Pattern Recognition Lab

FRIEDRICH-ALEXANDER UNIVERSITÄT ERLANGEN-NÜRNBERG

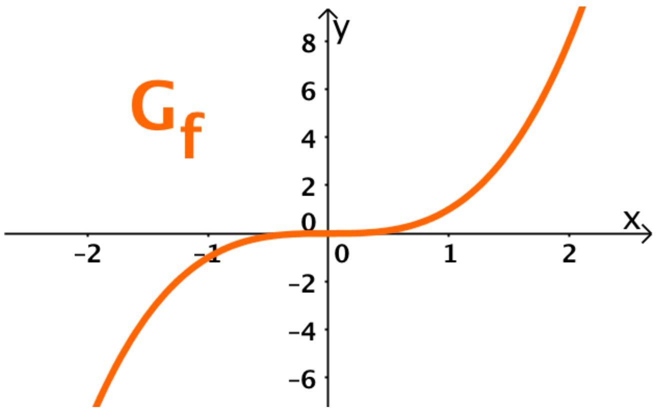
FACULTY OF ENGINEERING

Graph Deep Learning
Part 1

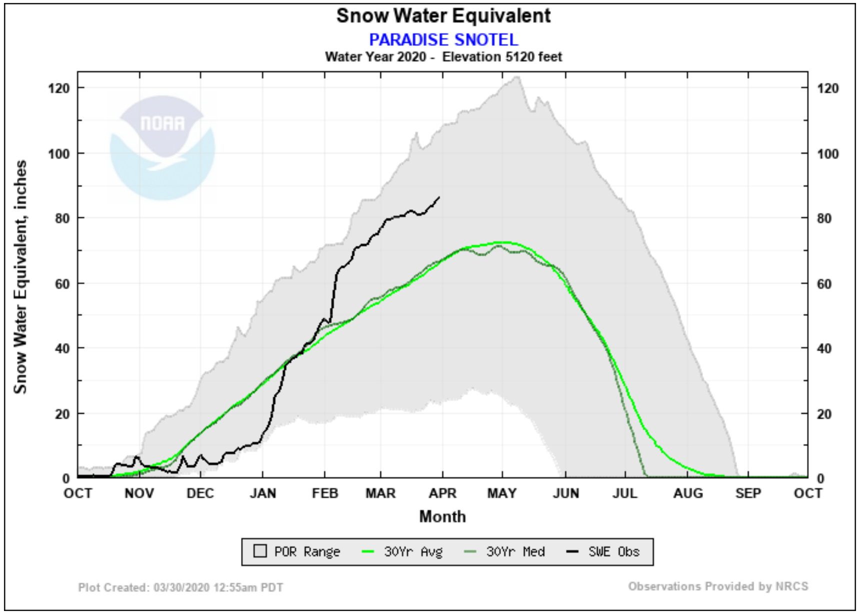
A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, C. Bergler, F. Denzinger, B. Geissler, Z. Yang, A. Popp, M. Nau

Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

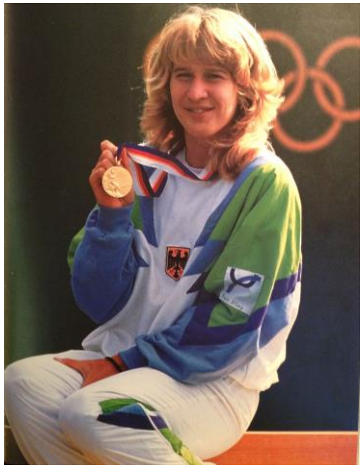
Graph Deep Learning



[a]



[b]



Steffi Graf (1999)

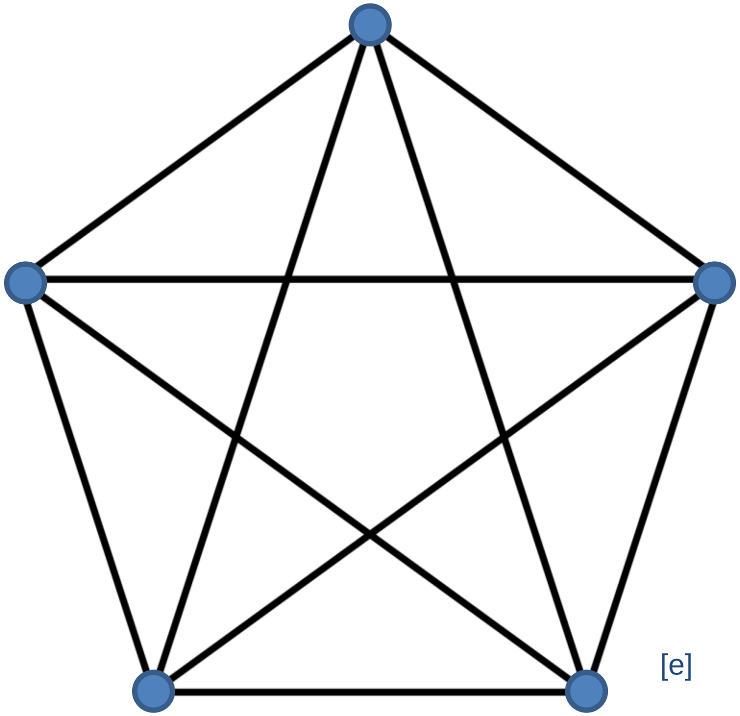
[c]

[d]

Graph Deep Learning

Computer Scientist:

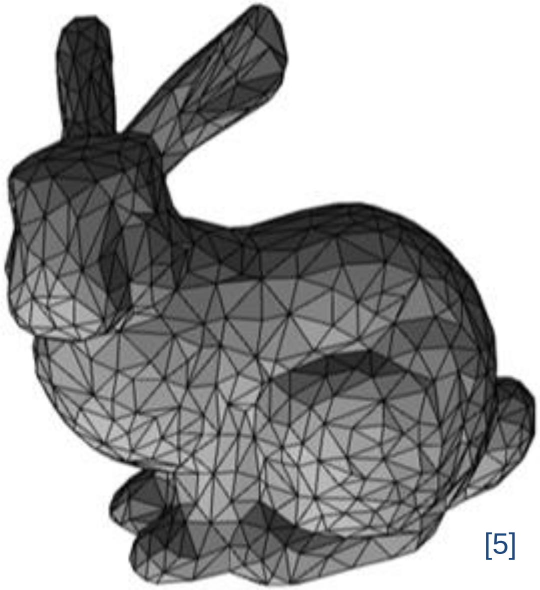
A **Graph** is
a set of nodes
connected through edges



[e]

Mathematician:

A **Graph** is
a manifold
but a discrete one



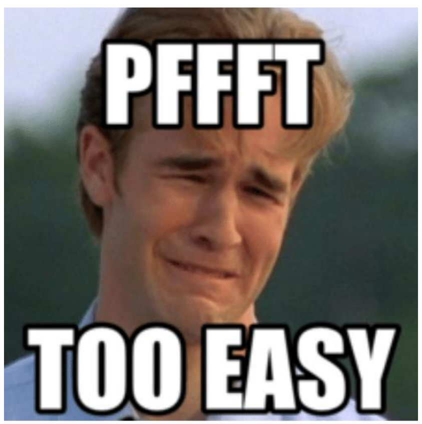
[5]

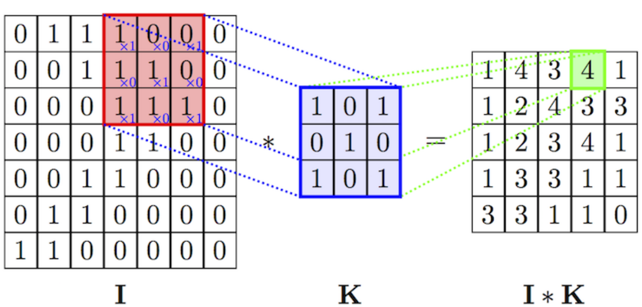
How would you define a convolution on Euclidean space?

Computer Scientist + Mathematician:

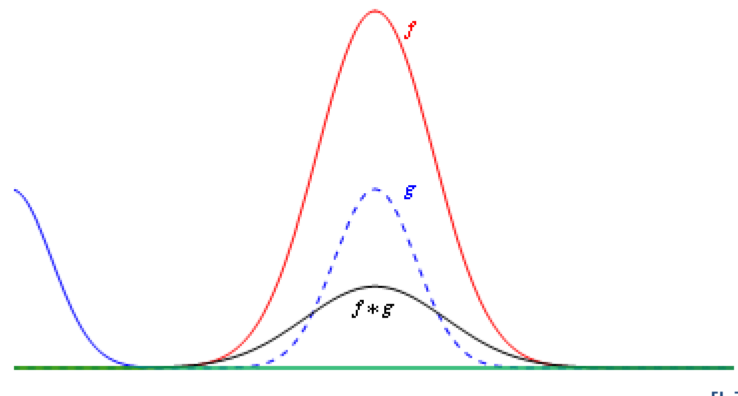
$$(f \star g)(n) = \sum_{k \in D} f(k) g(n-k)$$

$$(f \star g)(n) = \int_{\mathbb{R}^n} f(\tau) g(x-\tau) d\tau$$





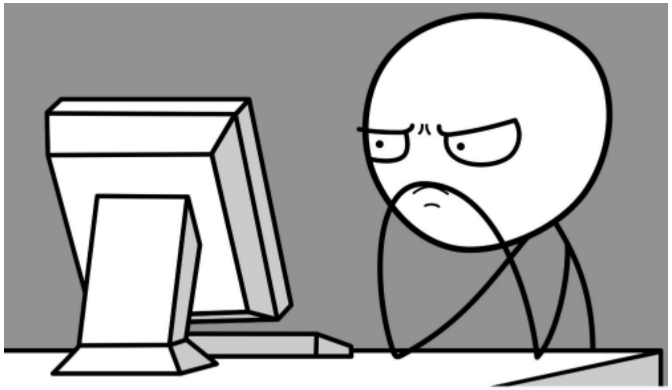
[g]



[h]

How would you define a convolution on graphs?

Computer Scientist:



Mathematician:

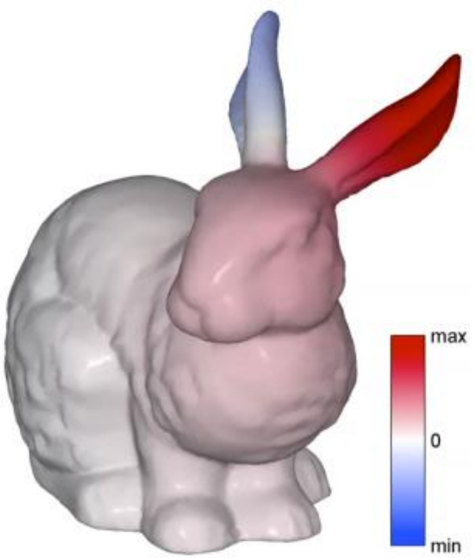
$$\Delta f = -\mathrm{div}(\nabla f)$$

How would you define a convolution on graphs?

Manifold Idea:

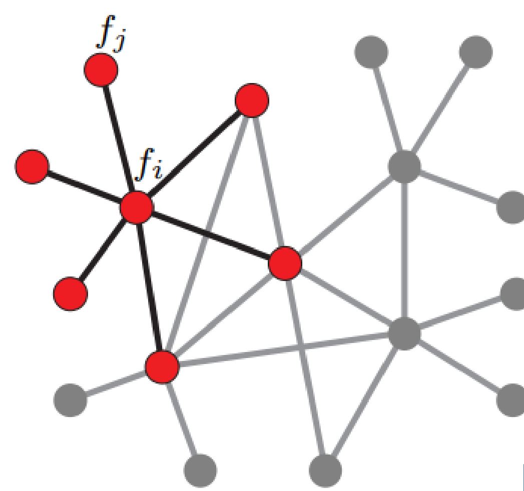
1. We know to convolve manifolds
2. We can discretize convolutions
3. Thus, we know how to convolve graphs

Convolution on manifolds



[5]

Convolution on graphs



[5]

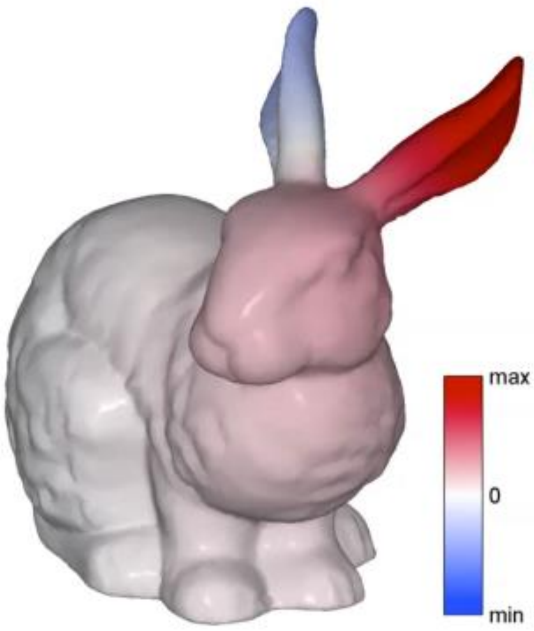
Graph Deep Learning

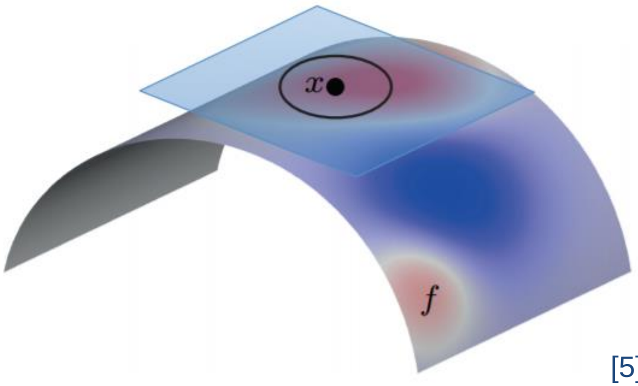
Lets diffuse some heat with Newton's Law of Cooling [6]:

$$f_t(x,t) = -\Delta f(x,t)$$

$$f(x,0) = f_0(x)$$

- $f(x,t)$ amount of heat at point x at time t
- $f_0(x)$ initial heat distribution
- $\Delta f(x) = -\,\mathrm{div}(\nabla f)$ (Laplacian) difference between $f(x)$ and the average of f on an infinitesimal sphere around x





[5]

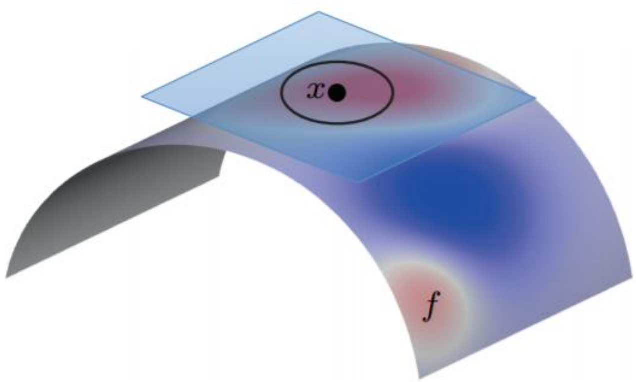
Graph Deep Learning

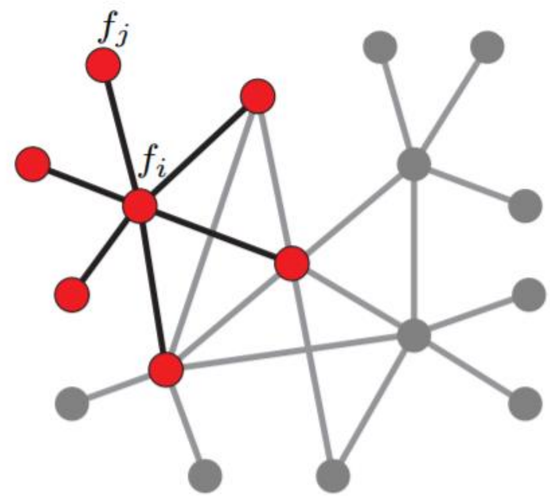
How do we express the Laplacian in a discrete form?

- $\Delta f(x) = -\operatorname{div}(\nabla f)$

  "difference between $f(x)$ and the average of f on an infinitesimal sphere around x"

$$(\Delta f)_i = \frac{1}{d_i} \sum_{j:(i,j)\in \mathrm{E}} a_{ij}(f_i - f_j)$$





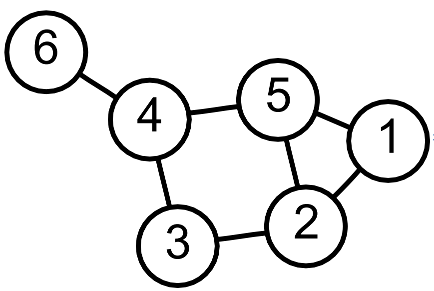
[5]

Graph Deep Learning

Is there another way of expressing this? (Below without the normalization d_i)

$$(\Delta f)_i = \sum_{j:(i,j)\in \mathrm{E}} a_{ij}(f_i - f_j)$$

Yes.



$$\boldsymbol{D} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

$$\boldsymbol{\Delta} = \boldsymbol{D} - \boldsymbol{A}$$

Graph Deep Learning

- $\boldsymbol{\Delta} \in \mathbb{R}^{N \times N}$ is known as the **Laplacian Matrix** of a (sub-)graph consisting of N nodes
- $\boldsymbol{D} \in \mathbb{R}^{N \times N}$ is the **Degree Matrix** and describes the number of edges connected to each node
- $\boldsymbol{A} \in \mathbb{R}^{N \times N}$ is the **Adjacency Matrix** and describes the connectivity of the graph

- For a directed graph $\boldsymbol{\Delta}$ is not s.p.d. thus we normalize $\boldsymbol{\Delta}$ and get $\boldsymbol{\Delta}_{sym}$ s.t.

$$\boldsymbol{\Delta} = \boldsymbol{D} - \boldsymbol{A}$$

$$\boldsymbol{\Delta}_{sym} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{\Delta} \boldsymbol{D}^{-\frac{1}{2}}$$

$$\boldsymbol{\Delta}_{sym} = \boldsymbol{D}^{-\frac{1}{2}} (\boldsymbol{D} - \boldsymbol{A})\, \boldsymbol{D}^{-\frac{1}{2}}$$

$$\boldsymbol{\Delta}_{sym} = \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{D}\, \boldsymbol{D}^{-\frac{1}{2}} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A}\, \boldsymbol{D}^{-\frac{1}{2}}$$

$$\boldsymbol{\Delta}_{sym} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}} \boldsymbol{A}\, \boldsymbol{D}^{-\frac{1}{2}}$$

Graph Deep Learning

Let's do some magic!

- $\boldsymbol{\Delta}_{sym}$ is now s.p.d.

$$\boldsymbol{\Delta}_{sym} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T$$

$$\boldsymbol{U} = [\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1}] \in \mathbb{R}^{N \times N}$$

$$\boldsymbol{\Lambda} = \mathrm{diag}([\lambda_0, \dots, \lambda_{N-1}]) \in \mathbb{R}^{N \times N}$$

- $\boldsymbol{u}_0, \dots, \boldsymbol{u}_{N-1} \in \mathbb{R}^N$: Eigenvectors are known as the graph Fourier modes
- $\lambda_0, \dots, \lambda_{N-1} \in \mathbb{R}$: Eigenvalues are known as the spectral frequencies

- **That means**: We can use $\boldsymbol{U}$ and $\boldsymbol{U}^T$ in order to Fourier transform a graph whereas $\boldsymbol{\Lambda}$ are the spectral filter coefficients!

Graph Deep Learning

Let's do some magic!

- Let $\boldsymbol{x} \in \mathbb{R}^{\mathrm{N}}$ be some signal (a scalar for every node)
- and using the Laplacians eigenvectors we can define its Fourier transform using $\boldsymbol{U}^T$

$$\widehat{\boldsymbol{x}} = \boldsymbol{U}^T \boldsymbol{x}$$

and inverse

$$\boldsymbol{x} = \boldsymbol{U}\widehat{\boldsymbol{x}}$$

- We can therefore describe as convolution with a filter $\boldsymbol{g}$ in spectral domain

$$\boldsymbol{g} \star \boldsymbol{x} = \boldsymbol{U}((\boldsymbol{U}^T \boldsymbol{g}) \cdot (\boldsymbol{U}^T \boldsymbol{x}))$$

- Lets construct a filter $\widehat{\boldsymbol{G}}$ composed by a k-th order polynomial of Laplacians with $\theta_i \in \mathbb{R}$

$$\widehat{\boldsymbol{G}} = \sum_i^k \theta_i \boldsymbol{\Lambda}^i = \theta_k \boldsymbol{\Lambda}^k + \ldots + \theta_1 \boldsymbol{\Lambda}^1 + \theta_o$$

Graph Deep Learning

Let’s do some magic!

- Lets construct a filter $\widehat{\boldsymbol{G}}$ composed by a k-th order polynomial of Laplacians

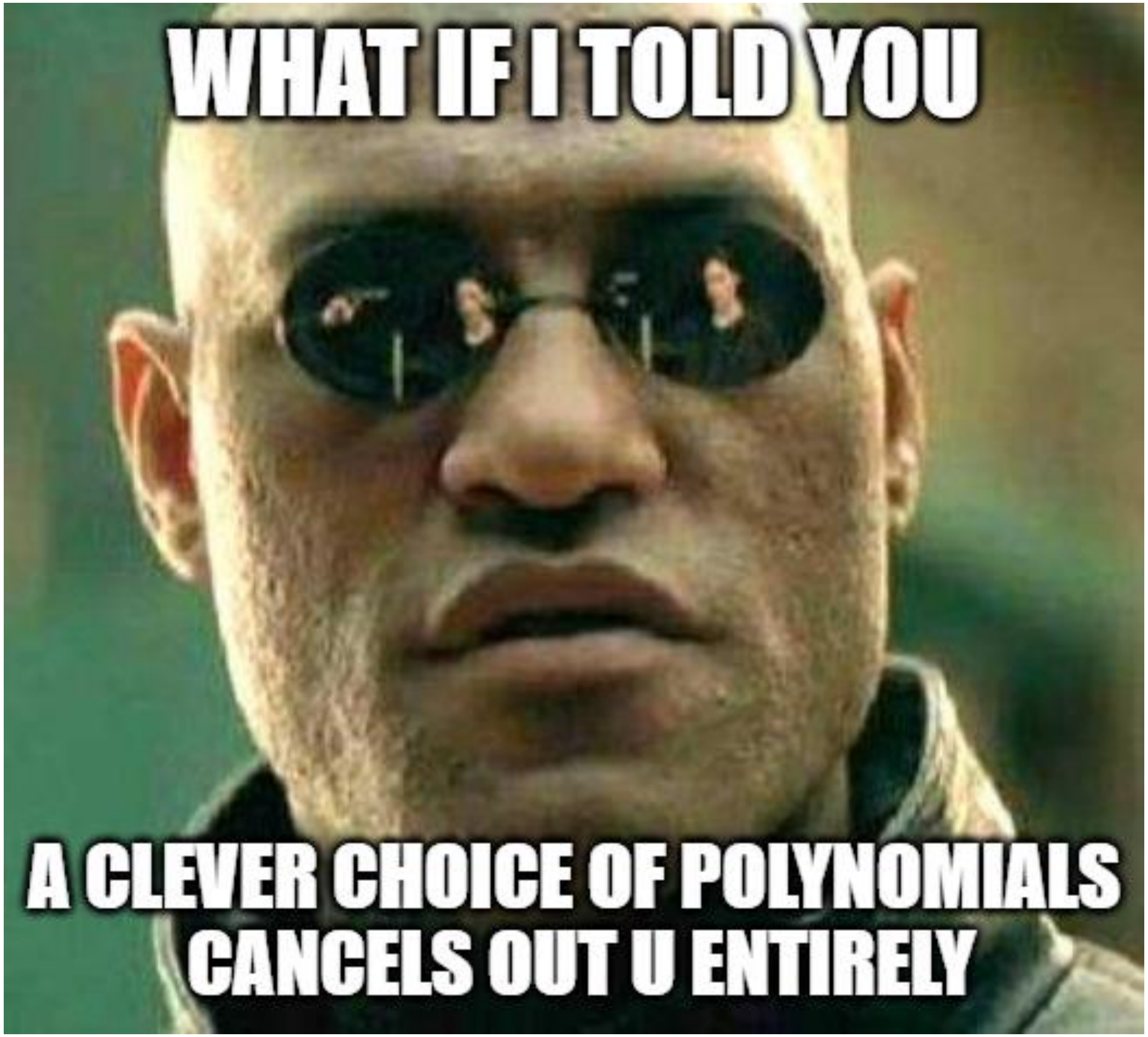
$$\widehat{\boldsymbol{G}} = \sum_{i}^{k} \theta_i \,\boldsymbol{\Lambda}^i$$

- and filter some signal

$$\boldsymbol{U}\widehat{\boldsymbol{G}}\boldsymbol{U}^T\boldsymbol{x} = \boldsymbol{U}\left(\sum_{i}^{k} \theta_i \,\boldsymbol{\Lambda}^i\right)\boldsymbol{U}^T\boldsymbol{x}$$

- **And now what?!**
  - We can convolve now **x** using Laplacian as we adapt θ_i
  - … but $\boldsymbol{U}$ is heavy to compute for every (sub-)graph we want to convolve!

Graph Deep Learning



Graph Deep Learning
Part 2

A. Maier, V. Christlein, K. Breininger, S. Vesal, F. Meister, C. Liu, S. Gündel, S. Jaganathan, N. Maul, M. Vornehm, L. Reeb, F. Thamm, C. Bergler, F. Denzinger, B. Geissler, Z. Yang, A. Popp, M. Nau

Pattern Recognition Lab, Friedrich-Alexander-Universität Erlangen-Nürnberg

Graph Deep Learning

$$\boldsymbol{U}\widehat{\boldsymbol{G}}\boldsymbol{U}^T\boldsymbol{x} = \boldsymbol{U}\left(\sum_i^k \theta_i\, \boldsymbol{\Lambda}^i\right)\boldsymbol{U}^T\boldsymbol{x}$$

- **Remedy**: We choose k and θ such that we get rid of $\boldsymbol{U}$
- Let $k = 1$, $\theta_0 = 2\theta$ and $\theta_1 = -\theta$ we get the following polynomial

$$\begin{aligned}
\boldsymbol{U}\widehat{\boldsymbol{G}}\boldsymbol{U}^T\boldsymbol{x} &= \boldsymbol{U}(2\theta\boldsymbol{\Lambda}^0 - \theta\boldsymbol{\Lambda}^1)\boldsymbol{U}^T\boldsymbol{x} \\
&= (\boldsymbol{U}2\theta\boldsymbol{\Lambda}^0\boldsymbol{U}^T - \boldsymbol{U}\theta\boldsymbol{\Lambda}\boldsymbol{U}^T)\boldsymbol{x} \\
&= (2\theta\boldsymbol{U}\boldsymbol{U}^T - \theta\boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T)\boldsymbol{x} && \boldsymbol{\Delta}_{sym} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^T \\
&= (2\theta\boldsymbol{I} - \theta\boldsymbol{\Delta}_{sym})\boldsymbol{x} \\
&= \theta\big(2\boldsymbol{I} - \boldsymbol{\Delta}_{sym}\big)\boldsymbol{x} && \boldsymbol{\Delta}_{sym} = \boldsymbol{I} - \boldsymbol{D}^{-\frac{1}{2}}\boldsymbol{A}\,\boldsymbol{D}^{-\frac{1}{2}} \\
&= \theta\big(2\boldsymbol{I} - \boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\big)\boldsymbol{x} \\
&= \theta\big(\boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\big)\boldsymbol{x}
\end{aligned}$$

Graph Deep Learning

We can convolve x in spectral domain

Polynomial is $k = 1$, $\theta_0 = 2\theta$ and $\theta_1 = -\theta$ now only depends on θ

$$\boldsymbol{U}\widehat{\boldsymbol{G}}\boldsymbol{U}^T\boldsymbol{x} = \theta\left(\boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\right)\boldsymbol{x}$$

We construct $\widehat{\boldsymbol{G}}$ as a polynomial of Laplacian filters

$$\widehat{\boldsymbol{G}} = \sum_i^k \theta_i \boldsymbol{\Lambda}^i$$

With all restrictions we got rid of the Fourier transform $\boldsymbol{U}^T$

Graph Deep Learning

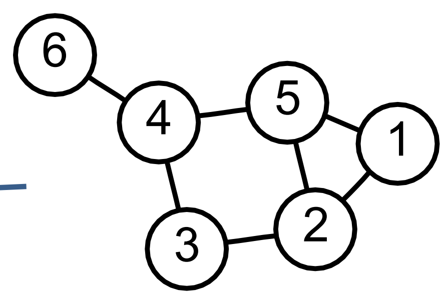
Basic GCN Operation [1]:

$$\boldsymbol{U}\widehat{\boldsymbol{G}}\boldsymbol{U}^T\boldsymbol{x} = \theta\left(\boldsymbol{I} + \boldsymbol{D}^{-1/2}\boldsymbol{A}\boldsymbol{D}^{-1/2}\right)\boldsymbol{x}$$

$$\boldsymbol{D} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 3 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 3 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix}$$

Optimize
θ

$$\boldsymbol{A} = \begin{pmatrix} 0 & 1 & 0 & 0 & 1 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 1 & 1 \\ 1 & 1 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$



[1]: Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

Graph Deep Learning

Question:

Is it *really* necessary to motivate the Graph Convolution from Spectral Domain?

No.

We can motivate spatially as well

Graph Deep Learning

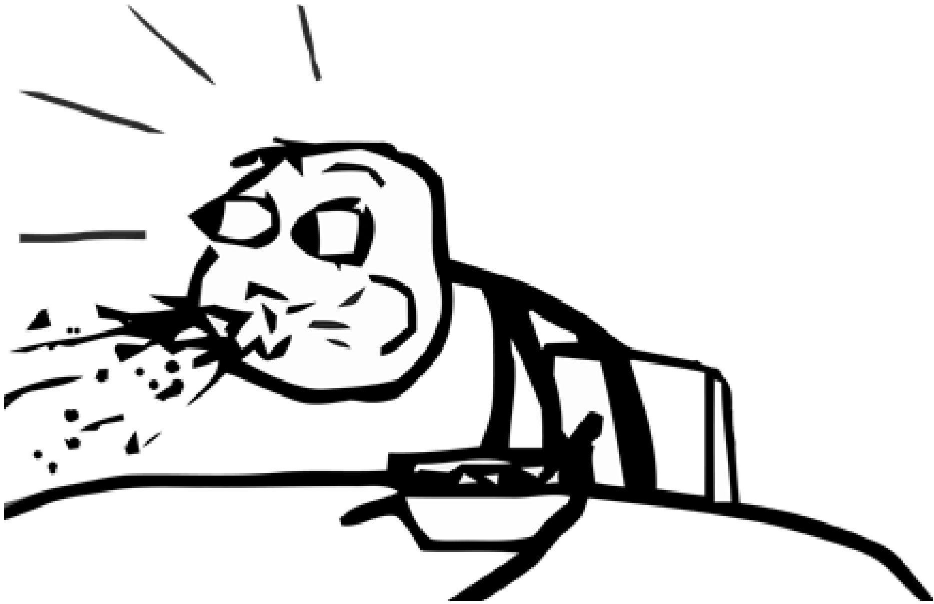
Computer Scientist:

A **Graph** is
a set of nodes (vertices)
connected through edges

We define how to **aggregate** the information of one Vertex through its **Neighbors**

Spatial Graph Convolution

Mathematician:



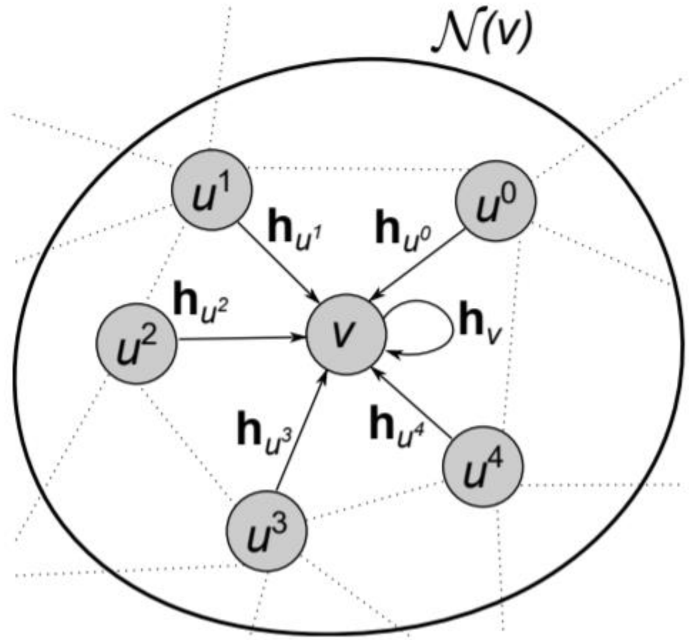
Spectral Graph Convolution

Graph Deep Learning

GraphSAGE [2]

Practically:

- We define a vertex of interest
- We define how neighbors contribute to vertex of interest



[3]

Technically:

- Feature vector at node v in k-th layer: $\boldsymbol{h}_v^k$
  e.g. the 0-th layer may contains the input: $\boldsymbol{h}_v^0 = \boldsymbol{x}_v$
- We aggregate $\boldsymbol{h}_v^k$ over $\boldsymbol{h}_v^{k-1}$ with its neighbors $\boldsymbol{h}_u^{k-1}\ \forall\, u \in N(v)$

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

[3]: Wolterink, Jelmer M., Tim Leiner, and Ivana Išgum. "Graph convolutional networks for coronary artery segmentation in cardiac CT angiography." *International Workshop on Graph Learning in Medical Imaging*. Springer, Cham, 2019.

Graph Deep Learning

GraphSAGE [2] - The Algorithm

Algorithm 1: GraphSAGE embedding generation (i.e., forward propagation) algorithm

Input : Graph $\mathcal{G}(\mathcal{V}, \mathcal{E})$; input features $\{\mathbf{x}_v, \forall v \in \mathcal{V}\}$; depth K; weight matrices $\mathbf{W}^k, \forall k \in \{1, ..., K\}$; non-linearity σ; differentiable aggregator functions $\text{AGGREGATE}_k, \forall k \in \{1, ..., K\}$; neighborhood function $\mathcal{N} : v \rightarrow 2^{\mathcal{V}}$

Output : Vector representations $\mathbf{z}_v$ for all $v \in \mathcal{V}$

$\mathbf{h}_v^0 \leftarrow \mathbf{x}_v, \forall v \in \mathcal{V}$;
for $k = 1...K$ **do**
    for $v \in \mathcal{V}$ **do**
        $\mathbf{h}_{\mathcal{N}(v)}^k \leftarrow \text{AGGREGATE}_k(\{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$;
        $\mathbf{h}_v^k \leftarrow \sigma\left(\mathbf{W}^k \cdot \text{CONCAT}(\mathbf{h}_v^{k-1}, \mathbf{h}_{\mathcal{N}(v)}^k)\right)$
    end
    $\mathbf{h}_v^k \leftarrow \mathbf{h}_v^k / \|\mathbf{h}_v^k\|_2, \forall v \in \mathcal{V}$
end
$\mathbf{z}_v \leftarrow \mathbf{h}_v^K, \forall v \in \mathcal{V}$

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

Graph Deep Learning

GraphSAGE [2] - Aggregators

- Mean Aggregator:

$$\mathbf{h}_v^k \leftarrow \sigma(\mathbf{W} \cdot \text{MEAN}(\{\mathbf{h}_v^{k-1}\} \cup \{\mathbf{h}_u^{k-1}, \forall u \in \mathcal{N}(v)\})$$

- GCN Aggregator
- Pooling Aggregator

$$\text{AGGREGATE}_k^{\text{pool}} = \max(\{\sigma\left(\mathbf{W}_{\text{pool}}\mathbf{h}_{u_i}^k + \mathbf{b}\right), \forall u_i \in \mathcal{N}(v)\}),$$

- LSTM Aggregator

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

Graph Deep Learning [4]

GNN MPNN MoNet GraphSAGE PATCHY-SAN GAT

DGCNN FastGCN SumPooling AvgPooling

GGNN DGI CGMM GraphESN SortPooling

SSE DiffPool GAAN Huang et al. PGC-DGCNN

SpectralCNN NN4G AGCN GeniePath

ChebNet DualGCN MaxPooling

Henaff et al. ClusterGCN FastGCN

CayleyNet GlobalAttentionPooling

LGCN GCN StoGCN

DCNN Set2Set SortPooling

[4]: Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *arXiv preprint arXiv:1901.00596* (2019).

References

[1]: Kipf, Thomas N., and Max Welling. "Semi-supervised classification with graph convolutional networks." *arXiv preprint arXiv:1609.02907* (2016).

[2]: Hamilton, Will, Zhitao Ying, and Jure Leskovec. "Inductive representation learning on large graphs." *Advances in neural information processing systems*. 2017.

[3]: Wolterink, Jelmer M., Tim Leiner, and Ivana Išgum. "Graph convolutional networks for coronary artery segmentation in cardiac CT angiography." *International Workshop on Graph Learning in Medical Imaging*. Springer, Cham, 2019.

[4]: Wu, Zonghan, et al. "A comprehensive survey on graph neural networks." *arXiv preprint arXiv:1901.00596* (2019).

[5]: Bronstein, Michael et al. Lecture “Geometric deep learning on graphs and manifolds” held at SIAM Tutorial Portlan (2018)

Image References

[a] https://de.serlo.org/mathe/funktionen/funktionsbegriff/funktionen-graphen/graph-funktion

[b] https://www.nwrfc.noaa.gov/snow/plot_SWE.php?id=AFSW1

[c] https://tennisbeiolympia.wordpress.com/meilensteine/steffi-graf/

[d] https://www.pinterest.de/pin/624381935818627852/

[e] https://www.uihere.com/free-cliparts/the-pentagon-pentagram-symbol-regular-polygon-golden-five-pointed-star-2282605

[f] http://geometricdeeplearning.com/ (Geometric Deep Learning on Graphs and Manifolds)

[g] https://i.stack.imgur.com/NU7y2.png

[h] https://de.wikipedia.org/wiki/Datei:Convolution_Animation_(Gaussian).gif

[i]https://www.researchgate.net/publication/306293638/figure/fig1/AS:396934507450372@1471647969381/Example-of-centerline- extracted-left-and-coronary-artery-tree-mesh-reconstruction.png

[j] https://www.eurorad.org/sites/default/files/styles/figure_image_teaser_large/public/figure_image/2018-08/0000015888/000006.jpg?itok=hwX1sbCO