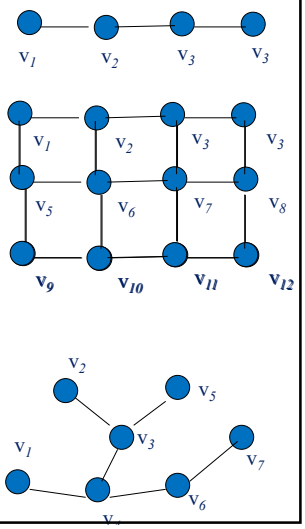
Graph neural networks

240

Graph Neural Networks (GNNs)

The main idea

- graphs are an extension of sequences
  - Recurrent networks operate on sequences of data
- graphs are irregular grids
  - Convolutional networks operate on grids of data
- GNNs a class neural network models that extend both recurrent networks and convolutional networks and operates on graphs



241

Graphs: they are a general tool to represent data

In any software

- Graphs can represent entities, their relationships and their information

In machine learning, two types of domains

- Graph focused: each graph represents a complex pattern made by parts with their interecation
  - The goal is usually that of predicting some property of the whole graph
  - Links represent friendships (may have attached information)
- Node/edge focused: each graph represents a set of patterns with their relationships
  - The goal is usually that predicting some property of a node/edge

242

Graphs: they are a general tool to represent data

- Social networks
  - Nodes represent users (may have attached information)
  - Links represent friendships (may have attached information)
  - Predicting friendships is an edge focused application
- Protein networks
  - Node stand for proteins (may have attached information)
  - Edges denote their possible interaction (may have attached information)
  - Predicting protein property is anode focused application
- Molecules
  - Nodes stand for atoms (may have attached information)
  - Edges stand for physically close atoms or those atoms having an atomic interaction (may have attached information)
  - Predicting molecules property is graph focused application

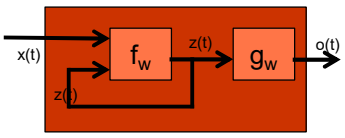
243

Graph Neural Networks (GNNs)

- First models for graph processing introduced in 90es
  - Mainly for graph focused applications
- GNN is a class of models for graph processing
- In the following, first I introduce the original model and then I explain how this has been modified

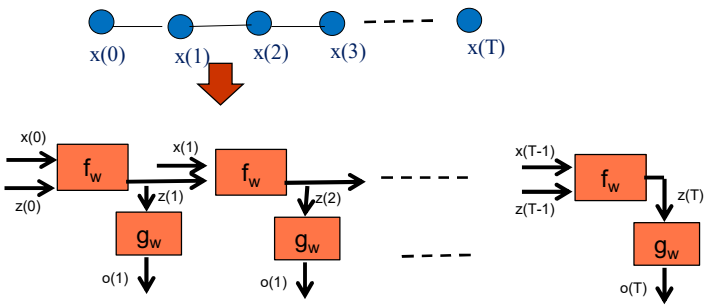
244

GNNs can be defined by extending recurrent networks



Remember the unfolding network

- I recurrent networks, we have a transaction network f_w and an output network g_w
- A copy of f_w, g_w for each time instance, connected in sequence

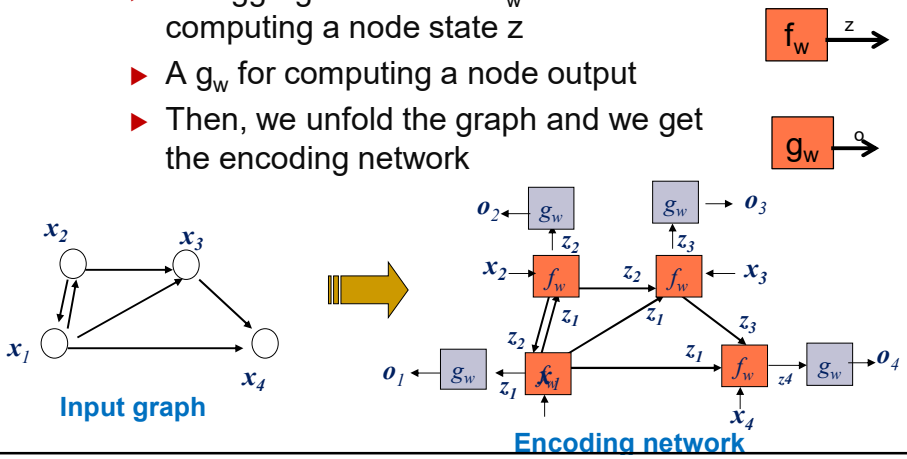


245

GNNs can be defined by extending recurrent networks

We still have two neural networks

- An aggregation funtion f_w for computing a node state z
- A g_w for computing a node output
- Then, we unfold the graph and we get the encoding network

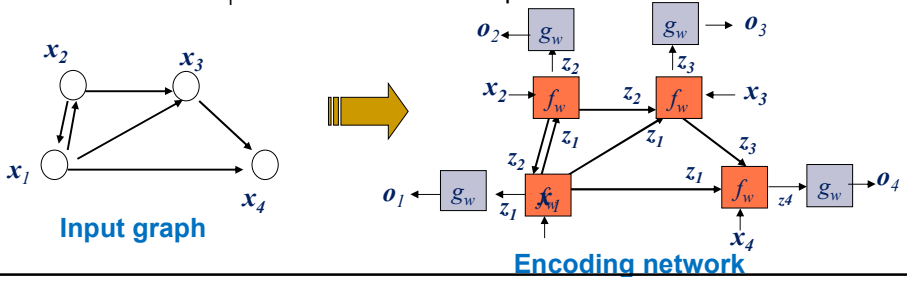


246

GNNs can be defined by extending recurrent networks

For a general graph, we can construct the encoding network

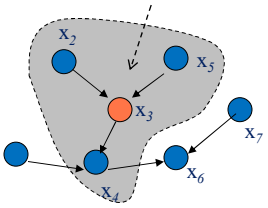
- The x_i are the feature vectors attached to nodes (the node information)
- The edges represent the relationship between patterns
- The o_i are the network output
- The z_i are an internal state representation for the node



247

GNNs: computing the states

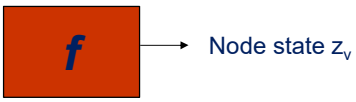
The neighborhood of this node



- The state z_n of a node are computed by combining the information on node neigborhood
- In the original model, we used

$$z_v = \sum_{n \to v} h_w(x_n, z_n)$$

where h_w is a three layer neural network



- but, several other networks are now used ..

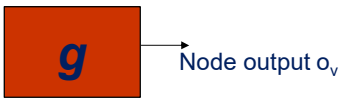
248

GNNs: computing the outptut

- The state o_n of a node are computed from the states z_n
- In the original model, we used

$$o_v = g_w(x_v, z_v)$$

where g_w is a three layer neural network



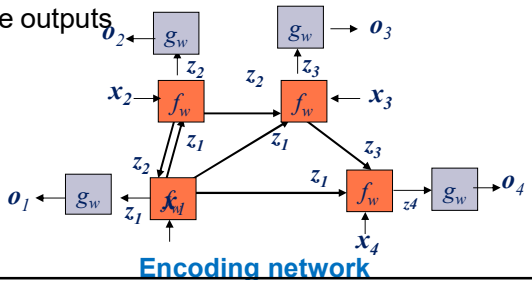
- but, several other networks are now used and sometime is just not used…. back on this later

249

Computing GNN outputs

The enconding network is cyclic … how are outputs computed?

1. Set initial states z(t)=0
2. **Repeat**
3. activate units *f* to compute new states z(t+1)
4. **Until** z(t) do not change any more
5. activate units g to compute outputs



Encoding network

Computing GNN outputs

The enconding network is cyclic …

- Does the forward phase always converge?
- Does the forward phase always converge to the same state despite the initial state?

Yes

- The original GNNs adopt a mechanism based on contraction systems

A mathematical point of view

Consider the whole encoding network as adynamic system/system of equation does the system always converge/ does the system has a unique solution?

$$Z(t) = F(X, Z(t))$$

- Yes, provided that it is a contraction, i.e, if the norm of the Jacobian is smaller than
$$\left\| \frac{\partial F(l, Z)}{\partial Z} \right\| < 1$$
- The original GNNs adopt a mechanism which

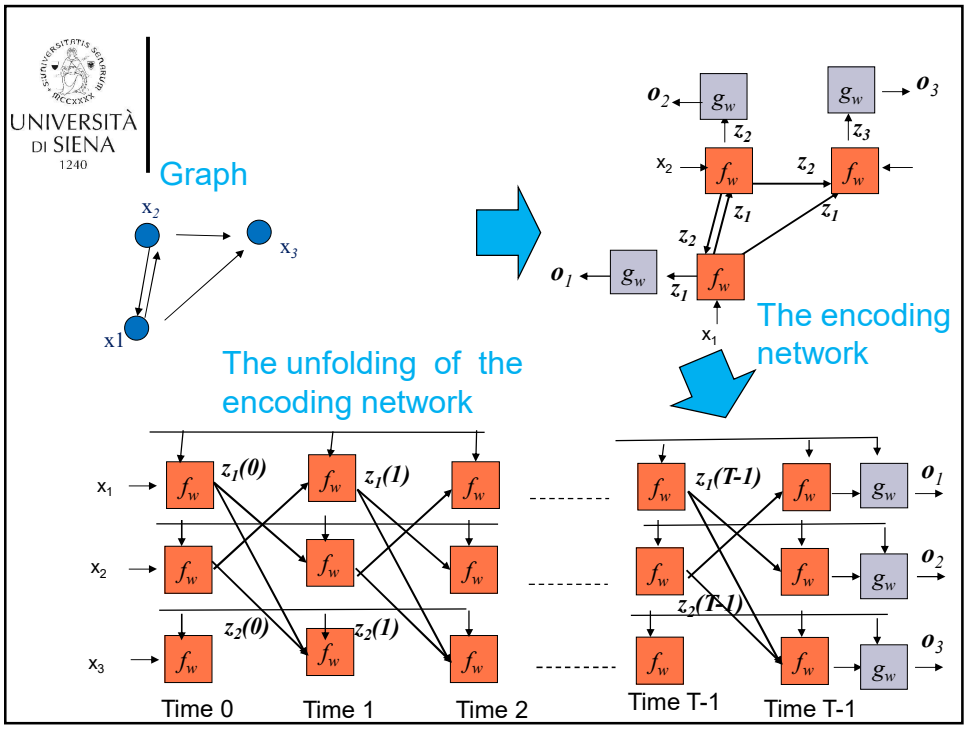
252

Computing GNN gradient

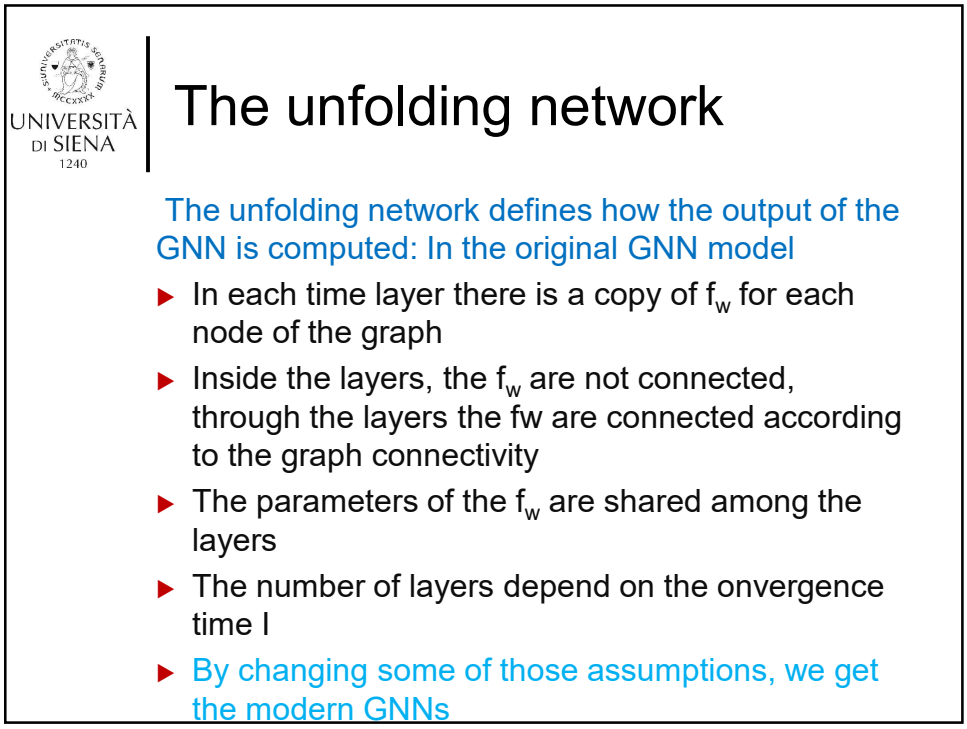
Based on unfolding of encoding network

- The encoding network is unfolded through time
  - The result is a feedforward network equivalent to the encoding network
- The gradient is computed on the unfolding using backpropagation through time algorithm

253



254



The unfolding network

The unfolding network defines how the output of the GNN is computed: In the original GNN model

- In each time layer there is a copy of f_w for each node of the graph
- Inside the layers, the f_w are not connected, through the layers the fw are connected according to the graph connectivity
- The parameters of the f_w are shared among the layers
- The number of layers depend on the onvergence time l
- By changing some of those assumptions, we get the modern GNNs

255

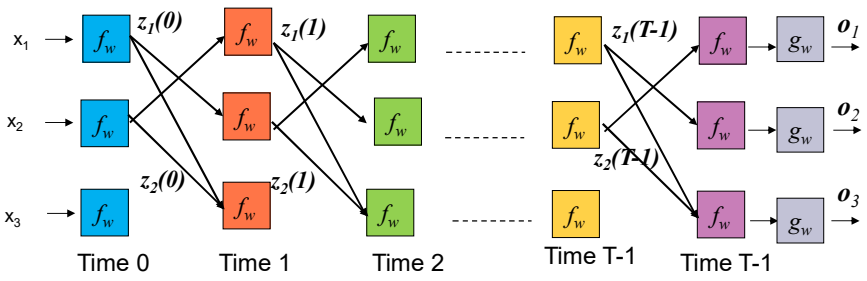
Modern GNNs

- Almost all of them
  - Fix the number of layers
  - Use different parameter set on each layer
  - Adopt novel aggregation function w.r.t. the orginal model
- Two big classes
  - Graph Convolutional Networks
  - Recursive GNNs

256

Graph Convolutional Networs (GCNs)

- Fix the number of layers (iterations)
- Use different parameter set on each layer
- Remove the direct input to the internal layers
- Now, the unfolding network looks like a deep network



257

An example: the first CGN by Kipf and Welling

- z is initialized with the input features
- The weight matrix W_k is normalized by column and rows
- The aggregation function is

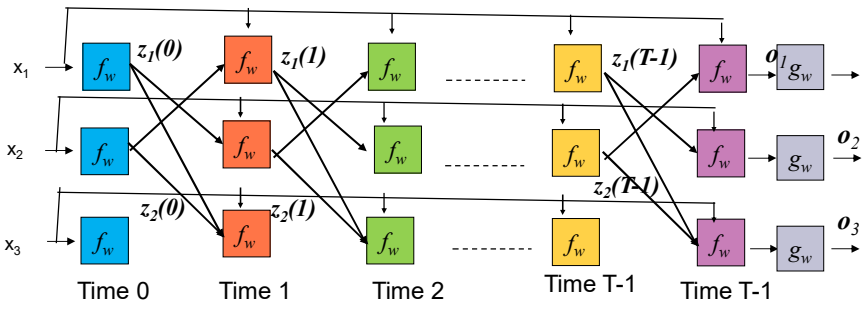
$$z_v(0) = l_v$$

$$z_v(k) = \sum_{n \to v} relu\left(\frac{z_n(k-1)\, Wk}{|ne[v]||ne[n]|}\right)$$

258

Recursive Graph Neural Networks (RGNNs)

- Fix the number of layers (iterations)
- Use different parameter set on each layer
- Keep the direct input to the internal layers
- Good architecture for dynamic GNNs!



259

An example: Gated Graph Neural Networks

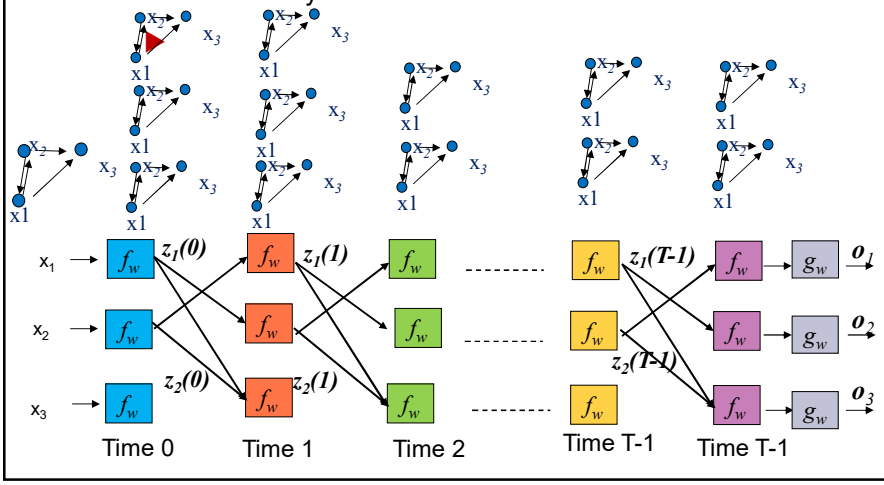
- A GRU unit is used to combine the features over the layers

$$z_v(k) = GRU\left(z_v(k-1), \sum_{n \to v} z_n(k-1),\right)$$

260

GNNs: another observation

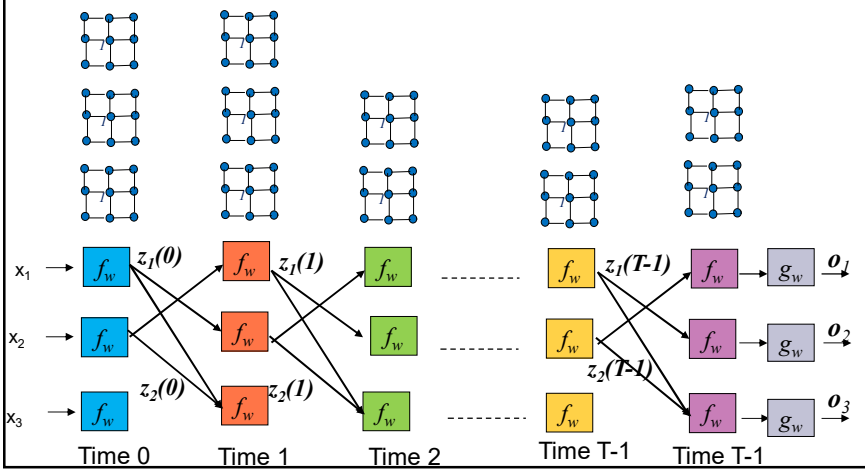
- You may assume that each feature corresponds to a graph in a layer
- Nodes between layers are connected according to graph connectivity



261

GNNs: another observation

The unfolding of Graph Convolutional networks resembles common convolution networks, with the difference that CNs operates on grids



262

GCNs are sort of convolutional neural networks

- In convolutional networks, a kernel is applied to each point n of a receptive field of an image
- The oputput of the kernel is

$$z_v(k) = \sum_{n\ in\ recfield[v]} relu(z_{n(k-1)} W_{k,\ n-v})$$

- In a CGN or a GNN, an aggregation function is used to combine the contributions of the neigbours, e.g.

$$z_v(k) = \sum_{n \to v} relu(\frac{z_n(k-1)\ Wk}{|ne[v]||ne[n]|})$$

263

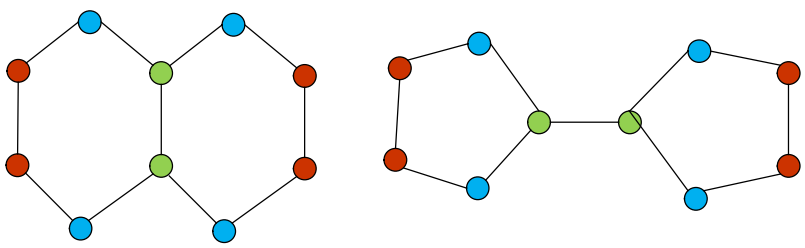
About expressive power of GNNs

- Which applications on graph can GNNs implement?
- Two different questions to answer
  - Which graphs/nodes a GNN can distinguish?
  - Which functions a GNN can approximate?
- Intuitive preliminar response: GNNs can distinguish all the graphs and are universal approximators except for
  - the limits due to local computation framework
  - provided a sufficiently complex aggregation function is used

264

Local computation limit: intuitive version

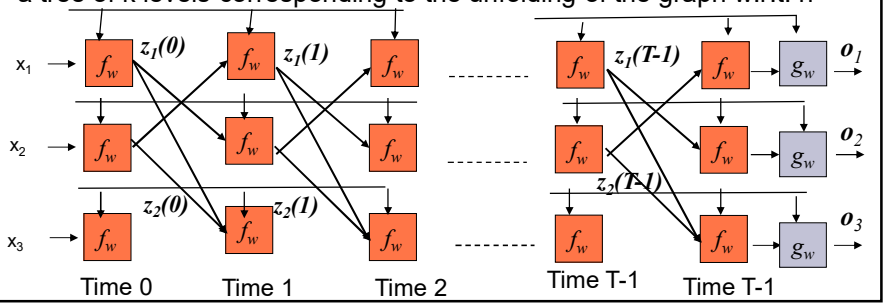
- A GNN process the graph locally:
  - if two nodes have the same neighboorhood, then they look equal to the GNN
  - If all the nodes of two graphs have the same neighborhood, the two graphs look equal to the GNN



265

Local computation limit: unfolding trees

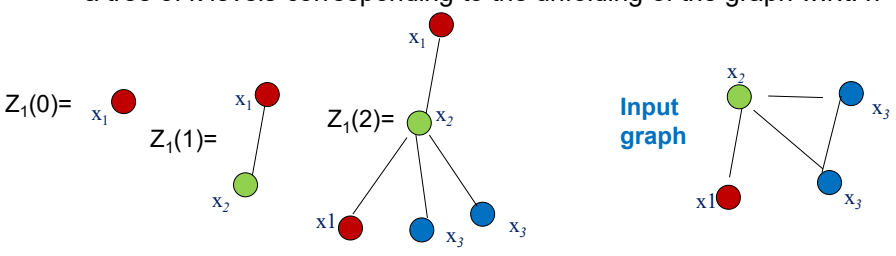
- The state $z_n(0)$ can contain only info about the features of node n
- The state $z_n(1)$ can contain only info about node n and of its neighboors
- The state $z_n(2)$ can contain only info about node n, about its neigboors and its neighboors of neighboors
- ....
- The best you can do, it is to store store in $z_n(k)$ the unfolding tree:
  a tree of k levels corresponding to the unfolding of the graph w.r.t. n



266

Unfolding tree

- The state $z_n(0)$ can contain only info about the features of node n
- The state $z_n(1)$ can contain only info about node n and of its neighboors
- The state $z_n(2)$ can contain only info about node n, about its neigboors and its neighboors of neighboors
- ....
- The best you can do, it is to store store in $z_n(k)$ the unfolding tree:
  a tree of k levels corresponding to the unfolding of the graph w.r.t. n



267

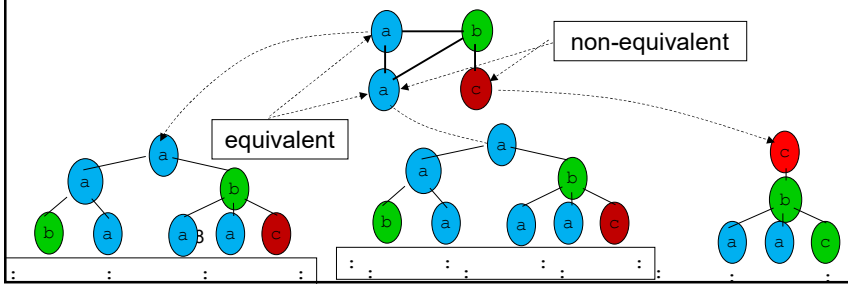
The unfolding equivalence

Unfolding tree

- The unfolding tree T_v^d is the tree obtained by unfolding the graph starting in v up to d levels

Unfolding equivalence

- two nodes are unfolding equivalent $n \sim v$ if $T_n^d = T_v^d$ holds for every d



268

Limit of local computation framework

Formally,

- GNNs cannot distinguish unfolding equivalent nodes

Intuitively

- Two copies of the same node or two different nodes with the same features?
  - The unfolding trees may contain copies of the same nodes due to cycles and the bidirectional links
  - The GNN is not able to distinguish between two copies of the same node from two different nodes with the same features

269

Limit of local computation: Weisfeler-Lehman test

Weisfeler-Lehman test (1-WL)

- An algorithm to test whether two graphs are isomorphic or not
- Based on a signature obtained by assigning colors to each node
- Colors are defined iteratively on the base of a local mehanishm

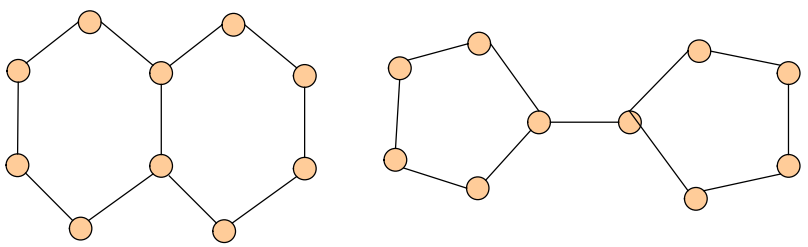
$$c_v(0) = HASH(l_v)$$
$$c_v(k) = HASH(cv(k-1), \{cn(k-1) | n \in ne[v]\})$$

- The algorithm stops when colors do not changes
- Notice: 1-WL may fail to recognize that two graphs are not isomorphic

270

Weisfeler-Lehman test

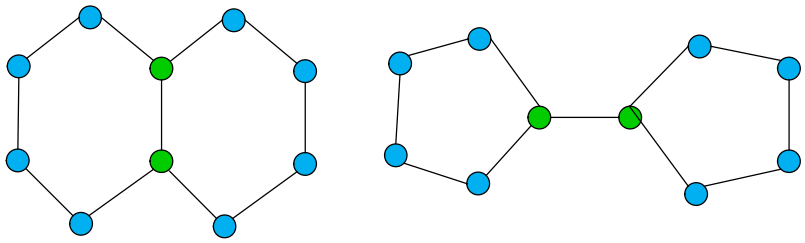
- Step 0 ….
  - In those graphs we assume that all nodes have the same inpute features $l_1=l_2$….



271

Weisfeler-Lehman test

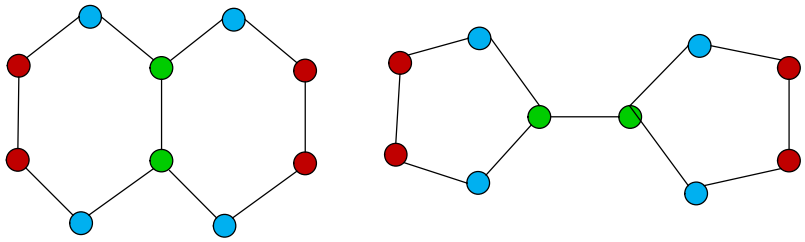
- Step 1 ….



272

Weisfeler-Lehman test

- Step 2 ….



273

Limit of local computation: Weisfeler-Lehman test

I has been proved that

- GNNs cannot be better than 1-WL
  - They cannot distinguish graphs that cannot be distingued by 1-WL

I has been proved that

- 1-WL equivalence and unfolding equivalence are equal

274

Capabilities of GNN

- Can a GNN be as powerful as 1-WL?
- Can a GNN be able to store the unfolding tree into node featues
  - The answer is positive to both questions provided aggregation function is sufficiently general to simulate 1-WL / unfolding tree construction

We will consider GNNs in the form of

$$z_v(k) = \mathrm{comb}_k\left(z_v(k-1), \mathrm{agg}_k(\{z_n(k-1):\ v \to n\})\right)$$

readout(z_{v1},….,)

which includes most of modern Convolutional GCNs

275

GNNs are as powerful as 1-WL

Xu et al.

$$z_v(k) = \text{comb}_k\big(z_v(k-1), \text{agg}_k(\{z_n(k-1):\ v \to n\})\big)$$

readout(z_{v1},….,)

- If $comb, agg$ aggregates recursively the children features
- if $comb, agg,$ and $readout$ are injective
- then the GNN produces an embedding of the graphs
  - that is equivalent to the embedding produced by Weisfeiler-Lehman test in order to recognize whether the graphs are isomorphic or not

276

GNNs can store the unfolding trees

$$z_v(k) = \text{comb}_k\big(z_v(k-1), \text{agg}_k(\{z_n(k-1):\ v \to n\})\big)$$

- If $comb, agg$ construct recursively the unfolding trees
- if $comb, agg,$ are injective (each tree is mapped to a different coding)

- It is a consequence of the equivalence between unfolding equivalence and 1-WL
- But there are also constructive proofs

277

About GNN approximation capability

- Which funtions on graph can GNNs approximate?
- Let us consider a function φ that takes in input a graph G and return a real output φ(G,n) for some node n
- Answer 1.0

A GNN model, can approximate any function modulo the limits due to local computation mechanism

278

About GNN approximation capability

Intuitive idea of the proof

- We adopt the same reasoning we used for recurrent networks
  - If the comb, agg function can store into features a coding of the graph (the unfolding tree)
  - then readout can decode the emcoding and produce any desired ouput

$$z_v(k) = \mathrm{comb}_k\big(z_v(k-1), \mathrm{agg}_k(\{z_n(k-1):\ v \to n\})\big)$$

φ(G,n) =readout(z_{v1},….,)

279

Functions that preserve the unfolding equivalence

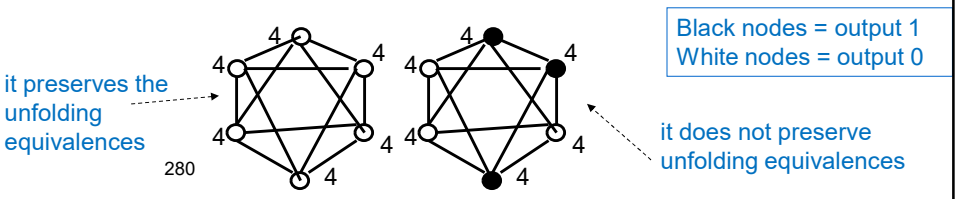
Functions that preserve the unfolding equivalence/ 1-WL

- A function $\tau : \mathcal{G} \times \mathcal{N} \rightarrow R^m$ preserves the unfolding equivalence/ 1-WL if

  n~v implies $\tau(G,n)=\tau(G,v)$

Example

- a function that preserves the unfolding equivalence has the some output on a uniform graph where all the node has the same labels



280

280

About GNN approximation capability

- GNNs can approximate any function that preserves the unfolding equivalence/1-WL
  - The result holds for original GNN model and modern GNNs
  - The result holds in probability for measurable functions (D'Inverno & Alt.) and also for any continuous functions (Azizian & Alt.)

281

281

Which aggregation functions?

Advanced question

- We would like to get some details about the aggregation and the output function we should use

Recall main idea

$$z_v(k) = \text{comb}_k\big(z_v(k-1), \text{agg}_k(\{z_n(k-1): v \to n\})\big)$$

φ(G,n) =readout(z_{v1},….,)

- $comb, agg$ construct recursively the unfolding trees
- $comb, agg,$ are injective (each tree is mapped to a different coding)
- *readout* decodes representation and produce output

282

Which aggregation function?

- $comb, agg$ construct recursively the unfolding trees
- *readout* decodes representation and produce output
- then, the comb+aggregation and read function should have at least two layers (one hidden layer)
  - but most modern GNNs exploit just a single layer, e.g. in CGN

$$z_v(k) = \sum_{n \to v} relu\left(\frac{z_n(k-1)\, Wk}{|ne[v]||ne[n]|}\right)$$

  - The advantage is in the fact that using a single layer makes the architecture simpler, faster, and improves generalization
  - A big advantage for applications with a lot of features

283

Which aggregation function?

$$z_v(k) = \text{comb}_k\big(z_v(k-1), \text{agg}_k(\{z_n(k-1): v \to n\})\big)$$

$\varphi(G,n) = \text{readout}(z_{v1}, \ldots,)$

$comb, agg$ and readout are injective

- But some modern GNNs exploit AVERAGE, MAX. In this case, $xomb, agg\ readout$ are not injective.

284

Which aggregation function?

An other case

- the average (normalization) may be useful to avoid that the gradient become smaller during back propagation
- but with average, you cannot count, e.g. the number of children of a nodes
  - If your target application is to recognize the users that have a lot of friends in a a social network, using AVERAGE in your aggragation function is nota good idea!

285

Back to what GNNs cannot approximate

- GNNs cannot do better than 1-WL, namely they must preseserve unfolding equivalence
- Notice that such a behaviour may not be a limitation
  - two concepts having the same labels and the same relationships with the other concepts provide the same information:
  - why those concepts should be distinguished
- with an appropriate modification of the graphs, a functions may preserve the unfolding equivalence
  - f.i. an extreme example all the labels of the input graph are distinct labels

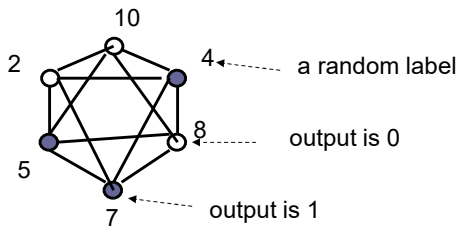
286

286

A funny experiment about GNN approximation properties

A GNN produces the same result on a uniform graphs but …

- if a noise is introduced into the labels, the nodes become distinguishable and, in theory, GNNs can implement any function
- is it possible to learn a GNN that return -1 for a half of the nodes (white nodes) and 1 for the other half (black nodes)?



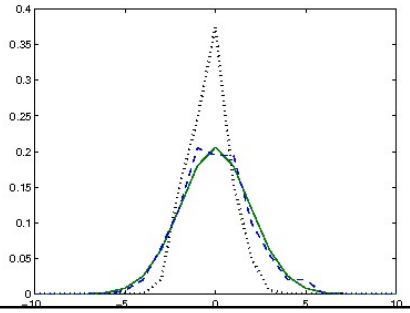
287

287

A funny experiment on GNN approximation properties

300 random uniform graphs

- random labels
- the error was measured by the difference between the desired number of black nodes on those obtained by the GNN
- results were compared with those obtained by random process and a feedforward neural network



288

288

Beyond 1-WL: Weisfeiler-Lehman hierachy

- Weisfeiler-Lehman hierarchy
  - 1-WL exploits single nodes
  - k-WL exploits k-tuples of nodes
  - k-WL is more powerful than (k-1)-WL

289

289

k-WL

- k-tuple of nodes T=(v_1,..v_k)
- j-th neighbourhood of the tuple: replace the j-th node with any other node in graph
  - $N_j(T)=\{(v_1,..,v_{j-1},n,v_{j+1...}.v_k): n \in N\}$

The algorithm

- Inizialize each tuple color, using node features and local connectivity
- Iterate
  - Compute neighborhoods for each j
  - Calculate using new colors using colors of previous neighboroods

$$c_T(0) = \text{HASH}(G, l_v)$$
$$c_T^j(k) = \{c_H(k-1) | H \in N_j(V)\} \text{ for each j}$$
$$c_T(k) = \text{HASH}(cT(k-1), \{C_H{}^1, \ldots C_H{}^k\})$$

290

Going beyond GNNs

k-tuple of nodes T=(v_1,..v_k)

The general idea

- replace nodes with tuples of nodes
- use the tuple neighbourhood defined so as in the above slide

$$z_T(k) = \text{comb}_k\left(z_T(k-1), \text{agg}_k(\{z_H(k-1): H \in ne[T]\})\right)$$

readout(z_{T1},….,)

291

Going beyond GNNs

An example (Morris &Alt)

k-tuple of nodes $T=(v_1,..v_k)$

$$z_T(k) = \text{relu}\left(z_T(k-1)W_1{}^k + \sum_{H \in ne[T]} z_H(k-1)W_2{}^k\right)$$

- ne[T] may consider
  - local neighbourhood for decreasing complexity (tuples which differ for two nodes linked by edges)
  - global neighbourhood

292

Going beyond GNNs

An example (Morris &Alt)

k-tuple of nodes $T=(v_1,..v_k)$

$$z_T(k) = \text{relu}\left(z_T(k-1)W_1{}^k + \sum_{H \in ne[T]} z_H(k-1)W_2{}^k\right)$$

It has been proved that

- Such a model can distinguish graphs so as k-WL
- More powerful than standard GNN

Obviously, it is not a local model

- more powerful, may be less generalizzation

293

Generalization in GNNs: VCD on graphs

- shattering on graphs
  - A set of graphs with their nodes $(G_1,v_1)\ldots. (G_k,v_k)$
  - An assignament ((G_1,v_1) +1) …. $((G_k,v_k)$, -1)
- A GNN shatters a set $(G_1,v_1)\ldots. (G_k,v_k)$, if it can produce any assignament
- VCD is maximal dimension of a set shattered by a GNN

294

Generalization in GNNs

- From a theoretical view point, there are few results generalization in GNNs
  - One for the old GNN model (Scarselli, Tsoi, et al)
    - Based on Vapnik-Chernovenkis
    - The bounds are similar to the of other networks
    - Surprisingly, the length of unfolding in time does not play any role: this may be due to converge of the GNN model

295

Generalization in GNNs

- From a theoretical view point, there are few results generalization in GNNs
  - One for modern GNNs (Garg et al.)
    - Based on Rademacher complexity (not studied here)
    - The bound depend on the lenth of unfolding
    - Similar to that or recurrent networks

296

Bound on VCD dimension of the orginal GNNs

p=number of paramters
N=number of graph nodes

Activation	Bound
Graphs (GNNs and non–positional RecNNs)	
Polynomial	O(p log(N))
Tanh,logsig, atan	$O(p^4N^2)$
Sequences (recurrent networks)	
Polynomial	O(p log(N))
Tanh,logsig, atan	$O(p^4N^2)$
Vectors (multialyer neural networks)	
Polynomial	O(p log(p))
Tanh, logsig, atan	$O(p^4)$

297

297

Bound on Rademacher complexity of some modern GNNs

d=brancing factor
r=state dimension
L=depth
m=sample size
γ= margin used in the margine loss

C = a sort of Lipschitz constant for aggregation

	GNNs	RNNs
C<1/d	$O\left(\frac{rd}{\sqrt{m}\gamma}\right)$	$O\left(\frac{r}{\sqrt{m}\gamma}\right)$
C=1/d	$O\left(\frac{rdL}{\sqrt{m}\gamma}\right)$	$O\left(\frac{rL}{\sqrt{m}\gamma}\right)$
C>1/d	$O\left(\frac{rd\sqrt{rL}}{\sqrt{m}\gamma}\right)$	$O\left(\frac{r\sqrt{rL}}{\sqrt{m}\gamma}\right)$

298

298

Generalization in GNNs in practice

- Depending on the application, the generalization in GNNs is difficult to control
  - Test distribution must be same as training distribution both for node features and node connectivity
  - For complex applications, a lot of data may be required

299

UNIVERSITÀ DI SIENA 1240

Bibliography

- Auer, P., Herbster, M., & Warmuth, M. K. (1996). Exponentially many local minima for single neurons. In *Advances in neural information processing systems* (pp. 316-322).
- W. Azizian,M. Lelarge. Expressive power of invariant and equivariant graph neural networks. In Int. Conf. on Learning Representations (ICLR), 2021
- Baldi, P., & Hornik, K. (1989). Neural networks and principal component analysis: Learning from examples without local minima. *Neural networks*, *2*(1), 53-58.
- Bartlett, P. L. (1998). The sample complexity of pattern classification with neural networks: the size of the weights is more important than the size of the network. *IEEE transactions on Information Theory*, *44*(2), 525-536.

300

UNIVERSITÀ DI SIENA 1240

Bibliography

- Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157-166.
- Bianchini, M., Gori, M., & Maggini, M. (1994). On the problem of local minima in recurrent neural networks. *IEEE Transactions on Neural Networks*, *5*(2), 167-177.
- Bianchini, M., & Gori, M. (1996). Optimal learning in artificial neural networks: A review of theoretical results. *Neurocomputing*, *13*(2-4), 313-346.
- Bianchini, M., & Scarselli, F. (2014). On the complexity of neural network classifiers: A comparison between shallow and deep architectures. *IEEE transactions on neural networks and learning systems*, *25*(8), 1553-1565.
- Barron, A. R. (1993). Universal approximation bounds for superpositions of a sigmoidal function. *IEEE Transactions on Information theory*, *39*(3), 930-945.

301

Bibliography

- Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems*, *2*(4), 303-314.
- D'Inverno, G. A., Bianchini, M., Sampoli, M. L., & Scarselli, F. (2021). A unifying point of view on expressive power of GNNs. *arXiv preprint arXiv:2106.08992*.
- Hammer, B. (2000). On the approximation capability of recurrent neural networks. *Neurocomputing*, *31*(1-4), 107-123.
- Hornik, K., Stinchcombe, M., & White, H. (1989). Multilayer feedforward networks are universal approximators. *Neural networks*, *2*(5), 359-366.
- Funahashi, K. I. (1989). On the approximate realization of continuous mappings by neural networks. *Neural networks*, *2*(3), 183-192.
- Garg, Vikas K., Stefanie Jegelka, and Tommi Jaakkola. "Generalization and representational limits of graph neural networks.", ICML, 2020.

302

Bibliography

- Gori, Marco, and Alberto Tesi. "On the problem of local minima in backpropagation." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 14.1 (1992): 76-86.
- Gori, M., & Scarselli, F. (1998). Are multilayer perceptrons adequate for pattern recognition and verification?. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *20*(11), 1121-1132.
- Huang, G. B., Zhu, Q. Y., & Siew, C. K. (2006). Extreme learning machine: theory and applications. *Neurocomputing*, *70*(1-3), 489-501.
- Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." ICML, (2016).
- Judd, J. S. (1990). *Neural network design and the complexity of learning*. MIT press.

303

UNIVERSITÀ DI SIENA 1240

Bibliography

- Lawrence, S., Tsoi, A. C., & Giles, C. L. (1996, June). Local minima and generalization. In *Neural Networks, 1996., IEEE International Conference on* (Vol. 1, pp. 371-376). IEEE.
- Hochreiter, S., Bengio, Y., Frasconi, P., & Schmidhuber, J. (2001). Gradient flow in recurrent nets: the difficulty of learning long-term dependencies.
- Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., & Grohe, M. (2019, July). Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI conference on artificial intelligence* (Vol. 33, No. 01, pp. 4602-4609).
- F Scarselli, AC Tsoi, M Hagenbuchner, The Vapnik–Chervonenkis dimension of graph and recursive neural networks, Neural Networks 108, 248-259

304

UNIVERSITÀ DI SIENA 1240

Bibliography

- F Scarselli, M Gori, AC Tsoi, M Hagenbuchner, G Monfardini, Computational capabilities of graph neural networks,IEEE Transactions on Neural Networks 20 (1), 81-102, 90, 2008
- F Scarselli, M Gori, AC Tsoi, M Hagenbuchner, G Monfardini, The graph neural network model, IEEE Transactions on Neural Networks 20 (1), 61-80
- Shawe-Taylor, J. (1993). Symmetries and discriminability in feedforward network architectures. *IEEE Transactions on Neura Networks*, *4*(5), 816-826.
- Yu, X. H., & Chen, G. A. (1995). On the local minima free condition of backpropagation learning. *IEEE Transactions on Neural Networks*, *6*(5), 1300-1303.

305

UNIVERSITÀ DI SIENA 1240

Bibliography

- Vapnik, V. N. (1999). An overview of statistical learning theory. *IEEE transactions on neural networks*, *10*(5), 988-999.
- Xu, Keyulu, et al. "How powerful are graph neural networks?." ICLR,2019
- Zhang, Chiyuan, et al. "Understanding deep learning requires rethinking generalization." *arXiv preprint arXiv:1611.03530* (2016).
- Zou, D., Cao, Y., Zhou, D., & Gu, Q. (2020). Gradient descent optimizes over-parameterized deep ReLU networks. *Machine learning*, *109*(3), 467-492.

306

UNIVERSITÀ DI SIENA 1240

Thank you for your attention!

307

UNIVERSITÀ DI SIENA 1240

Lecture of September 20th

- The lecture will start at 9.00 am
- We will have a test
  - Written
  - Close answers
  - About 1,5 hour time to complete
- The test is mandatory for students of Smart Computing PhD who want to have to pass an exam to have the credits,
  - but it is open to anybody
- After the test, we will discuss the responses, thus this will be used as a course summarization

308