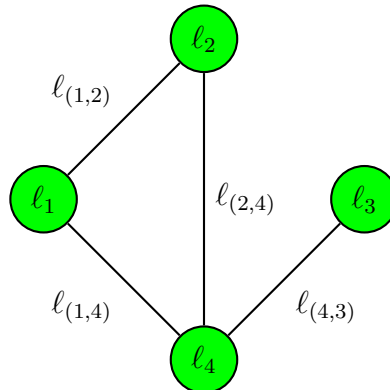
$$\begin{aligned} x &= F_w(x, \ell) \\ o &= G_w(x, \ell_N) \end{aligned}$$

for any initial value  $x(0)$ , Theorem A.1. We can, therefore, think of  $x(t)$  as the state that is updated by the transition function  $F_w$ . In fact,  $x(t+1) = F_w(x(t), \ell)$  implements the Jacobi iterative method for solving nonlinear equations [2]. Thus, the outputs and the states can be computed by iterating,

$$\begin{aligned} x_n(t+1) &= f_w(\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}(t), \ell_{\mathcal{N}(n)}) \\ o_n(t) &= g_w(x_n(t), \ell_n) \end{aligned}$$

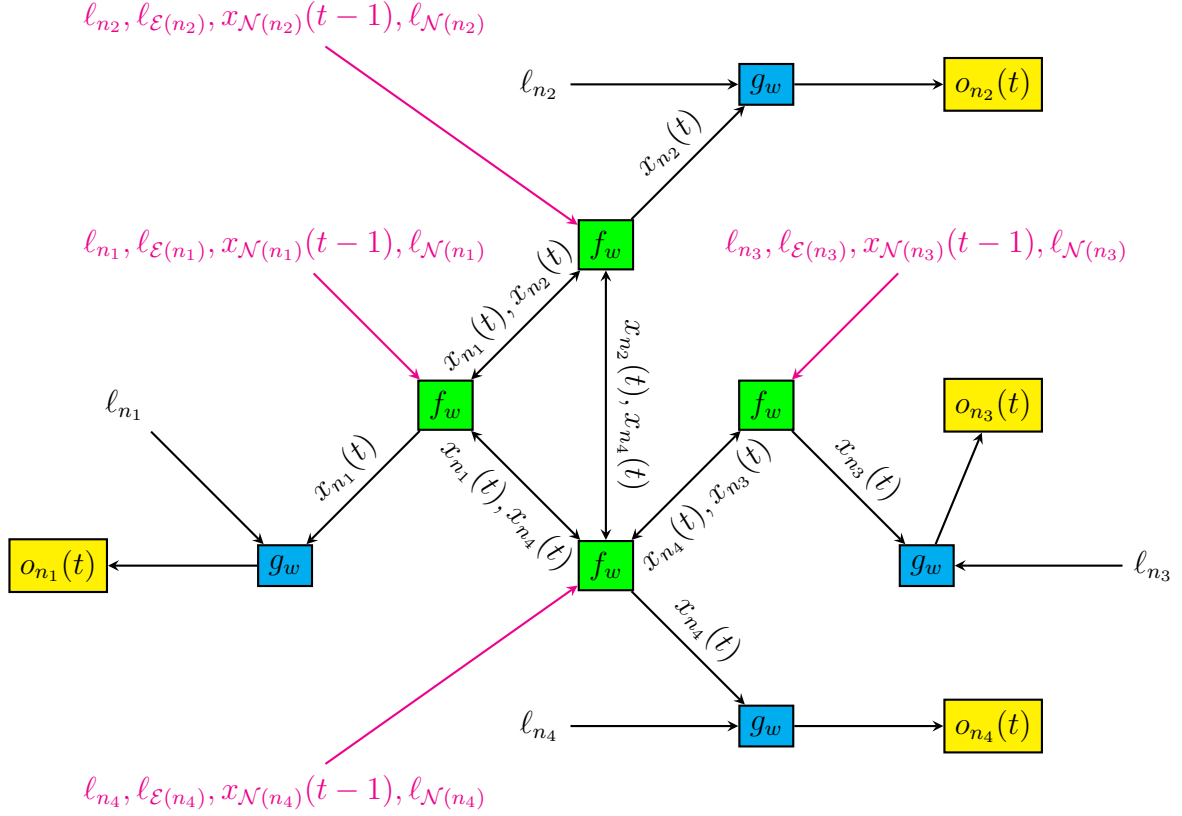
#### Example

We already discussed the variations for directed, and mixed graph. Thus without loss of generality of discussion, we can consider an undirected graph as follows,

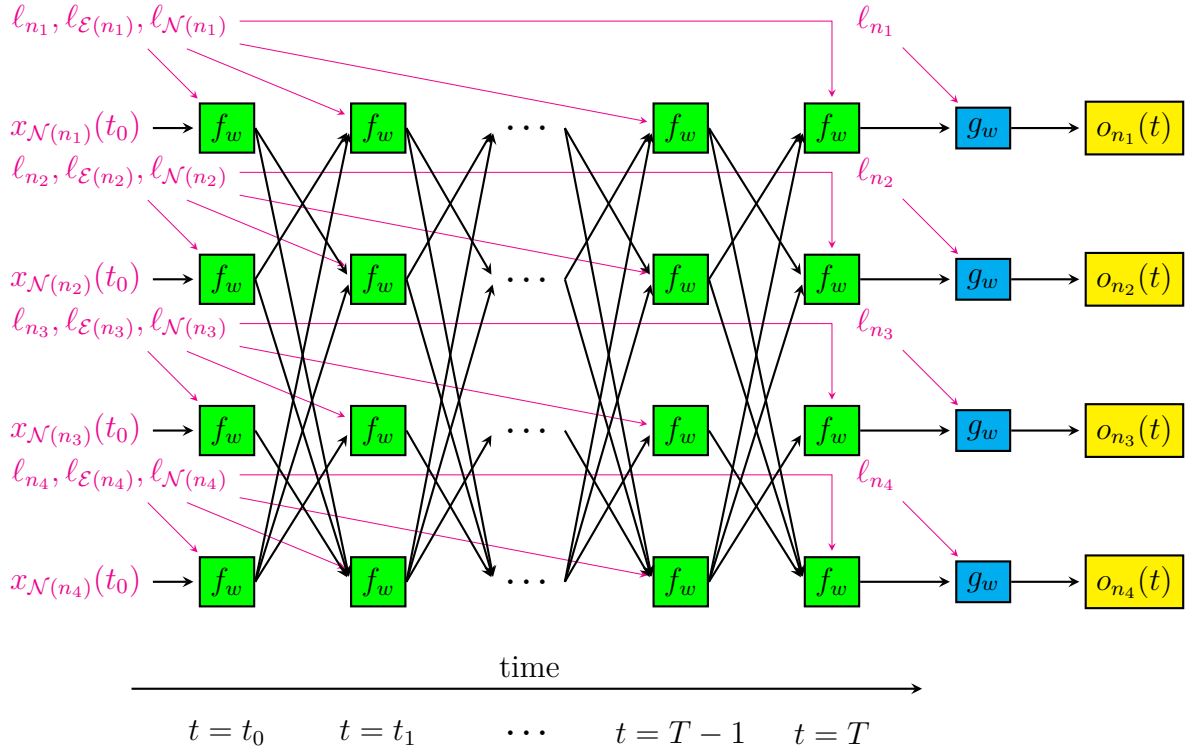




Now from this graph we reach to the following diagram which we call *encoding network*.



The encoding network is a recurrent neural network. And the network obtained by unfolding the encoding network looks like the following,



When  $f_w$  and  $g_w$  are implemented by feedforward neural networks, the encoding network

turns out to be a recurrent neural network where the connections between the neurons can be divided into internal and external connections. The internal connectivity is determined by the neural network architecture used to implement the unit. The external connectivity depends on the edges of the processed graph.

## 1.2.2 The Learning Algorithm

the parameters  $w$  are estimated using examples contained in the training data set.

### Learning set

Let us consider an arbitrary set of graphs  $\mathcal{G}$ .

$$\mathcal{G} = \bigcup_{\alpha \in \Lambda} G_{\alpha}, \quad \text{where } G_{\alpha} = (N_{\alpha}, E_{\alpha})$$

Let us also consider,

$$\mathcal{N} = \bigcup_{\alpha \in \Lambda} N_{\alpha}$$

#### Definition 3: Domain of a supervised learning framework

The domain is the set of pairs of a graph and a node, i.e.  $\mathcal{D} = \mathcal{G} \times \mathcal{N}$ .

#### Definition 4: Learning Set

We take a finite set of graphs from  $\mathcal{G}$  as,

$$\{G_1, G_2, \dots, G_i, \dots, G_{p-1}, G_p\} \subseteq \mathcal{G} \implies p \leq |\mathcal{G}|.$$

Let us take a subset of the node set of  $G_i = (N_i, E_i)$  as,

$$\{n_{i,1}, n_{i,2}, \dots, n_{i,j}, \dots, n_{i,q_i-1}, n_{i,q_i}\} \subseteq N_i \in \mathcal{N} \implies q_i \leq |N_i|.$$

We assume a supervised learning framework with the following set,

$$\mathcal{L} = \{(G_i, n_{i,j}, t_{i,j}) | G_i = (N_i, E_i) \in \mathcal{G}, n_{i,j} \in N_i, t_{i,j} \in \mathbb{R}^m, 1 \leq i \leq p, 1 \leq j \leq q_i\}$$

$\mathcal{L}$  is called the learning set.

We consider a desired target association  $t_{i,j} \in \mathbb{R}^m$  of the node  $n_{i,j}$ , which is a vector in some euclidean space of dimension  $m$ . Note that  $\mathcal{G}$  or  $\{G_1, G_2, \dots, G_i, \dots, G_{p-1}, G_p\}$  can also be considered as one single graph which may have multiple connected components and modify the definition of learning set accordingly.

Learning in GNNs consists of estimating the parameter  $w$  such that the following function,

$$\varphi_w : \mathcal{D} \rightarrow \mathbb{R}^m$$

approximates the data in the learning set.

The learning task can be posed as the minimization of a quadratic cost function,

$$e_w = \sum_{i=1}^p \sum_{j=1}^{q_i} (t_{i,j} - \varphi(G_i, n_{i,j}))^2.$$

As common in neural network applications, the cost function may include a penalty term to control other properties of the model. For example, the cost function may contain a smoothing factor to penalize any abrupt changes of the outputs and to improve the generalization performance.

## Learning algorithm

The learning algorithm is based on a gradient-descent strategy and is composed of the following steps.

1. The states  $x_n(t)$  are iteratively updated by

$$\begin{aligned} x_n(t) &= f_w(\ell_n, \ell_{\mathcal{E}(n)}, x_{\mathcal{N}(n)}(t-1), \ell_{\mathcal{N}(n)}) \\ o_n(t) &= g_w(x_n(t), \ell_n) \end{aligned}$$

until at time  $t = T$  they approach the fixed point solution of  $x(T) \approx x^*$ . the hypothesis that  $F_w$  is a contraction map ensures the convergence to the fixed point.

2. The gradient  $\frac{\delta e_w(T)}{\delta w}$  is computed. This can be carried out in a very efficient way by exploiting the diffusion process that takes place in GNNs. Interestingly, this diffusion process is very much related to the one which takes place in recurrent neural networks, for which the gradient computation is based on backpropagation-through-time algorithm. However, backpropagation through time requires to store the states of every instance of the units. When the graphs and  $T - t_0$  are large, the memory required may be considerable. On the other hand, in our case, a more efficient approach is possible, based on the Almeida-Pineda algorithm [C.1](#).
3. The weights  $w$  are updated according to the gradient. This is carried out within the traditional framework of gradient descent.

### 1.2.3 Transition and Output Function Implementations

# Contraction mapping theorem

## A.1 Contraction mapping

### Definition 5: Contraction map

Let  $(X, d)$  be a complete metric space. Then a map  $T : X \rightarrow X$  is called a contraction map on  $X$  if there exists  $q \in [0, 1)$  such that

$$d(T(x), T(x')) \leq q d(x, x')$$

for all  $x, x' \in X$ . Contraction map is also known as contractive map. If the above condition is instead satisfied for  $q \in [0, 1]$ , then the mapping is said to be a non-expansive map.

More generally, if  $(X, d_X)$  and  $(Y, d_Y)$  are two metric spaces, then  $T : X \rightarrow Y$  is a contractive mapping if there is a constant  $q \in [0, 1)$  such that

$$d_Y(T(x), T(x')) \leq q d_X(x, x')$$

for all  $x$  and  $x'$  in  $X$ .

### Definition 6: Lipschitz continuous map

Given two metric spaces  $(X, d_X)$  and  $(Y, d_Y)$ , a function  $f : X \rightarrow Y$  is called Lipschitz continuous if there exists a real constant  $K \geq 0$  such that, for all  $x$  and  $x'$  in  $X$ ,

$$d_Y(f(x), f(x')) \leq K d_X(x, x').$$

Any such  $K$  is referred to as a Lipschitz constant for the function  $f$  and  $f$  may also be referred to as  $K$ -Lipschitz.

- The smallest constant is sometimes called the (best) Lipschitz constant of  $f$  or the dilation or dilatation of  $f$ .
- If  $K = 1$  the function is called a short map.
- If  $0 \leq K < 1$ , the function is called a contraction [Definition 5].

### Lemma A.1

Contraction map  $\implies$  Lipschitz continuous map.

*Proof.* Clearly by definition. ■

### Definition 7: Uniformly continuous map

Let  $M_1 = (A_1, d_1)$  and  $M_2 = (A_2, d_2)$  be metric spaces. Then a mapping  $f : A_1 \rightarrow A_2$  is uniformly continuous on  $A_1$  if and only if:

$$\forall \epsilon \in \mathbb{R}_{>0} : \exists \delta \in \mathbb{R}_{>0} : \forall x, y \in A_1 : d_1(x, y) < \delta \implies d_2(f(x), f(y)) < \epsilon.$$

### Definition 8: Continuous map

Let  $M_1 = (A_1, d_1)$  and  $M_2 = (A_2, d_2)$  be metric spaces. Then a mapping  $f : A_1 \rightarrow A_2$  is continuous at a point  $c \in A_1$  if and only if:

$$\forall \epsilon \in \mathbb{R}_{>0} : \exists \delta \in \mathbb{R}_{>0} : d_1(c, y) < \delta \implies d_2(f(c), f(y)) < \epsilon.$$

### Lemma A.2

Uniformly continuous map  $\implies$  Continuous map.

*Proof.* Given  $\epsilon > 0$ . Then by uniform continuity, there is  $\delta$  such that for all  $x, y \in A_1$  we have

$$d_1(x, y) < \delta \implies d_2(f(x), f(y)) < \epsilon.$$

In particular if we take  $x = c$  we get that for all  $y \in A_1$

$$d_1(c, y) < \delta \implies d_2(f(c), f(y)) < \epsilon.$$

This is the definition of continuity [Definition 8]. ■

### Lemma A.3

Lipschitz continuous map  $\implies$  Uniformly continuous map.

*Proof.* Given two metric spaces  $(X, d_X)$  and  $(Y, d_Y)$ , a Lipschitz continuous map  $f : X \rightarrow Y$ , we get from Definition 6, there exists a real constant  $K \geq 0$  such that, for all  $x$  and  $x'$  in  $X$ ,

$$d_Y(f(x), f(x')) \leq K d_X(x, x').$$

- Case 1 ( $K = 0$ ): we get for all  $x$  and  $x'$  in  $X$ ,

$$d_Y(f(x), f(x')) = 0.$$

Hence,  $f$  is a constant map. Clearly  $f$  is uniformly continuous [use Definition 6].

- Case 2 ( $K > 0$ ): By Lipschitz continuity,

$$\forall x, x' \in X : d_Y(f(x), f(x')) \leq K d_X(x, x').$$

Take  $\epsilon > 0$ . Then there exists  $\delta = \frac{\epsilon}{K}$ . Suppose,  $d_X(x, x') < \delta$ . Hence, we have,

$$\begin{aligned} \forall \epsilon \in \mathbb{R}_{>0} : \exists \delta = \frac{\epsilon}{K} \in \mathbb{R}_{>0} : \forall x, x' \in X : d_X(x, x') < \delta \\ \implies K d_X(x, x') < \epsilon \\ \implies d_Y(f(x), f(x')) < \epsilon. \end{aligned}$$

■

#### Lemma A.4

Contraction map  $\implies$  Lipschitz continuous map  $\implies$  Uniformly continuous map  
 $\implies$  Continuous map.

*Proof.* Using Lemma A.1, A.3 and A.2.

■

#### Definition 9: Fixed point of a function

$c$  is a fixed point of a function  $f$  if  $c$  belongs to both the domain and the codomain of  $f$ , and  $f(c) = c$ .

#### Definition 10: Cauchy sequence

A sequence  $a_1, a_2, \dots$  is called a Cauchy sequence if the terms of the sequence eventually all become arbitrarily close to one another. That is,

$$\forall \epsilon > 0, \exists N \in \mathbb{N} \ni m, n > N \implies |a_m - a_n| < \epsilon.$$

#### Definition 11: Complete metric space

A metric space  $(X, d)$  is complete if every Cauchy sequence in  $X$  converges in  $X$  (that is, to some point of  $X$ ).

## A.2 Contraction mapping theorem

#### Theorem A.1: Contraction mapping theorem

Let  $(X, d)$  be a non-empty complete metric space with a contraction mapping  $T : X \rightarrow X$ . Then  $T$  admits a unique fixed-point  $x^* \in X$  (i.e.  $T(x^*) = x^*$ ). Furthermore,  $x^*$  can be found by starting with an arbitrary element  $x_0 \in X$  and defining a sequence  $(x_n)_{n \in \mathbb{N}}$  by  $x_n = T(x_{n-1})$  for  $n \geq 1$ . Then  $\lim_{n \rightarrow \infty} x_n = x^*$ .

**Proof. Existence of fixed point**

Let  $x_0 \in X$  be arbitrary and define a sequence  $(x_n)_{n \in \mathbb{N}}$  by setting  $x_n = T(x_{n-1})$ . We first note that for all  $n \in \mathbb{N}$ , we have the inequality

$$d(x_{n+1}, x_n) \leq q^n d(x_1, x_0).$$

This follows by induction on  $n$ , using the fact that  $T$  is a contraction mapping. More explicitly, note that as  $T$  is a contraction mapping and using the definition  $x_n = T(x_{n-1})$ , we claim that there exists  $q \in [0, 1)$  such that,

$$\begin{aligned} d(x_2, x_1) &= d(T(x_1), T(x_0)) \leq q d(x_1, x_0) \\ d(x_3, x_2) &= d(T(x_2), T(x_1)) \leq q d(x_2, x_1) \leq q^2 d(x_1, x_0) \\ &\vdots \\ d(x_{n+1}, x_n) &\leq q^n d(x_1, x_0) \end{aligned}$$

This follows by induction on  $n$ , using the fact that  $T$  is a contraction mapping. Then we can show that  $(x_n)_{n \in \mathbb{N}}$  is a Cauchy sequence. In particular, let  $m, n \in \mathbb{N}$  such that  $m > n$ :

$$\begin{aligned} d(x_m, x_n) &\leq d(x_m, x_{m-1}) + d(x_{m-1}, x_{m-2}) + \cdots + d(x_{n+1}, x_n) \\ &\leq q^{m-1} d(x_1, x_0) + q^{m-2} d(x_1, x_0) + \cdots + q^n d(x_1, x_0) \\ &= q^n d(x_1, x_0) \sum_{k=0}^{m-n-1} q^k \\ &< q^n d(x_1, x_0) \sum_{k=0}^{\infty} q^k \\ &= q^n d(x_1, x_0) \left( \frac{1}{1-q} \right). \end{aligned}$$

Let  $\epsilon > 0$  be arbitrary. Since  $q \in [0, 1)$ , we can find a large  $N \in \mathbb{N}$  so that

$$q^N < \frac{\epsilon(1-q)}{d(x_1, x_0)}.$$

Therefore, by choosing  $m$  and  $n$  greater than  $N$  we may write:

$$d(x_m, x_n) \leq q^n d(x_1, x_0) \left( \frac{1}{1-q} \right) < \left( \frac{\epsilon(1-q)}{d(x_1, x_0)} \right) d(x_1, x_0) \left( \frac{1}{1-q} \right) = \epsilon.$$

This proves that the sequence  $(x_n)_{n \in \mathbb{N}}$  is Cauchy. By completeness of  $(X, d)$ , the sequence has a limit  $x^* \in X$ . Furthermore,  $x^*$  must be a fixed point of  $T$ :

$$x^* = \lim_{n \rightarrow \infty} x_n = \lim_{n \rightarrow \infty} T(x_{n-1}) = T\left(\lim_{n \rightarrow \infty} x_{n-1}\right) = T(x^*).$$

As a contraction mapping,  $T$  is continuous, so bringing the limit inside  $T$  was justified.

**Uniqueness of fixed point**

Let  $x, x' \in X$  are both fixed points where  $x \neq x'$ . Thus  $d(x, x') \neq 0$ . But notice,

$$\begin{aligned} d(x, x') &= d(T(x), T(x')) \leq q d(x, x') \\ \implies (1-q)d(x, x') &\leq 0 \\ \implies d(x, x') &\leq 0 \implies d(x, x') = 0 \iff x = x'. \end{aligned}$$

Hence, it is not possible to have two different fixed points.

### Calculating the fixed point

We notice that no matter where we choose our arbitrary starting point  $x_0 \in X$ , by the construction of the sequence using the contraction mapping (i.e.  $x_n = T(x_{n-1})$ ) we always reach to the unique fixed point as the limit of the sequence,

$$\lim_{n \rightarrow \infty} x_n = x^*.$$

### Speed of convergence:

For the fixed point  $x^*$  the speed of convergence is described by the following,

$$d(x^*, x_n) \leq \frac{q^n}{1 - q} d(x_1, x_0)$$

We can see that for any initial value  $x_0$  the system converges to the fixed point  $x^*$  exponentially. ■



# Classic iterative methods

## B.1 Jacobi and Gauss-Seidel iterative method

The current analysis is based on [3]. In this notes we discuss classic iterative methods on solving the linear operator equation,

$$Au = f$$

posed on a finite dimensional Hilbert space  $\mathbb{V} \cong \mathbb{R}^N$  equipped with an inner product  $(\mathfrak{u}, \mathfrak{u})$ . Here  $A : \mathbb{V} \mapsto \mathbb{V}$  is an *symmetric and positive definite* (SPD) operator,  $f \in \mathbb{V}$  is given, and we are looking for  $u \in \mathbb{V}$  such that  $Au = f$  holds.

### Definition 12

the matrix  $A$  is diagonally dominant if

$$|a_{ii}| \geq \sum_{j \neq i} |a_{ij}| \quad \text{for all } i.$$

where  $a_{ij}$  denotes the entry in the  $i$ th row and  $j$ -th column. Note that this definition uses a weak inequality, and is therefore sometimes called weak diagonal dominance. If a strict inequality ( $>$ ) is used, this is called strict diagonal dominance.

### Theorem B.1

A Hermitian diagonally dominant matrix  $A$  with real non-negative diagonal entries is positive semidefinite.

---

# Important Algorithms

## C.1 Almeida-Pineda algorithm

---

# Index

cauchy sequence, [12](#)  
complete metric space, [12](#)  
continuous map, [11](#)  
contraction map, [10](#)  
contraction mapping theorem, [10](#)  
  
dilatation, [10](#)  
dilation, [10](#)  
domain, [8](#)  
  
encoding network, [7](#)  
  
fixed point, [12](#)  
  
label, [2](#)  
learning Set, [8](#)  
Lipschitz constant, [10](#)  
Lipschitz continuous, [10](#)  
  
minimal model, [3](#)  
  
non-expansive map, [10](#)  
nonpositional form, [4](#)  
nonpositional graph, [4](#)  
  
output fuction, [2](#), [5](#)  
  
positional form, [4](#)  
positional graph, [4](#)  
  
short map, [10](#)  
  
transition fuction, [2](#), [5](#)  
  
uniformly continuous, [11](#)

---

# Bibliography

- [1] Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. “The Graph Neural Network Model”. In: *IEEE Transactions on Neural Networks* 20.1 (2009), pp. 61–80. DOI: [10.1109/TNN.2008.2005605](https://doi.org/10.1109/TNN.2008.2005605) (page - 1).
- [2] Michael JD Powell. “An efficient method for finding the minimum of a function of several variables without calculating derivatives”. In: *The computer journal* 7.2 (1964), pp. 155–162 (page - 6).
- [3] Roberto Bagnara. “A unified proof for the convergence of Jacobi and Gauss–Seidel methods”. In: *SIAM review* 37.1 (1995), pp. 93–97 (page - 15).