cesare alippi tommaso marzi gabriele dominici

teaching material

slides

- seceted papers prerequisites

– deep leraning lab – machine learning

course evaluation

one assignment (demo + report 90)

a quiz exam at the end of the course (10)

1. intros to graph 2. convolution of graphs 3. spatiotemporal graphs (the time invariant fails? and reacting some) 4. learning NSE 5. Mich Bronstein ,Erlangen, continous models for GNNs

17 may

24 mayp

WE START at 8:50 !!

to 12 :10

why graph?

so in many applications graphs come naturally

for example a molecula

it is a graph obviously

so the neurons are actually the graphs too

so in social nettworks the graphs are also natural

so the people are entities

so you have a team of people and you have the different relationships with those,

so you have several group of strong ties

then you have the another group of weak ties

and the temporary ties would be the outsiders

and the weight depends

we can derive the graphs from the time series,

sensor network, and they produce data, and there is a space and they generate a signal of the time and there is a temporal dimension too

so there is a power grid , and we hav e a set of measuring sensors, and they are physically connected of course by the cables. so it is naturally construts to the graphs, because there are nodes which are not connected

it is not functional though, it is just spatial

so in social networks we have a functional dependency in the graph,

so we'd like to see if the node is connected by the others, if one node is consuming the same as the other, they are probably have functional links

so we are , in addition to the spatial connections, searching the functinoal ones

so we have a time series for every day,

and we can decide how those signals related to the other.

causality dependency if one signal can be reconstructed to be the other one

so those graphs are connected through several things at a time

so the node attributes in this thing

how much power consumed so temperature date time holidays/weekdays number of inhabitants and the WEALTH of course A VECTOR OF ATTRIBUTES!!

edges attributes here

SIGNAL corellation (pearson) weather correlation (if one house in shadow, and the one is not or yes) and cable length

imagine that it is social network

so the strength of connection in edges (maybe number of tweets or message)

information related what you buy (political dependency), so we can try to go through a string inference

for example our group is strong tie, and the recommendation is more orelss comes naturally from that

we took a signal and build a graph

so we can of course take a time window, and beuild a graph, and another window, and the information can a graph will be generated

so you will end up with graph stream, and processing the GRAPH STREAM WILL bring you closer to the solution of those graphs

so the applications

social networks ofc

Monti et al fake news detection

traffic prediction, deepmind traffic

physics, graph neural networks in particle physics

recommender systems

ying et al conv

RL and graphs

too

do we really need to work with graphs?

why shou ld we

if we have graph, so we have the nodes associations and the edges , for example affinity inbetween nodes (arrived from the nodes), so all the information is in the metrics

so why we can't AI DL architecture (we have no need of that graph structure)

this is sometimes true, but in some cases not

but in some cases You actually have a graph ! it is natural!

so you have a chemical compound, which interacts with the body and compound s are interacting

so Ithe nodes and the interaction are transformed

can we argue that the edges are actually just a way how to guide the ML with a thing

some issues philosophical

- so end-to-end learned - hand engineered

tradeoof between those - rich reperesentations, the graph processing - inductive bias, something

so the inductive bias is an artifact, which allows a learning algorithm to prioritze over another (efficiently driving learning towards practilar regions)

so the goal the preprocessing is to guide the learning process

so the gradient is not actually looking the features,

the inductive bias is actually just something like is a prior

and the changes with that prior ARE REALLY BRUTAL

prior info can be encoded in the architerctuer

so the graphs are usually inferred from the systems which has relational nature

so the complex systmes can be a composition of niteracting entitise

so those relational things are VERY serious, so you'll probably better not to ignore that

the world is usually compositional or we understand the compositional

for example things on the table are related to them, not only by the table, by the physical proximity too!

so if there are relations

we can start with a physical relations

we have same colour hydrogen, and the bond between the atoms are bidirectional

in n-body system we have a bit of elemnts which are moving in space

so in each timme t the distances could be computed, so we will generate a graph steram

fortext sentences you can generate sytacitcal tre

in mass spring system

so you have a spring which is a continuos

you can finite leemnt the sprign, and the nodes will be collected, and the mass will interact with the elemnts of the spring

so the interaction can be achieved

so th eleemnts of a rigid body systems also can interact inbetween the closed system, iso the desk will become a node of a graph, and will be

so you can segment the picture and say that the entities are somehow connected

from signal to graphs

so we can ues the signal values and use the values to create the graph

so the horizontatl visibilty graph can be generated if you'll assign the node each feauter for a speific time

so you can think about the position, and think what cany ou physically see from the position of looking back, and then they are connected. If you cannot see other things, you are not connecting those

another example of graph extraction

eecg

so you segment the heartbeats

so you apply the the horizontal vision graph

and you convert this eecg into the graph steram

and the anomaly is now associated with the graph!!

so we have another way to process the signal !

types of graph

os the directed graph is one-way direction, in causality from a sender to receiver

thu undirected is actually bidirectied

multigraph – MULTIPLE EDGES BETWEEN VERTICES (also self edges)
attributes
so the attribute can be atrtibute of a node (e.g. v_i)
attribute can be associated with the edges, e.g. e_k
or GLOBAL ATTRIBUTES, so the entire graph
for example for a chemical compound, a solubility
for a lugano city, the power that you need tomorrow at twelve
so the global attribute can be used as sort of HIGH LEVEL info in the graph
so the tasks
can be node-focused, how much power will you spend tomorrow
the edge focused are about the relations and interactions inbetween entities
the graph -focuesd tasks is about lal the energy the ssytme generate
how to to represent the graph
so we are effective in processing th vector, tensor, and something like that
so you start from a graph, so you need to generate
is the description in the topology of the graphs
the infromation associated with the nodes
and the info associated with the edges
so the A is the binary adjacency matrix, $n \times n$, where n is nunmber of the nodes
so for the node 1 we have an association
so it is ssociated with node 3 and 4
so we put in the adjacency matrix for row 1 the 1-s in the column 3 and 4 and zeroes in all others
so we represent the topology of the graph
so for node you just puth the row of the features))
for edge features is a companion for the adjacency matrix (weights of the edges)
so the binary adjacency matrix is the good way to decouple those
the elements are generated PRIOR to the learning process
sometimes you need to learn the topology of a graph
so sometimes you learn the weights or the topology only
of course sometimes you can fix the topology, but the weight can be changed
so you have a vouleme , which depends the cardinality of the vector
so can we procsess the blocks keeping the fact that we have the relationships of structures
so we need to generate the new operators for that
so there are CNNs for examples
it takes the advantage of the spcae locality principle
we have some prior
the big cells satisfy the space locality principle
so if we have a pixel (i,j) we have a higher probability that nearby pixels are related
so if we have a neighborhood and some features of those pixels are related
so the layers of covolution and pooling are then globally pooled, so a vector, and you have FFN at the end

Figure 1:

so each layer computes a higher abstract representation, w.r.t. the previous one

the convolution layers

in the mona lisa, we start thwith a pixel, we look at the 3x3 neighboorhood and calculate the kernel function on that

so the convolution mask (kernel) $I * K$

so we have several kernels, and learn those

and the kernel here is a filter

so the characterisitc of the filter is learned mostly

many filters are applied ni parallel

each one filter K is different (is K unique or not ?)

so we got multiplied the amout of features, beacuse of multiple filters

so after that you will get the pooling layer

so you can do max-pooling (choosing maximum in a locality) or average pooling

so we have the locality principle and then we can take the pooling of a location, because we know that THOSE ARE RELATED

there are different poolings

so you can use activation function after the convolution or after the pooling

so the functional proximity coincides with the physical proximity inth

so the graph can be constructed in the image pixels, because they are physicall related

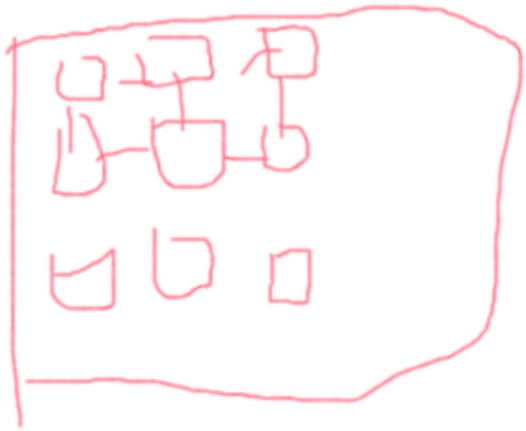


Figure 2:

so in graph this is not necerssaryl to be nodes conicide the functionally and physically, but heth locality principle is satisfiied

we can create a first neightboorhod of distance one and the neighboorhod ofthe second distance (green)

so here you'll have a convolution for that type of graph

so you can have a convolution filter for those types of the graphs

and we have a convolution over the first spaec (yellow)

and the convolution over the second neighboorhood

so we have neighborhood-based convolution filte r

so we have a one neighboorhod x_1 and get the z_1 for its neightborhood (for node representation)

so we see that message passes, and we want to update the status based on your negihborhood states, and the each edge status (other)

and all the every of those nodes will have the same convolution filter

you can then pool the aggregated nodes

so you start with a covoluted image nad get the pooled result

so in graph you want to have the graph topoolgy shrinked

so the pool here will collapse the nodes, so we need to destroy some of the nodes, and redefien the edges betwen those

global pooling you will need to turn a set of images to vector and then you go into classifier in example

embed the graph to a vector , so the simplest way you can just put the nodes int the vector, or you can increase the complexity, and add somet hngis

so we get a GNN by interleavintg the opertaors

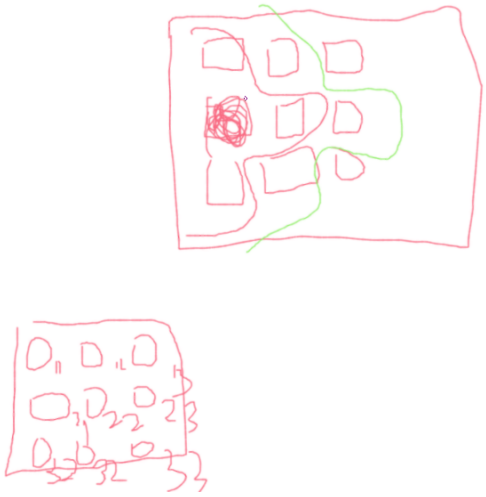


Figure 3:

so also we can , instead of putting nodes to the vector, have a single node after some point of pooling

so you can then do whatever you want

you also can apply the autoencoder to the type of structure

so you will have a lantent space after the graph convolutions, and then apply the decoder, which will reconstruct the architecture which will produce a new graph !

so the matrix-topology and the matrix of node features and edge weights matrix as a result of those

you of course can destroy the decoder part and create the FFN in the end, and receive output

SVMs)

vanilla operation framework is just to embed the graph stream into the vector, and then process the vector , it is the very easy

so let's to understand graphs and emedding spaces

attrbuted grahps is a big family of graphs

extract only one character and transform the characters to graphs

so the topology can change with time

Also the vertices can be unidentified

We have a set of vertices and attributed

So do those entities generate a probability space

For example by inducing graph distance metric

So the probability of generate a type of graph,and so is the graph distance happens

We are not sure how to define the probability

Figure 4:

Let's see some enzymes then
You can ask to generate enzyme
And so you can ask to generate the graph
And generate the features too!
How the components interacts because of their structure??
we have pdfs which generate the different topologies!!
How to define the graph distances?
If the structure is not difficult if you have the same structure
But the graph edit distance can help us to provide the metric to compare different topology
So we want to construct one A from another A
We need to add one node!! Or remove one
And we are at the attribute level
Node insertion or delete
We change mode
We change an attribute
We don't know the cost of operations, so we need to learn it
Some of the embedding operations
We need to move from graph to a vector then!
Distance based methods rely on distance
So we try to map the graph to vector
- dissimilarity repr

We also can approach the neural

Autoencoders

So the idea of dissimilarities representation

Set of diff graphs

You extract the prototype graphs according to distribution or just some random graphs

So having those graphs will have to generate a vector

So we will evaluate the graph in relation to one prototype: and generate the number

And then the second prototype generates the number

And so on

So the new space will be very different from the previous , it is highly no linear

Multidimensional scaling

You will get a number of prototype but will try to preserve the inter distances in the new space

So new space will try to preserve the distances between the prototype generated projections???

So we try to use vectors to make things easier

So in comparison of Euclidean and manifold those are very different

The curvature is zero

We also have want to have a way to present the non Euclidean spaces

Geometric deep learning has those computed

We try to constraint the embedding, but move to non Euclidean

So we can use the same method, it can be computed easily

You can control the curvature of the embedding?

Autoencoder

The latent space cannot guarantee that the graph closed one will be proximal in latent space

So we need to use the scaling And also we can imply the prob distribution on the latent space?

So if we have a distribution in graph space, we would like to impose it in latent soace

So we use adversarial encoders in here

Autoenxoder here

In addition to the autiencider

So you get the distribution from latent space and the prior one and combine those to make those relatablr

So you do one step of the gradient descent And no advers part

You use the discriminator after to check that the probability distribution of the space real or fake

So you try to discriminate ample from real distribution and sample from latent space

So the thrid step you want to confuse the discriminator so it will confused

So you try the update initial encoder to fool the discriminator

so the encoder will try to be like the distribution

So you'll try to impose the same distribution

At the end this is min max converges here and will be fine

You can add a regularizer for distribution but it won't be better and be less effective

vae

So you start at a random picture

And you'll discriminate between real pictures and artificial

Can that make the architecture

You can make use of the inductive bias in this case

You always can have a deep architecture attached to graph

But you'd like to preserve the

So in VAE you are controlling the distribution of the latent space

Within the adversarial mechanism we try to imply the distribution on the

You can do it through penalty and you'll trade off it Linear algebra You you'll imply the eigenvalues in graph theory

One vector x and we applying A so if the A is nxn so if you apply A to x you get the new vector $x = Ax$

Is there any interesting direction for x?

We want to find the A which makes scaled x

And the A which multiplies x to zero

So preserving the direction is interesting for us

$Ax = \lambda x$

So you need to say something

Equation $(A - \lambda I)x = 0$

So x is true eugen vectors and lambdas are eigenvalues

They cannot be computed if det of subtraction is 0

So you compute the determinant and get the lambdas

So you solve the Linear systems with those lambdas

So the privileged directions are found like so from vectors

So you use the Spectral theorem

You try the shrink the size of those vectors

You rely on spectral theory

So the covariance matrix can be regenerated

So this A matrix can be seen as a composition of matrixes products of eigenvalues and eigenvectors covariance

So little lambdas can be ignored in PCA because

So you can have a desire a only the relevant info

So we can use only eigenvalues and associated vectors and use only those

Very effective way of sending messages

Reduces the complexity of input

Consider matrix A

So then you get the eigenvectors

So then you can plot those as their values and their indices in 2D

So we can also try to make the A to graph

Those eigens are looking pretty much like signals

We consider mostly the undirected graphs here

The degree of node is amount ooc attached nodes

If we consider the A matrix and multiply it with 1-vector we get the degrees of the nodes in the topology so the

diag(D)= d

So we also to see the laplacian matrix

$L = D - A$

So you take the degree matrix and then subtract topology from it

So laplacian is symmetric

Also it is semi definite posititve

So the number of zero in laplacian eigenvalues equals the number of connected components

If the graph is fully connected, there is only one null eigenvalues

Case of chain graph

For example time

So we get the laplacian eigenvectors we see that it has spectral nature

So you can move the spectral things

So the laplacian is looking at the flow of a field and you try to constrain the flow

So the Laplacian is looking on the field about the space

So it converts

Torch soatiotemporal spatiotemporal processing

Spektral building gnns

Cdg anomaly felldetection

Dts multi step time series

Graph Deep Learning

SP 2024

Prof. Cesare Alippi

Università della Svizzera italiana

The people you will see

Cesare **Alippi**
(me)

Tommaso **Marzi**
(TA)

Gabriele **Dominici**
(TA)

Teaching material

- **Slides** provided by the lecturers and TAs (iCorsi3 platform)
- **Selected papers** given to the students
- **Prerequisites**:
  - Machine Learning
  - Deep learning lab

Course evaluation

- One assignment (demo+report) and its presentation to the class (90% of the score)
- A quiz exam at the end of the course (10%)

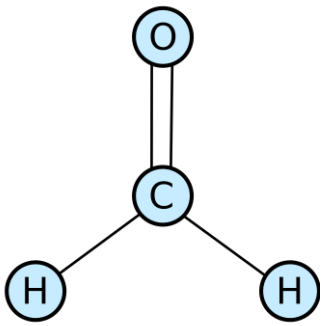
Program and time schedule

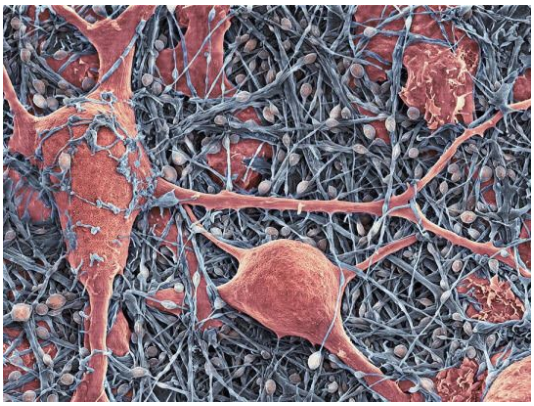
Date	Speaker	Topic
23 February	We	Introduction to graphs
1 March	We	Graph convolution and graph pooling
8 March	We	Spatiotemporal graphs
15 March	We+	Learning NSE
22 March	Michael Bronstein (U.Oxford)	Erlangen program of ML; Continuous models for GNNs
17 May	Students	Students presentations
24 May	Students	Students presentations

- On Fridays, Room D1.14
  - First slot: 8:50- 10:20(+)
  - Second slot: 10:40- 12:10(+)

Why graphs

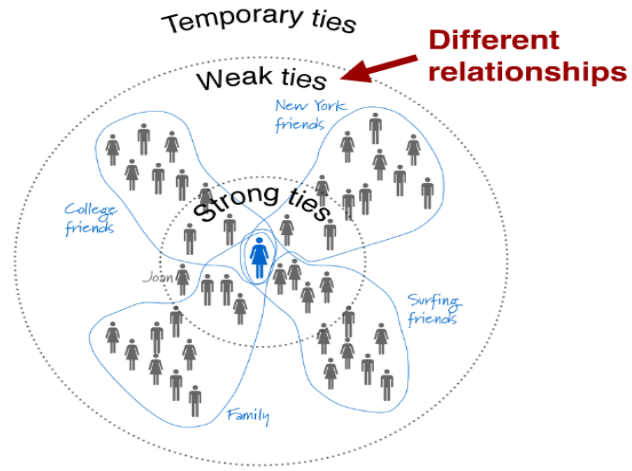
In many applications graphs come naturally





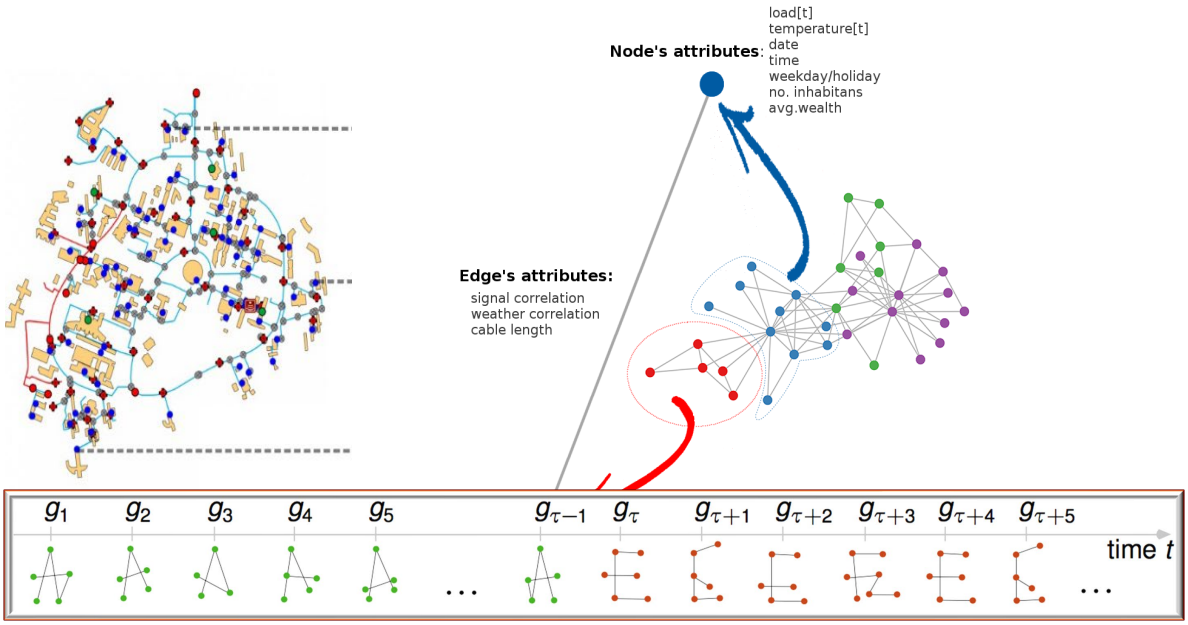
Why graphs

In others are latent

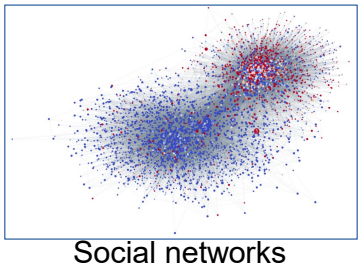


Data streams and graph streams

In others, we derive graphs from timeseries (signals)

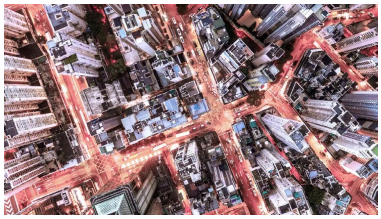


A plethora of applications



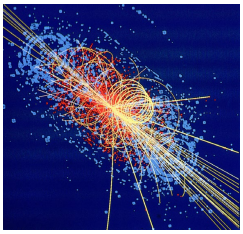
Social networks

Monti et al. "Fake news detection on social media using geometric deep learning."



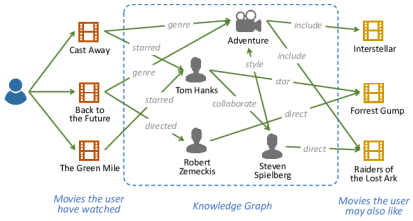
Traffic prediction

"Traffic prediction with advanced Graph Neural Networks" https://deepmind.com/blog/article/traffic-prediction-with-advanced-graph-neural-networks



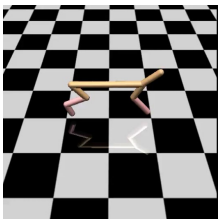
Physics

Shlomi, Jonathan, and Peter Battaglia. "Graph neural networks in particle physics."



Recommender systems

Ying et al. "Graph convolutional neural networks for web-scale recommender systems."

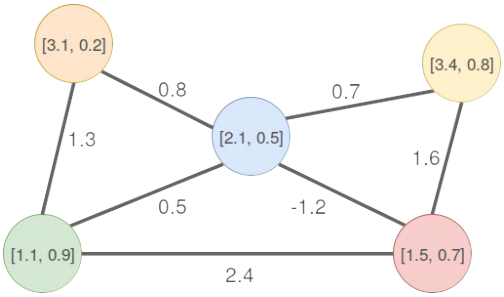


Reinforcement learning

Zambaldi et al. "Relational deep reinforcement learning."

Do we really need to represent and process graphs in a different way?

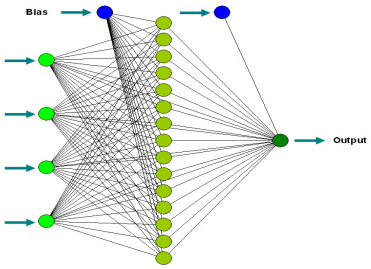
We might argue that having node features is enough provided that you have a strong inference engine (say) able to extract functional interdependencies.



e.g., Pearson's correlation coefficient

X =

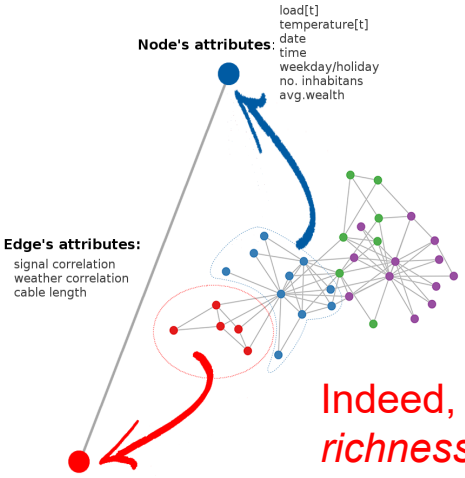
3.1	0.2
3.4	0.8
2.1	0.5
1.1	0.9
1.5	0.7



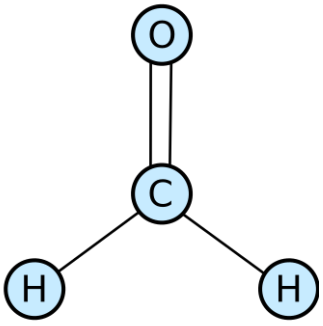
Do we really need to represent and process graphs in a different way?

That is true *in principle* in many cases and *partly* – whereas not – in others

in principle



not true



Indeed, this is related the *information richness of node features*

Some philosophical issues

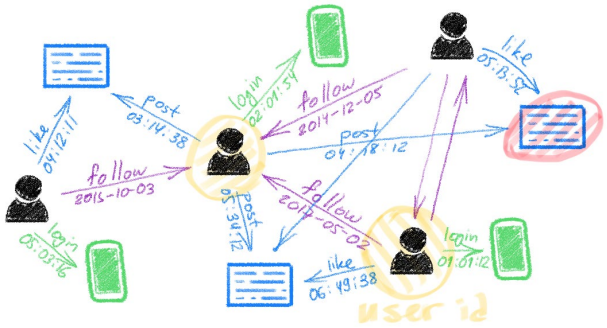
- Previous comments are related to the way we design features
  - End-to-end learned (e.g., DL)
  - Hand-engineered
- A school believes that we should place ourselves in-between by taking advantage of both
  - Rich representations
  - Inductive bias

Inductive bias

- *An inductive bias is an artifact that allows a learning algorithm to prioritize one solution over another*, say efficiently driving learning towards particular regions of the search space
- Prior information can be encoded in the architecture of the solution itself (e.g., we believe that our model is linear).
- Inductive bias improves the search for solutions, in general without diminishing performance.

Relational inductive bias

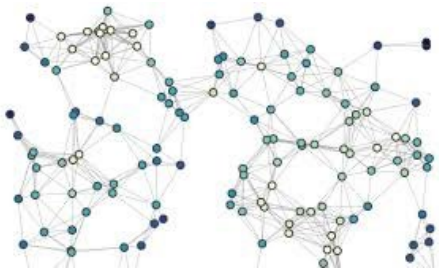
- Complex systems can be seen as a composition of interacting entities



- Not rarely, the world is compositional, or at least, we understand it in compositional terms.

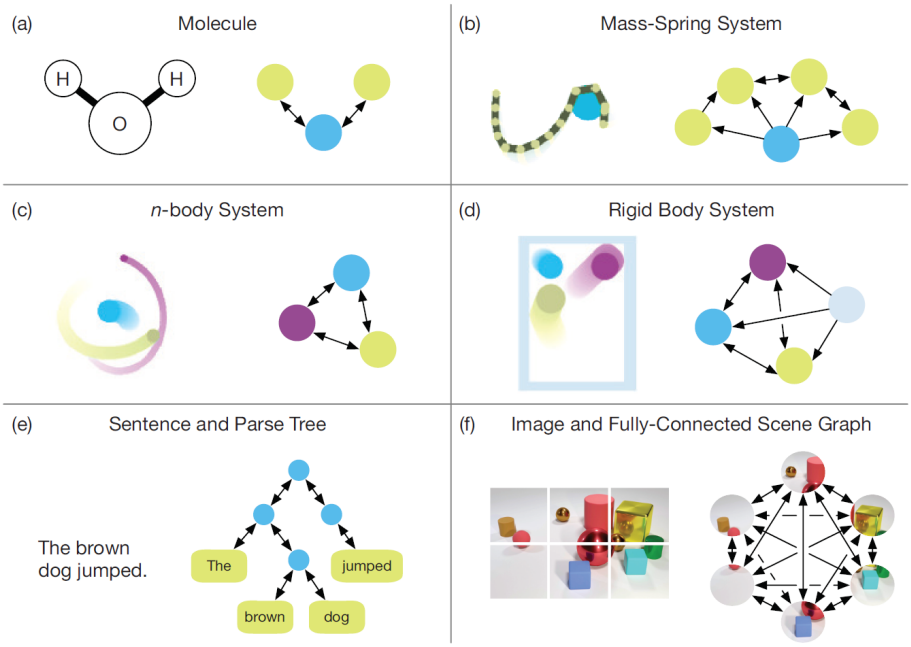
Relational inductive bias

- Recent research has proposed a class of models at the intersection of deep learning and structured approaches, which focuses on reasoning about *explicitly structured data*, in particular **graphs**



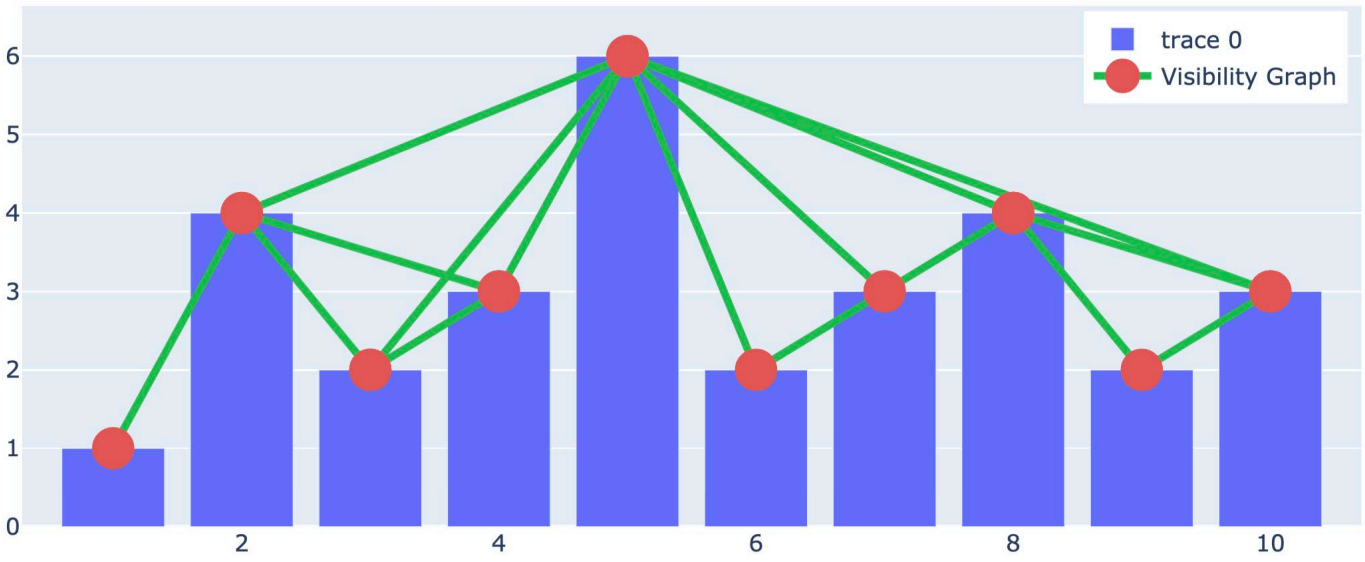
- We have entities (graph nodes)
- We have relations (graph edges)
- How to design a graph, represent information at node level and learn that assigned to edges?
- In other terms, how to compose and populate entities and relations and compute their implications to solve a task?

Graphs and graph representations

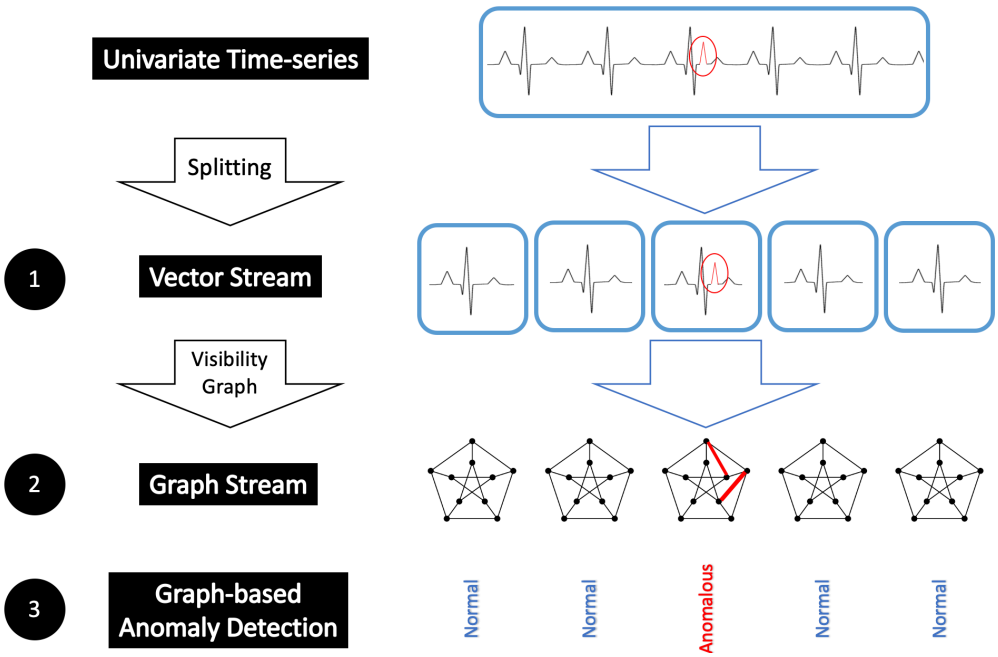


Graphs and graph representations

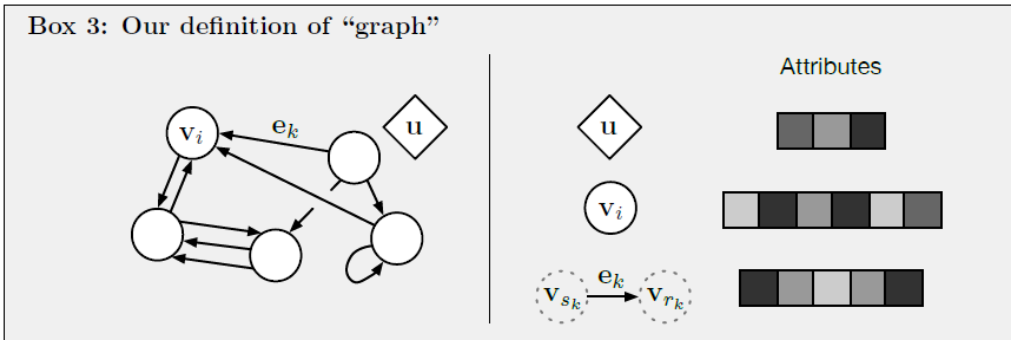
From signals to graphs: e.g., *horizontal visibility graph*



Graph extraction: example

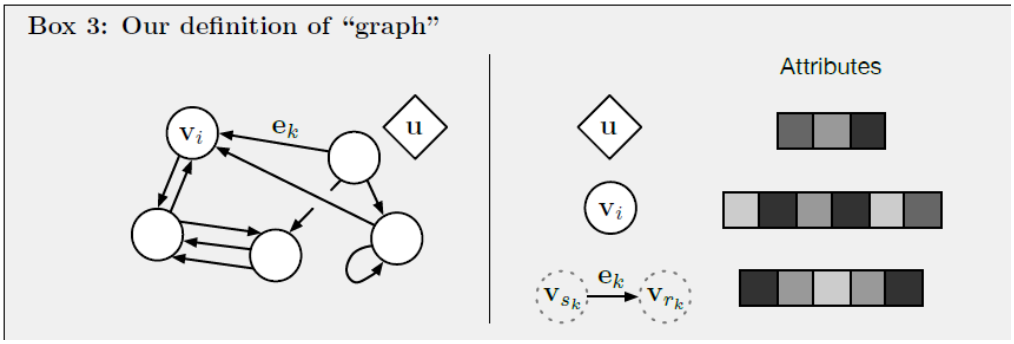


Graphs and graphs…



- Directed graph: one-way edges, from a sender node to a receiver node;
- Undirected graph: bidirectional edges;
- Multi-graph: there can be more than one edge between vertices (including self-edges).

Graphs and graphs…



- Attributes: properties that can be encoded e.g., vector, tensor, set, another graph, a model.
- Attributed graphs: edges and vertices have attributes associated with them.
- Global attribute: an attribute at the graph-level.

Graph processing

- In node-focused tasks features of nodes are our output, e.g., to reason about physical systems
- In edge-focused task edges represent the output we are interested in, e.g., to make decisions about interactions among entities
- In graph-focused tasks the entire network attributes constitute the output, e.g., to predict the potential energy of a physical system, the properties of a molecule, or answers to questions about a visual scene

How to represent a graph?

A

0	0	1	1	0
0	0	1	0	1
1	1	0	1	1
1	0	1	0	1
0	1	1	1	0

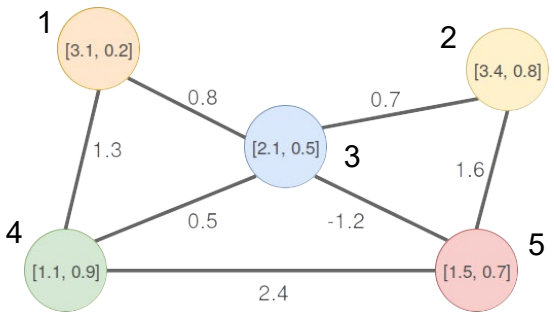
X

3.1	0.2
3.4	0.8
2.1	0.5
1.1	0.9
1.5	0.7

E

0.	0.	0.8	1.3	0.
0.	0.	0.7	0.	1.6
0.8	0.7	0.	0.5	-1.2
1.3	0.	0.5	0.	2.4
0.	1.6	-1.2	2.4	0.

- **A**: binary adjacency matrix
- **X**: node features
- **E**: edge features



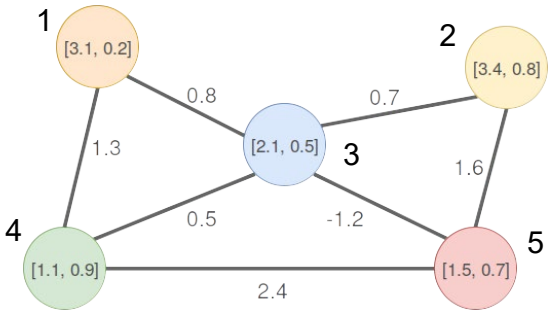
How to represent a graph?

X

3.1	0.2
3.4	0.8
2.1	0.5
1.1	0.9
1.5	0.7

A *in* E

0.	0.	0.8	1.3	0.
0.	0.	0.7	0.	1.6
0.8	0.7	0.	0.5	-1.2
1.3	0.	0.5	0.	2.4
0.	1.6	-1.2	2.4	0.



- **X**: node features
- **E**: edge features

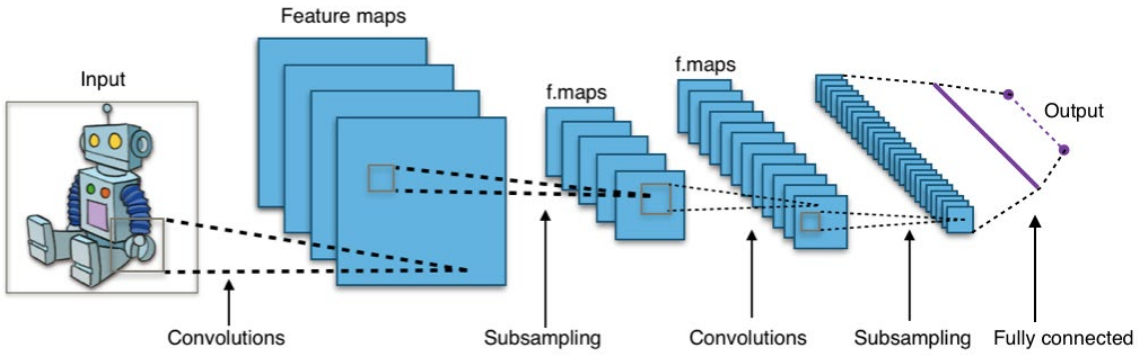
Yes, finally, we have matrices (or tensors)

Processing blocks

“a smooth transition to graph processing operators and architectures”

CNN: Deep Convolutional Networks

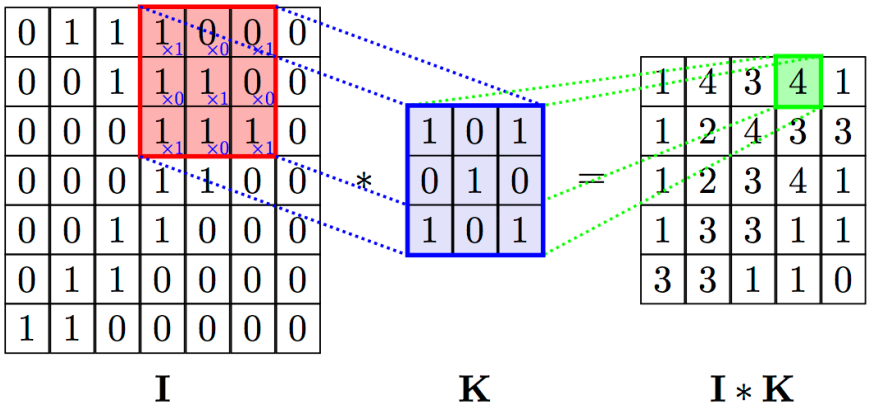
- A CNN takes advantage of the space locality principle (inductive bias)



- Subsequent steps of **convolutional** and **pooling** layers.
- Each layer computes a higher abstract representation w.r.t. the previous one; the image size shrinks at each step.

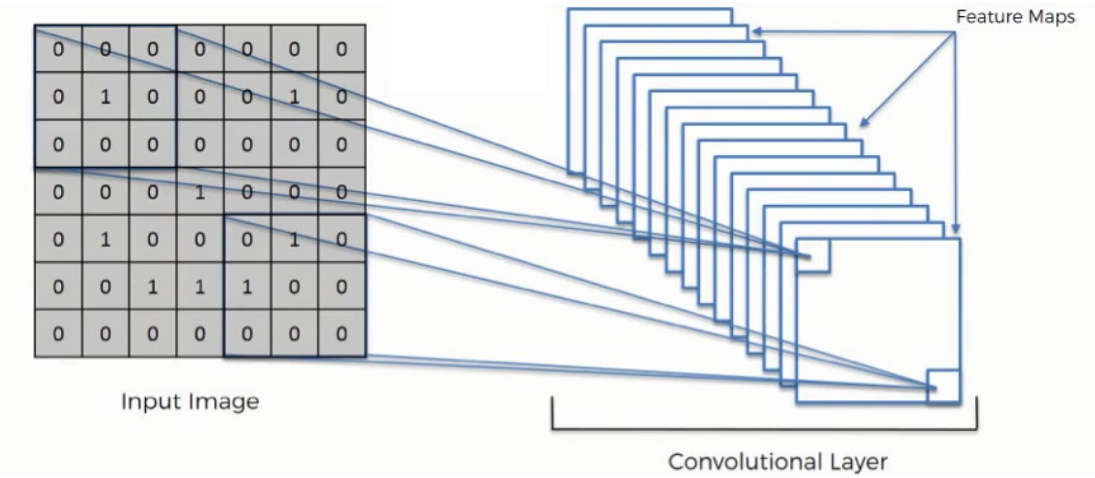
Convolution operator

- Convolutional layers: evaluate affinities based on the principle of locality.
- Receptive field applied to the image with a stride.
- The kernel/filter **K** contains learnable parameters.



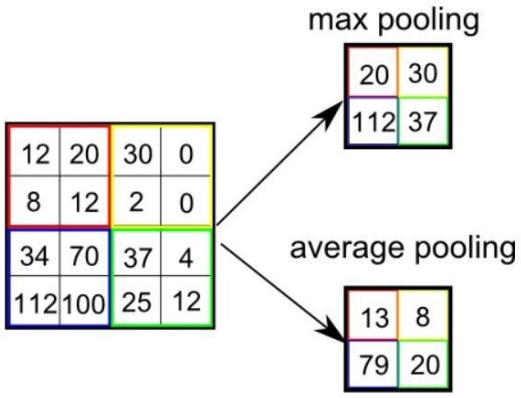
Convolution layer

- Many filters can be applied in parallel.
- As each one is learned, filters **K**s are different; after convolution, each one provides a different **feature map**.



Pooling operator

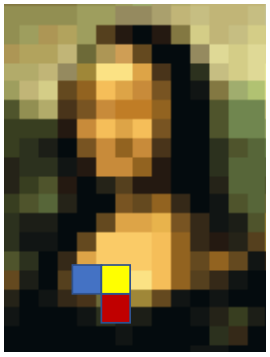
- Pooling layers reduce the image size based on some rules.



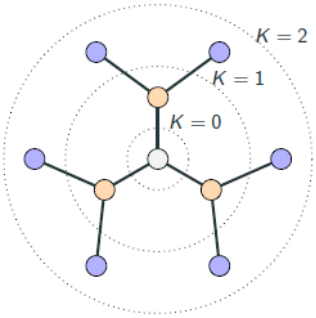
- Different pooling operators can be designed e.g., based on local properties.

Graph processing: Graph Neural Networks

- “Mutatis mutandis” we can naively extend the CNN to a GNN (Graph Neural Network)
- In images, functional proximity coincides with physical proximity

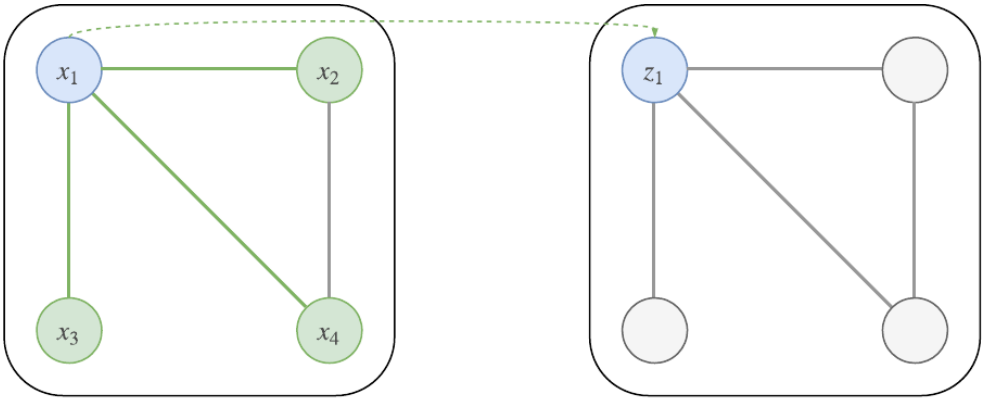


- In a graph, functional proximity does not necessarily coincide with physical proximity; yet the locality principle is there…



GNN operators: Graph Convolution

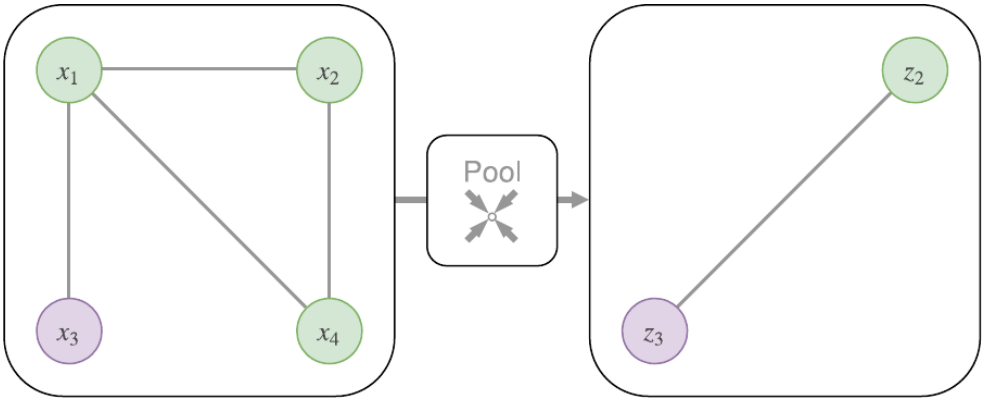
- Graph convolution exploits the local neighborhood of each node to compute a node value embedding



- The above convolution is at node level, but we might have information associated with edges too
- “Message passing” as an extension of the convolution mechanism

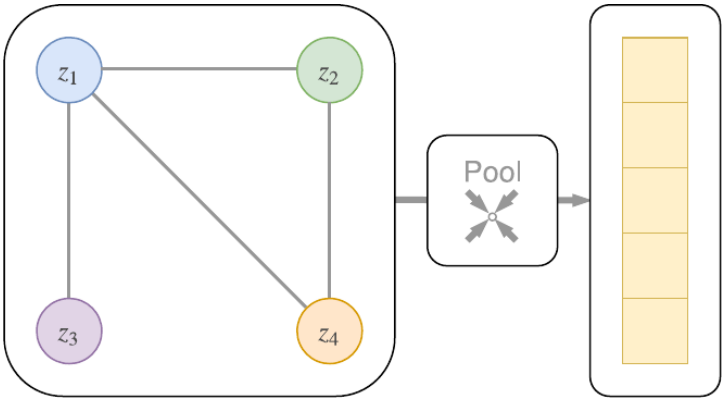
Graph Pooling

- Pooling aggregates nodes (shrinks the graph topology) to
  - provide a more abstract representation of the graph
  - reduce the graph complexity



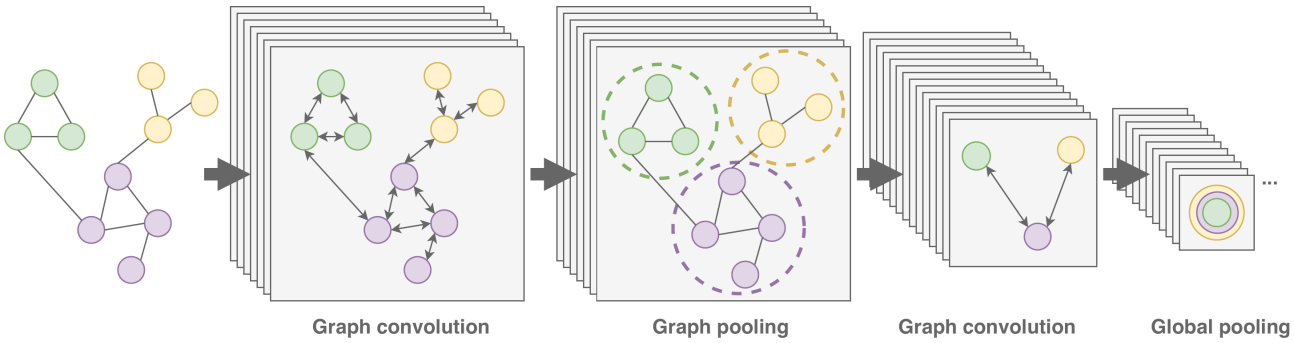
Global Graph Pooling

- Global pooling embeds the graph to a vector (more to come later)



GNN: Graph Neural Networks

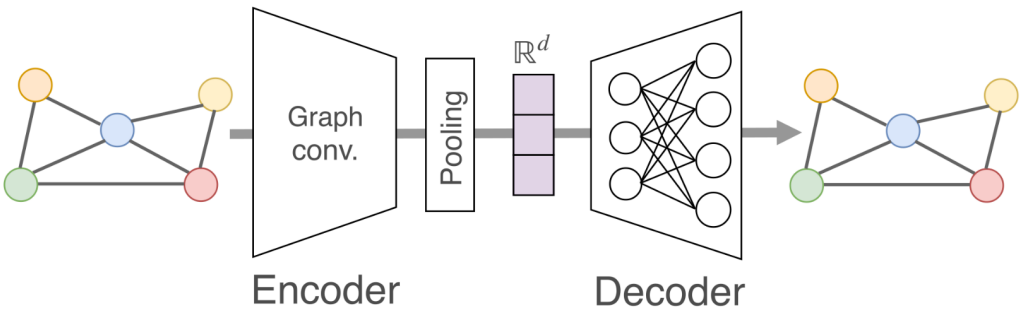
- We get a deep network – GNN – by interleaving operators



- Indeed, you can enjoy “conceptual transfer” of neural processing to other architectures…

Graph autoencoders

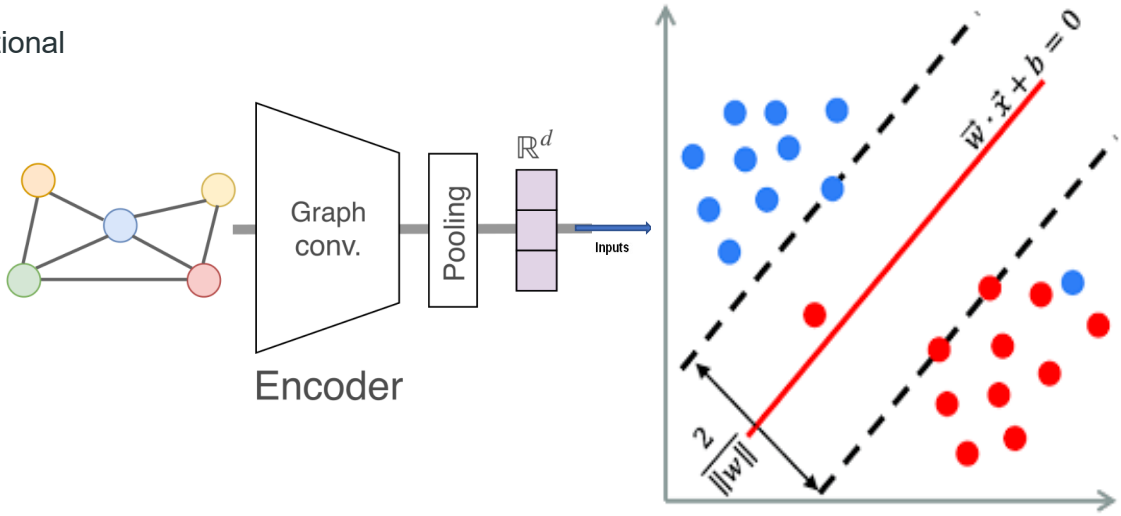
- The encoder is composed of graph convolutional layers with the pooling one
- A dense decoder reconstructs the matrices describing the graph



- The latent space represents a natural embedding

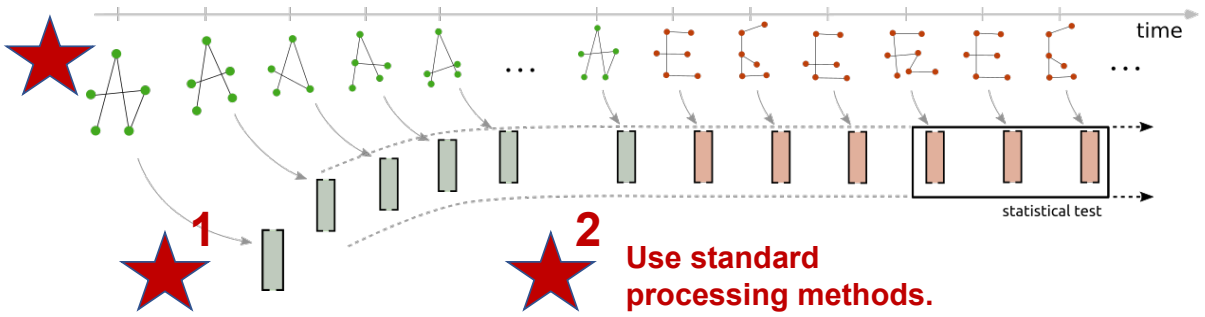
Latent space representation

- Once we have a graph-to-vector mapping (embedding) we can apply our favorite processing:
  - Neural
  - Traditional



The «vanilla» operational framework

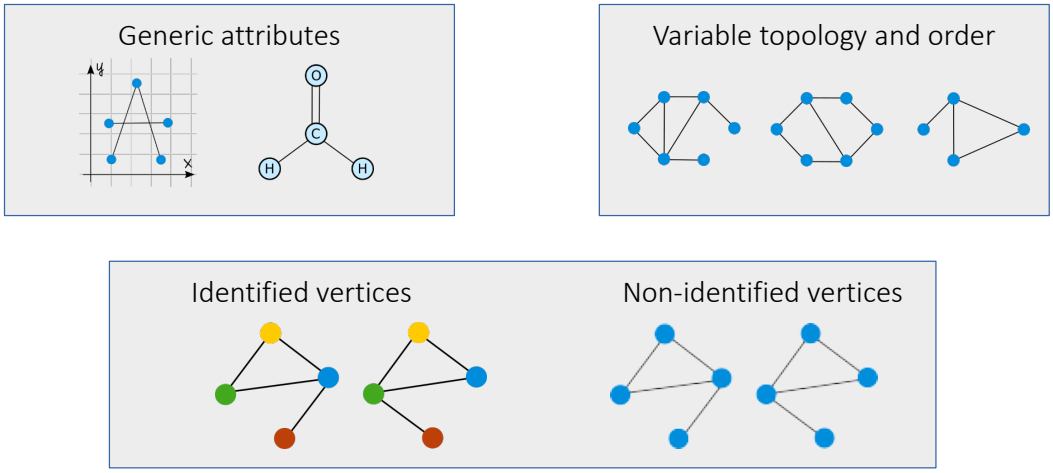
- Map graphs to vectors (graph embedding)



Graphs and embedding spaces

Which types of graphs are we interested in?

Attributed graphs represent a very large family of graphs



The graph space $(\mathcal{G}[\mathcal{A}], \mathrm{d})$

- Set $\mathcal{G}[\mathcal{A}]$ of graphs $g = (V, E, a)$
  - V , E (finite) sets of vertices and edges
  - $\mathcal{A}$ *:* set of attributes
  - ***a***: attribute function

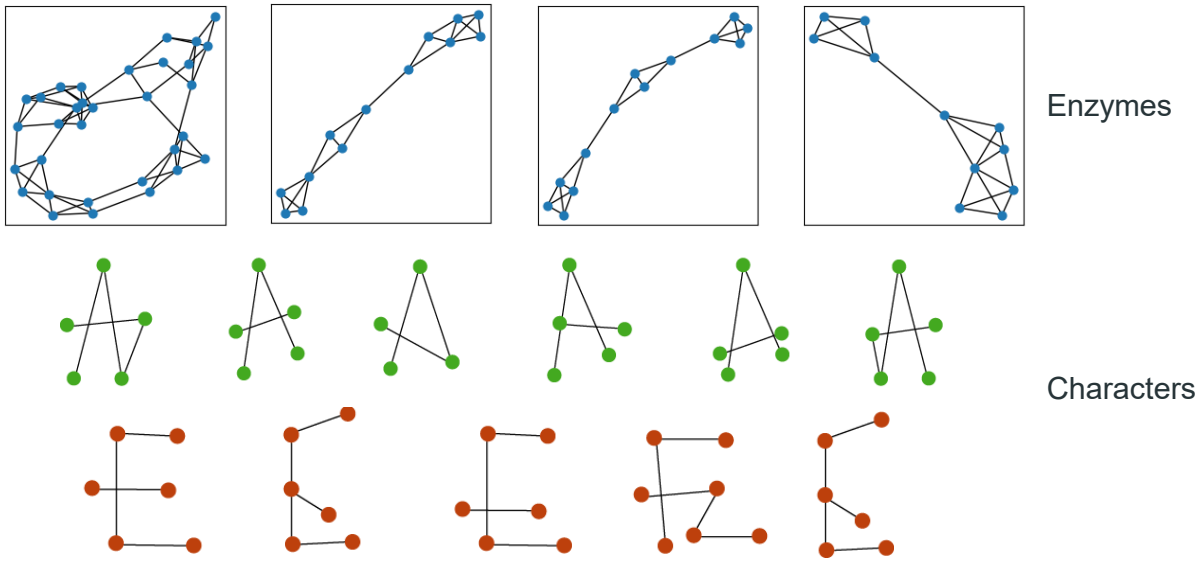
$$a : V \cup E \to \mathcal{A}$$

- Graph distance $\mathrm{d}(g_i, g_j)$
- On $(\mathcal{G}[\mathcal{A}], \mathrm{d})$ we can define a probability space

B.J. Jain, On the geometry of graph spaces, Discrete Applied Mathematics, December 2016.

Stationarity and graphs

- Mind, both topology and attributes can change under the stationarity hypothesis



Example of $\mathrm{d}(\cdot,\cdot)$: the graph (edit) distance

Sequence of edit operations to generate graph g_i starting from graph g_j

Operations:

- node insertion/deletion
- edge insertion/deletion

$$cost(o) = k(a,a)$$

- node modification
- edge modification

$$cost(o) = k(a,a) + k(a',a') - 2k(a,a')$$

$$\mathrm{d}(g_1,g_2) = \sqrt{\min_{(o_1,\ldots,o_m)} \sum_{i=1}^{m} cost(o_i)}$$

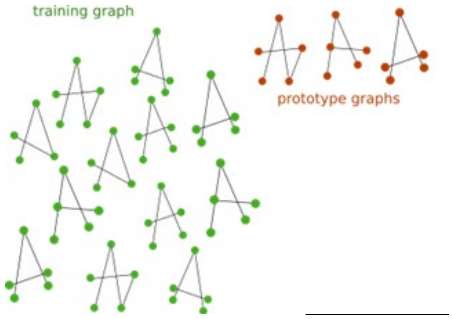
J. Jain, On the geometry of graph spaces, DAM, 2016.

Embedding methods (some)

- **Distance-based methods**:
  - Dissimilarity representation
  - Multi-dimensional scaling
- **Neural approaches**:
  - Autoencoders – already seen
    (latent space embedding)
  - Adversarial learning
    (inducing constraints on the latent space)
- **Train free random approaches** (later)

Dissimilarity representation

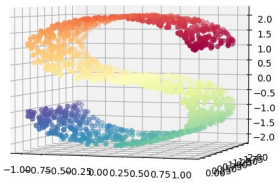
- Training phase:
  - Identify a set of prototypes R

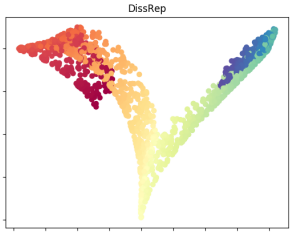


- Out-of-sample technique:

$$g_i \mapsto x_i = \begin{bmatrix} d(g_i, r_1) \\ \vdots \\ d(g_i, r_M) \end{bmatrix}$$

$$R = \{r_1, r_2, \ldots, r_M\}$$

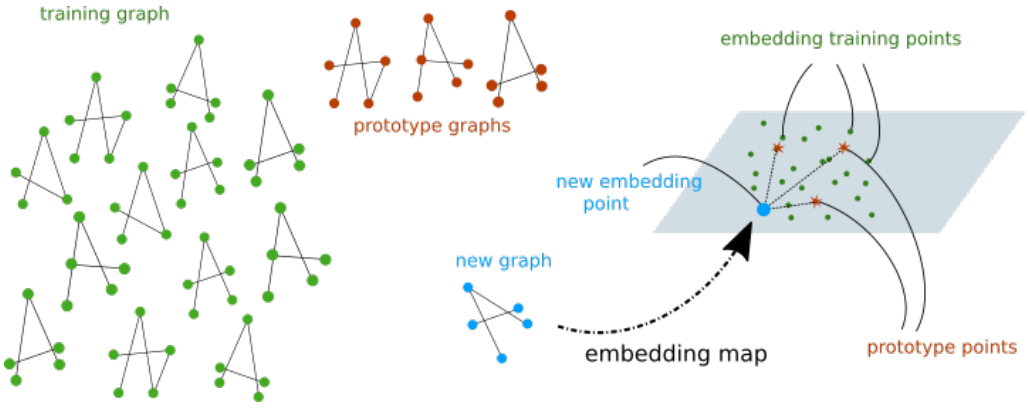




D. ZE. Pękalska and R. P. W. Duin, The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. WS, 2005.

Multi-dimensional scaling

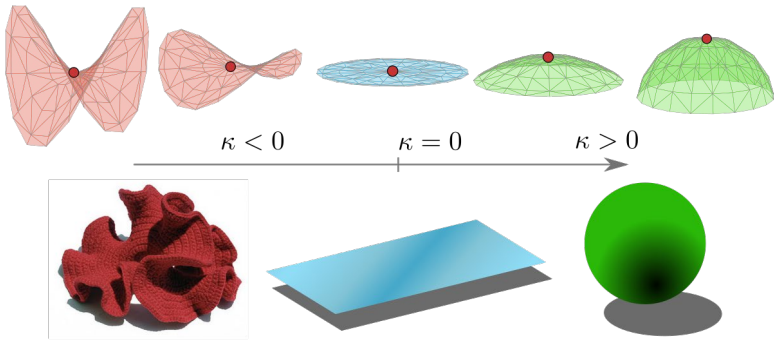
- Training phase:
  - Identify a set of prototypes
- Out-of-sample technique:
  - Use the dissimilarity representation by attempting to preserve the distance from prototypes



D. ZE. Pękalska and R. P. W. Duin, The Dissimilarity Representation for Pattern Recognition: Foundations and Applications. WS, 2005.

A step beyond Euclidean embedding

- Real-world data sets are often non-Euclidean
- Constant curvature Riemannian manifolds
  - Riemannian manifolds provide a metric structure
  - Mathematics is relatively easy (geodesic, distribution)



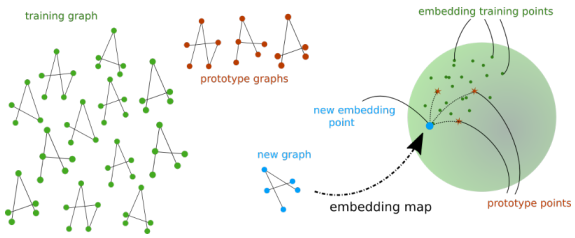
D. Zambon, L. Livi, and C. Alippi, Anomaly and Change Detection in Graph Streams through Constant-Curvature Manifold Embeddings, IEEE IJCNN, 2018

Embedding on constant curvature manifolds

Similar to MDS, but the optimization is on the manifold.

Training phase

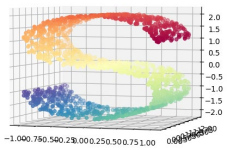
- Select a set of prototypes

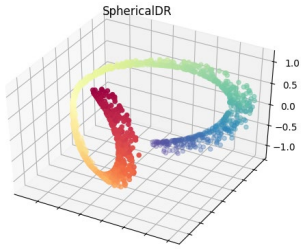


Out-of-sample technique

- Builds on the dissimilarity representation by attempting to preserve the distance from the prototypes

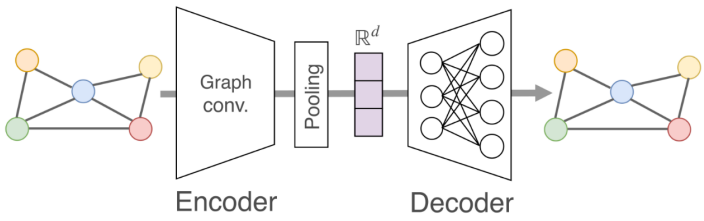
$$\rho(x, y) = \frac{1}{\sqrt{\kappa}} \arccos(\kappa \langle x, y \rangle_\kappa),$$



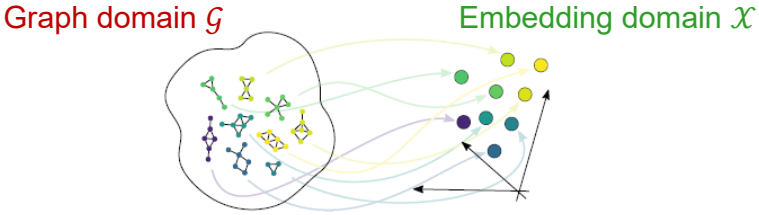


Graph autoencoders

- Autoencoders provide a nice way to automatically build the embedding

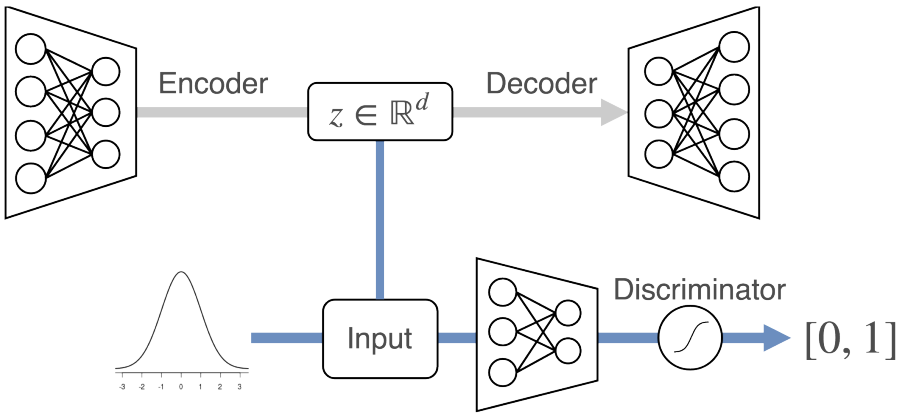


- But neither we can grant that the distance nor that the concept of distribution are preserved in the graph/embedding spaces



Adversarial autoencoders

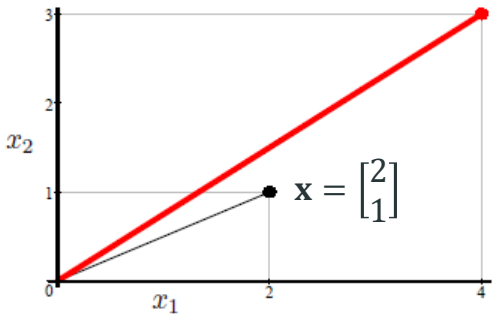
- Match the distribution of embedded information in the latent space with an arbitrary (given) prior
- In this way we impose the concept of distance and a wished distribution in the embedded space



Linear algebra (recall): a matrix as an operator

- Consider an n-dimensional real column vector x and a square matrix $\mathbf{A}$ of order n
- By taking $\mathbf{Ax}$, matrix $\mathbf{A}$ can be seen as a function (operator) mapping $\mathbf{x}$ to vector $\mathbf{x}' = \mathbf{Ax}$

$$\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$$



Eigenvalues and eigenvectors

- Is there any “interesting direction” for $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$?

- Well, one might be the one in which $\mathbf{x}$ is transformed into a $\mathbf{x}' = \mathbf{Ax}$ and both vectors share the same direction, the second eventually scaled in magnitude.
  This request can be modelled as

$$\lambda \mathbf{x} = \mathbf{Ax}$$

Eigenvalues and eigenvectors

- Equation $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$ is called the *eigenvalue equation*
- The λs for which the equation holds are called *eigenvalues*, and associated $\mathbf{x}$s are named *eigenvectors*
- A theorem states that the eigenvalue equation has non null solutions iff $\det(\mathbf{A} - \lambda\mathbf{I}) = 0$
- Therefore:
  - We have n eigenvalues;
  - Given eigenvalue λ, the eigenvector satisfies equation $(\mathbf{A} - \lambda\mathbf{I})\mathbf{x} = 0$

Eigenvalues and eigenvectors

- Simple example $\mathbf{A} = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$
- Eigenvalues satisfy

$$\det(\mathbf{A} - \lambda\mathbf{I}) = \det\left(\begin{bmatrix} 2-\lambda & 0 \\ 0 & 3-\lambda \end{bmatrix}\right) = 0$$

- Yielding to $\lambda_1 = 2,\ \lambda_2 = 3$

$$(\mathbf{A} - \lambda_1\mathbf{I})\mathbf{x} = \begin{bmatrix} 2-\lambda_1 & 0 \\ 0 & 3-\lambda_1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 0 & 0 \\ 0 & 1 \end{bmatrix}\begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = 0$$

Eigenvectors $\mathbf{u}_{\lambda_1} = \begin{bmatrix} x_1 \\ 0 \end{bmatrix};\ \ \mathbf{u}_{\lambda_2} = \begin{bmatrix} 0 \\ x_2 \end{bmatrix}$

- Privileged directions are $x_2 = 0,\ x_1 = 0$

Spectral Theorem

- Consider a square symmetric matrix $\mathbf{A}$ of order n, then,

$$\mathbf{A} = \sum_{i=1}^{n} \lambda_i \mathbf{u}_{\lambda_i} \mathbf{u}_{\lambda_i}^T$$

Each eigenvector is scaled to have magnitude 1 (*orthonormality*).

- In our example:

$$\mathbf{A} = 2\begin{bmatrix}1\\0\end{bmatrix}\begin{bmatrix}1 & 0\end{bmatrix} + 3\begin{bmatrix}0\\1\end{bmatrix}\begin{bmatrix}0 & 1\end{bmatrix}$$

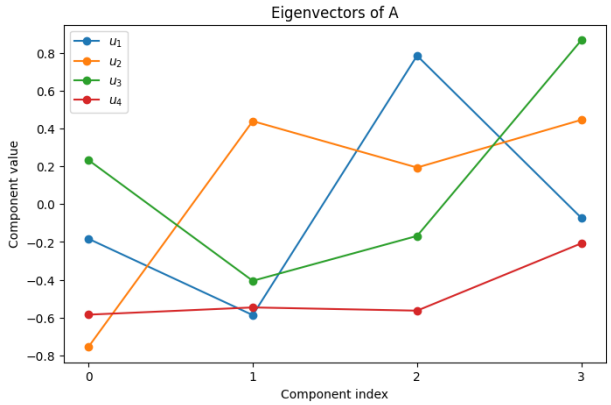
- The spectral theorem is very much used in PCA to reduce the complexity of the input space (it is applied to the covariance matrix of the inputs)

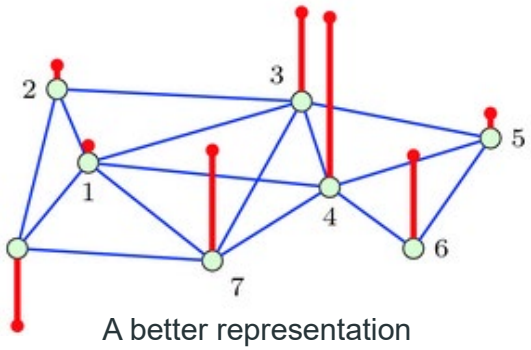
Eigenvectors

- Consider matrix

$$A = \begin{bmatrix} 2 & 3 & 4 & 2 \\ 3 & 1 & 5 & 0 \\ 4 & 5 & 0 & 1 \\ 2 & 0 & 1 & 1 \end{bmatrix}$$

- Plot eigenvectors and intend them as signals (whatever it does mean):

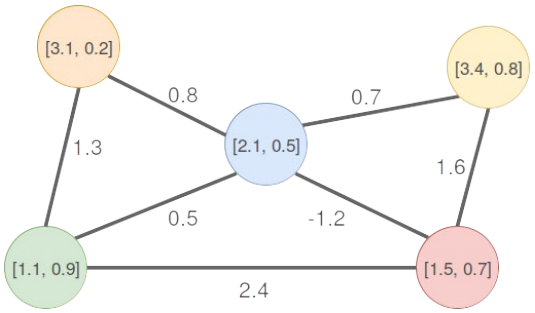




A better representation

Graphs: formalization: recall

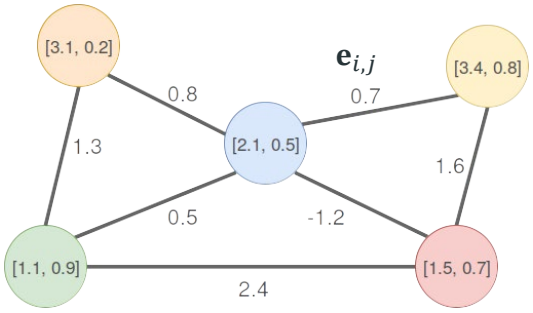
- Let a graph be a tuple $G = (V, E)$ where $V = \{1, \dots, n\}$ is the *node set* and $E \subseteq V \times V$ is the *edge set*.



- Each vertex can be associated with a real-valued *attribute* (or *feature*) vector. We indicate with $\mathbf{x}_i \in \mathbb{R}^{d_v}$ the d_v-dimensional attribute of the i-th node.

Graphs: formalization

- Edges can also have real-valued features. We indicate with $\mathbf{e}_{i,j} \in \mathbb{R}^{d_e}$ the d_e-dimensional attribute of edge (i, j).



$$\mathbf{A} = \begin{bmatrix} 0 & 0 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 1 & 1 & 0 & 1 & 1 \\ 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix}$$

- The *adjacency matrix* of a graph is a square matrix $\mathbf{A}$ whose entries are given by

$$\mathbf{A}_{i,j} = \begin{cases} 1, & \text{if } (i,j) \in E \\ 0, & \text{otherwise} \end{cases}$$

Graphs: formalization

- In an *undirected* graph, edges are unordered pairs of vertices. Alternatively, we can say that in an undirected graph $(i, j) \in \Leftrightarrow E(j, i) \in E$.
- The adjacency matrix of an undirected graph is symmetric.
- In this course, we mostly consider undirected graphs.
- The degree of a node in an unweighted graph is the number of nodes attached to it.
- If edges are weighted, then the vector of degrees $\mathbf{d}$ is

$$\mathbf{d} = \mathbf{A1}$$

- Define $\mathbf{D}$ to be a diagonal matrix so that

$$diag(\mathbf{D}) = \mathbf{d}$$

The Laplacian

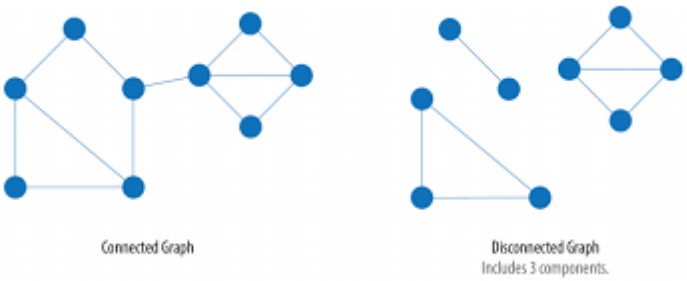
- In addition to the adjacency matrix there are other interesting matrices derived from that.
- The *Laplacian*

$$\mathbf{L} = \mathbf{D} - \mathbf{A}$$

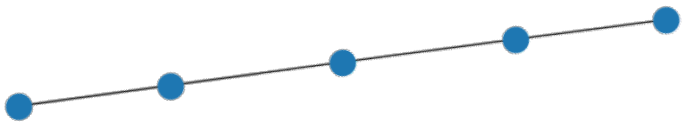
- The Laplacian is
  - Symmetric ($\mathbf{L}$ coincides with its transpose)
  - Semidefinite positive (all its eigenvalues are non-negative)

The Laplacian

- The Laplacian embeds several important properties of the graph:
  - The number of zero eigenvalues of the Laplacian (i.e., the multiplicity of the 0 eigenvalue) equals the number of connected components of the graph.
  - If the graph is fully connected, then there is a single null eigenvalue; its eigenvector is a constant vector for all nodes in the graph.



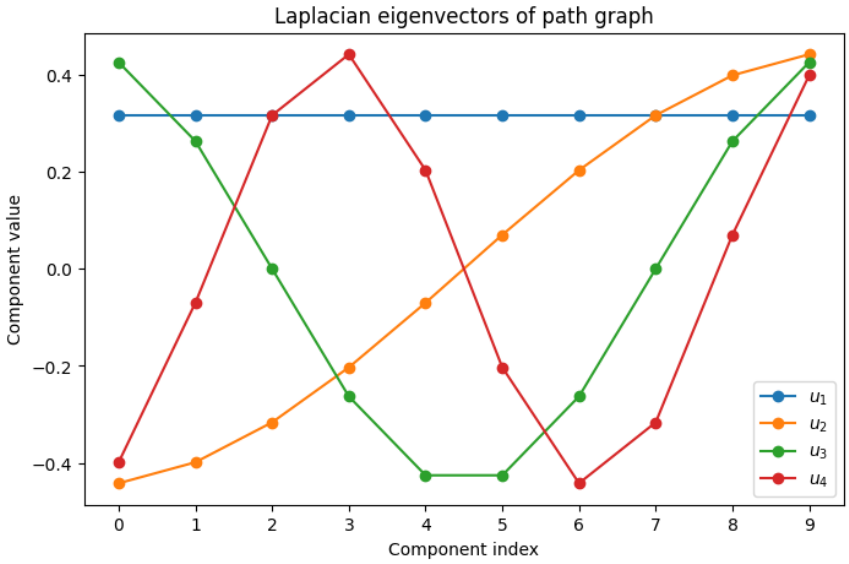
The case of a *chain graph*



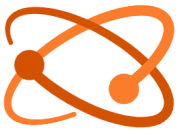
$$A = \begin{bmatrix} 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 \end{bmatrix} \qquad L = \begin{bmatrix} 1 & -1 & 0 & 0 & 0 \\ -1 & 2 & -1 & 0 & 0 \\ 0 & -1 & 2 & -1 & 0 \\ 0 & 0 & -1 & 2 & -1 \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

Eigenvectors

The plot of eigenvectors resembles that of signal



Our libraries



Torch Spatiotemporal

A library for neural spatiotemporal data processing, with a focus on Graph Neural Networks.

Spektral

A library for building graph neural networks in Keras and Tensorflow.

CDG

A Python library for detecting changes in stationarity in sequences of graphs.

DTS

A Keras library that provides multiple deep architectures for multi-step time-series forecasting.