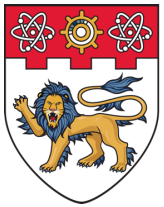
Convolutional Neural Networks on Graphs

Xavier Bresson

School of Computer Science and Engineering
Nanyang Technological University (NTU)



NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

NATIONAL RESEARCH FOUNDATION
PRIME MINISTER'S OFFICE
SINGAPORE

Joint work with M. Defferrard (EPFL), P. Vandergheynst (EPFL),
M. Bronstein (USI), F. Monti (USI), R. Levie (TAU)

Isaac Newton Institute
Generative Models, Parameter Learning and Sparsity
Nov 1st 2017

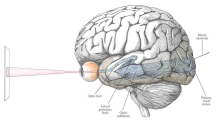
Outline

- **Deep Learning**
  - *A brief history*
  - *High-dimensional learning*
- **Convolutional Neural Networks**
  - *Architecture*
  - *Non-Euclidean Data*
- **Spectral Graph Theory**
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- **Spectral ConvNets**
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- **Spectral ConvNets on Multiple Graphs**
  - *Recommender Systems* [NIPS'17]*
- **Conclusion**

Outline

- **Deep Learning**
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

A brief history of artificial neural networks



Visual primary cortex
Hubel-Wiesel
1959

1958 Perceptron
Rosenblatt

Birth of
Data Science
Split from Statistics
Tukey

1962

1982 Backprop
Werbos

Neocognitron
Fukushima

1987 First
NIPS

1995 SVM/Kernel techniques
Vapnik

1997 RNN/LSTM
Schmidhuber

1998 CNN
LeCun

1999 First NVIDIA
GPU

Hardware
GPU speed 2x/1.5 year

Big Data
Volume 2x/1.5 year

2006 First Amazon
Cloud Center

2010 Kaggle
Platform

2012
Deep Learning
Breakthough
Hinton

2014 GAN, Goodfellow
ResNet, He
Deep RL, Mnih

2015 Google AI
TensorFlow
Facebook AI
Torch

Facebook Center
OpenAI Center

Data scientist
1st Job in US

4th industrial revolution?
Digital intelligence
revolution
or new AI bubble?

AI Birth

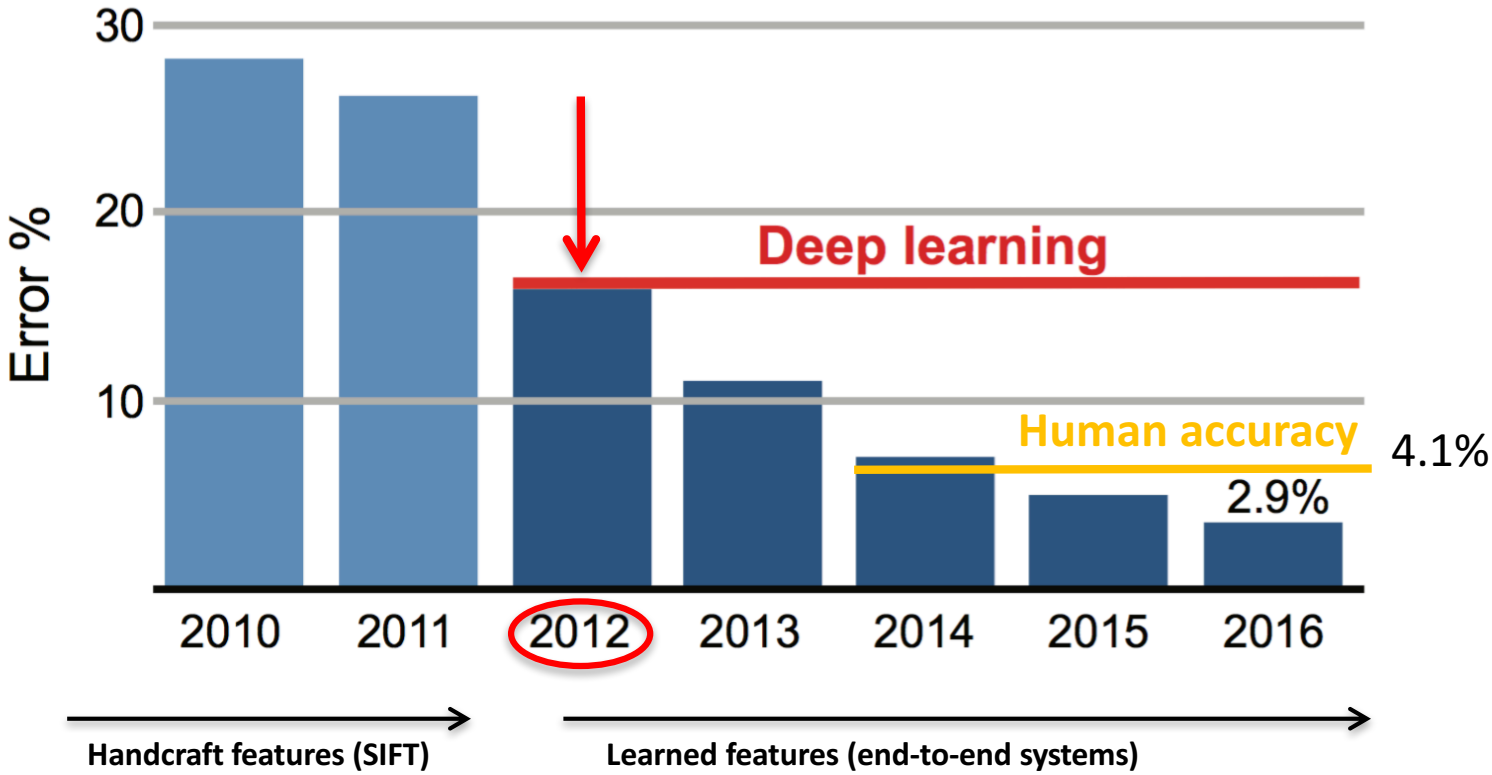
AI Winter [1960's-2012]

AI Renaissance

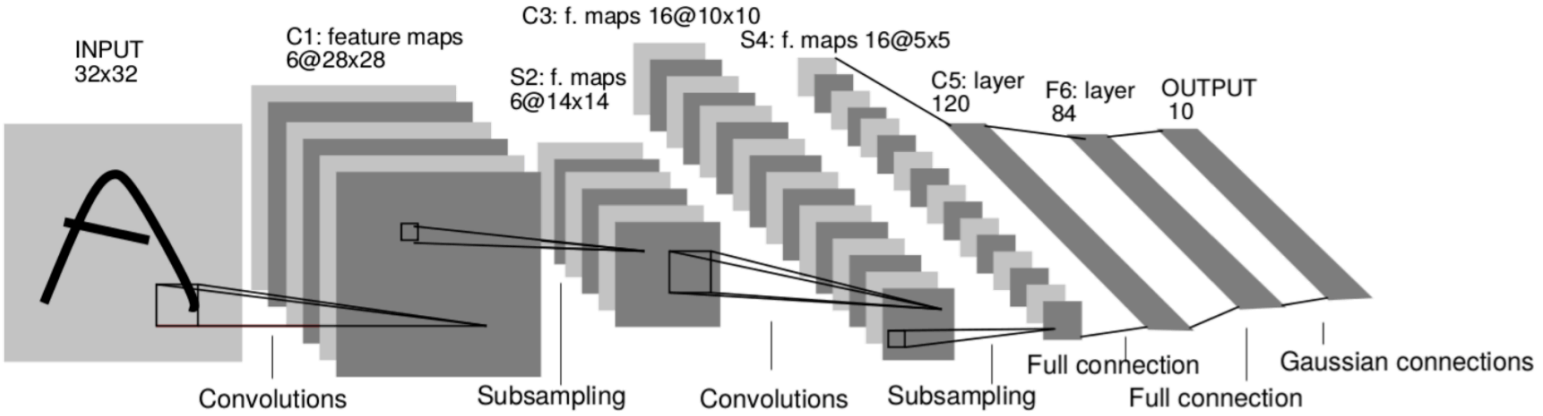
Kernel techniques
Graphical models
Wavelets, numerical PDEs
Handcrafted Features

Breakthrough in image recognition

IMAGENET



Convolutional neural networks: LeNet-5[1]

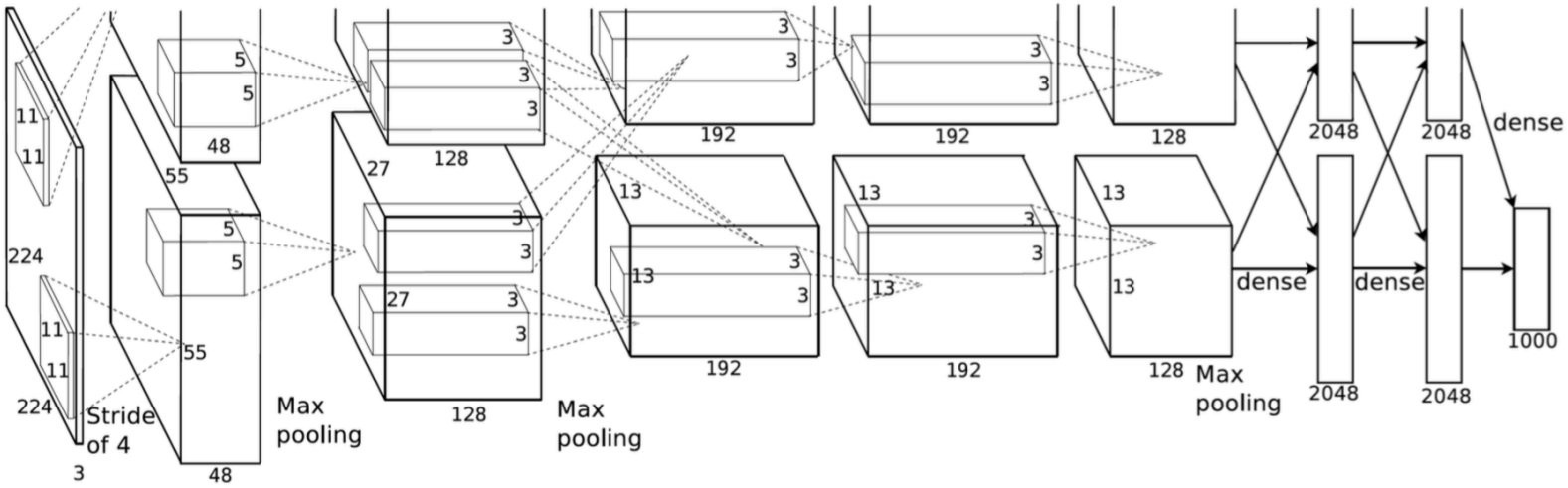


- 2 convolutional + 2 fully connected layers
- tanh non-linearity
- 1M parameters
- Training set: MNIST 70K images
- Trained on CPU

1998

[1] LeCun et al. 1998

Convolutional neural networks: AlexNet[2]



- 5 convolutional + 3 fully connected layers
- ReLU non-linearity
- Dropout regularization
- 60M parameters
- Trained set: ImageNet 1.5M images
- Trained on GPU

2012

More learning capacity

More data

More computational power

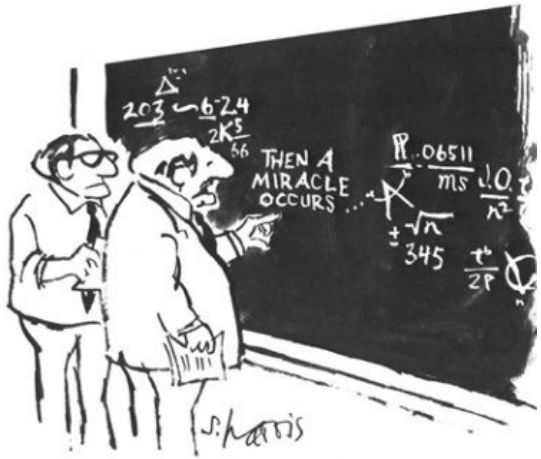
[2] Krizhevsky, Sutskever, Hinton 2012

Outline

- **Deep Learning**
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

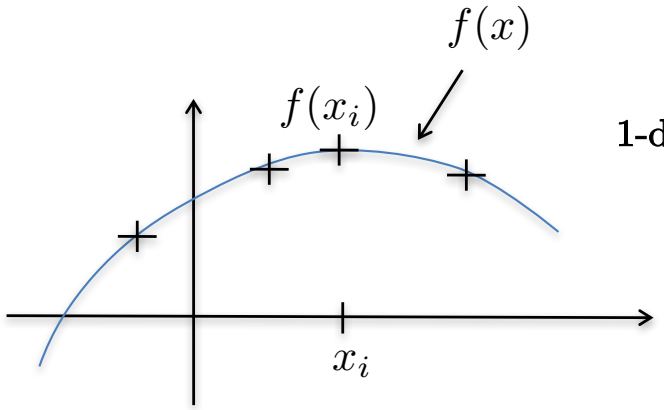
High-dimensional learning

- NNs are powerful to solve high-dimensional learning problems.



"I think you should be more explicit here in step two."

- Prototypal learning: Evaluate function $f(x)$ given n samples $\{x_i, f(x_i)\}$. How? Suppose f is regular, use interpolation of close points.

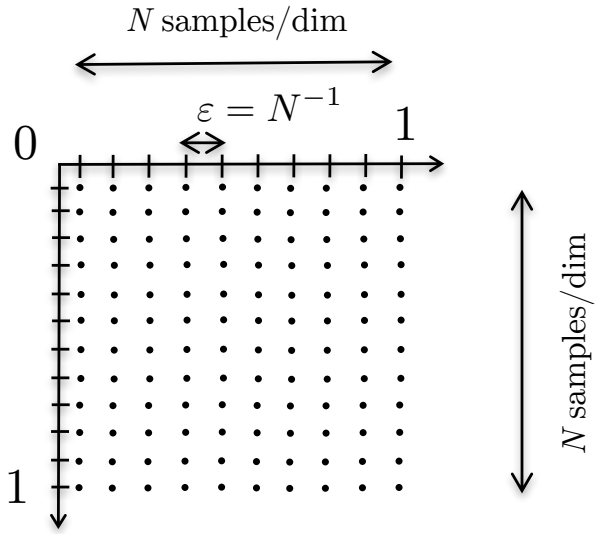


1-dimensional learning

Curse of dimensionality

- What about in high dimensions? No interpolation possible because we can never have enough close points to make a good interpolation.

- How many points to cover the 2D plane? ε^{-2}
  $\Rightarrow$ d-dim cube $[0,1]^d$ requires ε^{-d} points.



- dim(image) = 512 x 512 ≈ 10^6
  For N=10 samples/dim $\Rightarrow$ $10^{1,000,000}$ points!

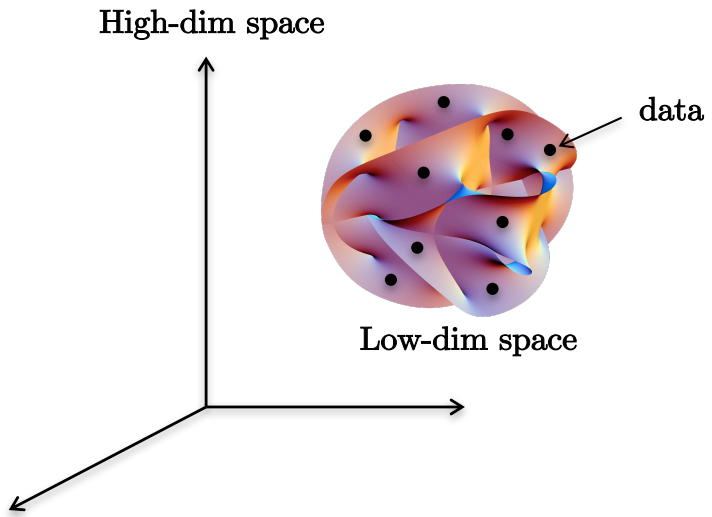
- Curse of dim:
  - ☹ The number of x of f is exponential.
  - ☹ All samples are far from each other in high-dim[3].
  - ☹ Optimization: Also need ε^{-d} points to find solution.

$$\min_f \; L(f) = \mathbb{E}[\ell(f(x_i), y_i)]$$

[3] Beyer 1998

Blessing of structure

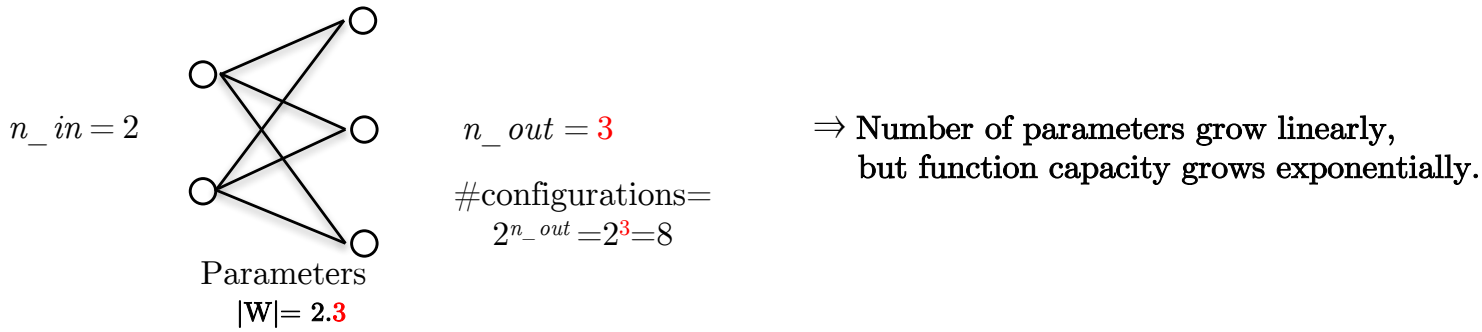
- In our universe:
  - ☺ Data have structures, they concentrate in (non-)smooth low-dim spaces.



  - ☺ These structures can be expressed as mathematical properties (statistics, geometry, groups, etc).

Learning exponential functions

- Neural networks can learn parametric exponential functions with linear number of parameters.



- A very large number of parameters can be learned by backpropagation, order of billions.

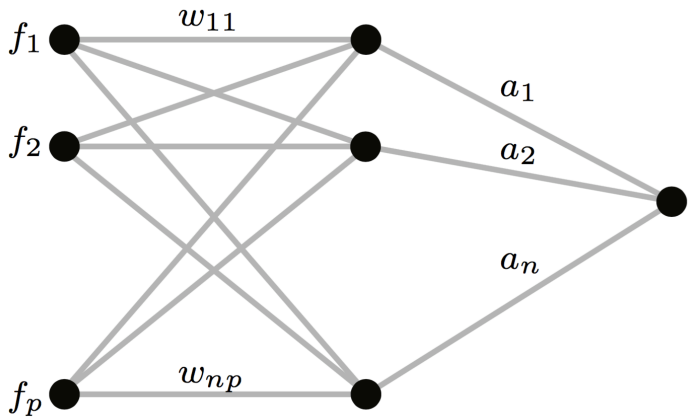
  ⇒ Huge capacity to learn complex data[4].

[4] Zhang, Bengio, Hardt, Recht, Vinyals, 2016

Universal function approximation[4]

Universal Approximation Theorem Let ξ be a non-constant, bounded, and monotonically-increasing continuous activation function, $y : [0,1]^p \to \mathbb{R}$ continuous function, and $\epsilon > 0$. Then, $\exists n$ and parameters $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$, $\mathbf{W} \in \mathbb{R}^{n \times p}$ s.t.

$$\left| \sum_{i=1}^{n} a_i \xi(\mathbf{w}_i^\top \mathbf{f} + b_i) - y(\mathbf{f}) \right| < \epsilon \qquad \forall \mathbf{f} \in [0,1]^p$$



[5] Cybenko 1989, Hornik 1991

Shallow neural nets as universal approximators?

☺ Any continuous function can be approximated arbitrarily well by a neural network with a single hidden layer. Then,

- How many neurons?
- How long is the optimization?
- Does it generalize well/overfit?

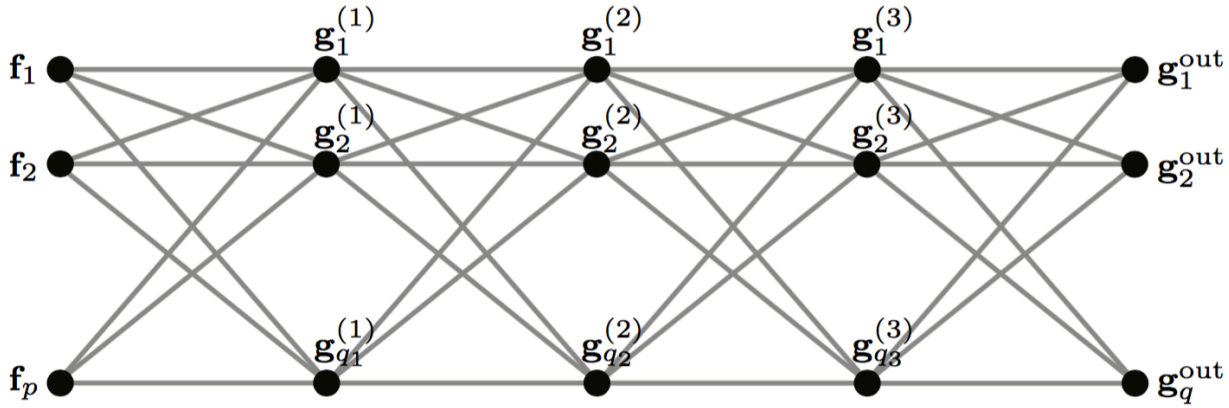
☹ Shallow NNs: Data (s.a. image) requires NNs with an exponential number of neurons for a single layer, optimizes badly, and does not generalize.

☺ Deep NNs: Multiple layers and data structure can alleviate the above issues[6].

[6] Montufar, Pascanu, Cho, Bengio 2014

Fully connected networks

$$\begin{aligned} \mathbf{f}_l &= l\text{-th image feature (R,G,B channels)}, \dim(\mathbf{f}_l) = n \times 1 \\ \mathbf{g}_l^{(k)} &= l\text{-th feature map}, \dim(\mathbf{g}_l^{(k)}) = n_l^{(k)} \times 1 \end{aligned}$$



Deep neural network consists of multiple layers.

Linear layer $$\mathbf{g}_l^{(k)} = \xi\left(\sum_{l'=1}^{q_{k-1}} \mathbf{W}_{l,l'}^{(k)} \xi\left(\sum_{l'=1}^{q_{k-2}} \mathbf{W}_{l,l'}^{(k-1)} \xi\left(\cdots \mathbf{f}_{l'}\right)\right)\right)$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Fully connected networks

- FC networks capture non-linear factorizations of data.
- They overfit: Too many parameters to learn.
- NN architectures require additional data structures to successfully analyze image, sound, document, etc.

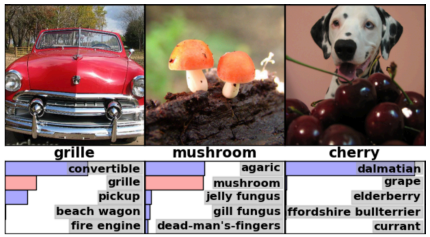
Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- **Convolutional Neural Networks**
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Convolutional neural networks[1]

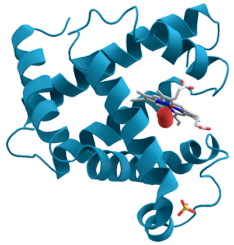
- Data (image, video, sound) are compositional, i.e. they are formed of patterns that are:
  - Local
  - Stationary
  - Multi-scale (/hierarchical)

- ConvNets leverage the compositionality structure: They extract compositional features and feed them to classifier, recommender, etc (end-to-end).

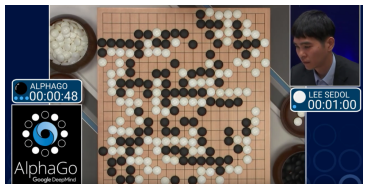


Computer Vision

NLP



Drug discovery

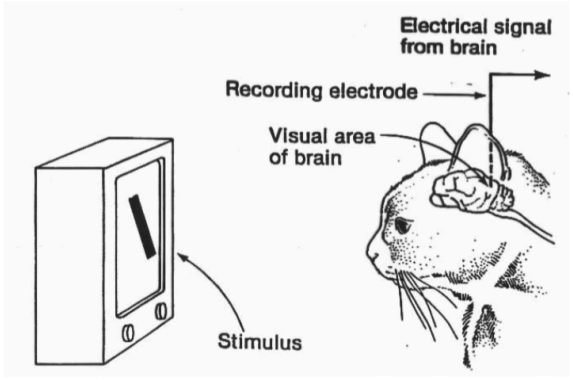


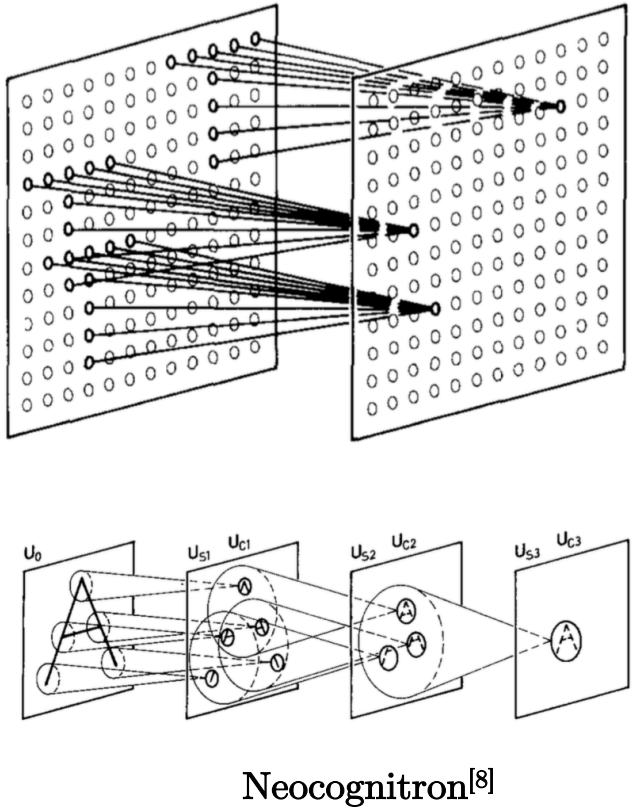
Games

[1] LeCun et al. 1998

Key properties

- Locality: Property inspired by visual cortex neurons[7].
- Local receptive fields activate in the presence of (learned) local features.



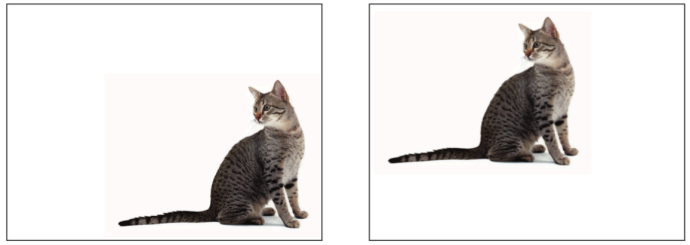


Neocognitron[8]

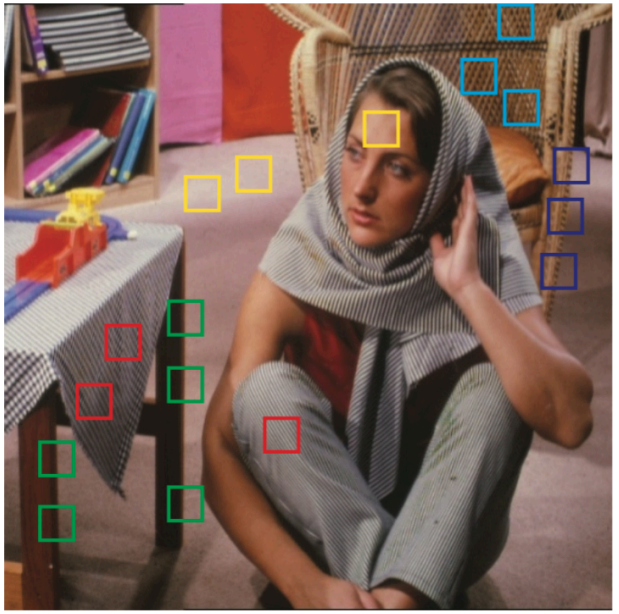
[7] Cybenko 1989, Hornik 1991, [8] Fukushima 1980

Key properties

- Stationarity $\Leftrightarrow$ Translation invariance



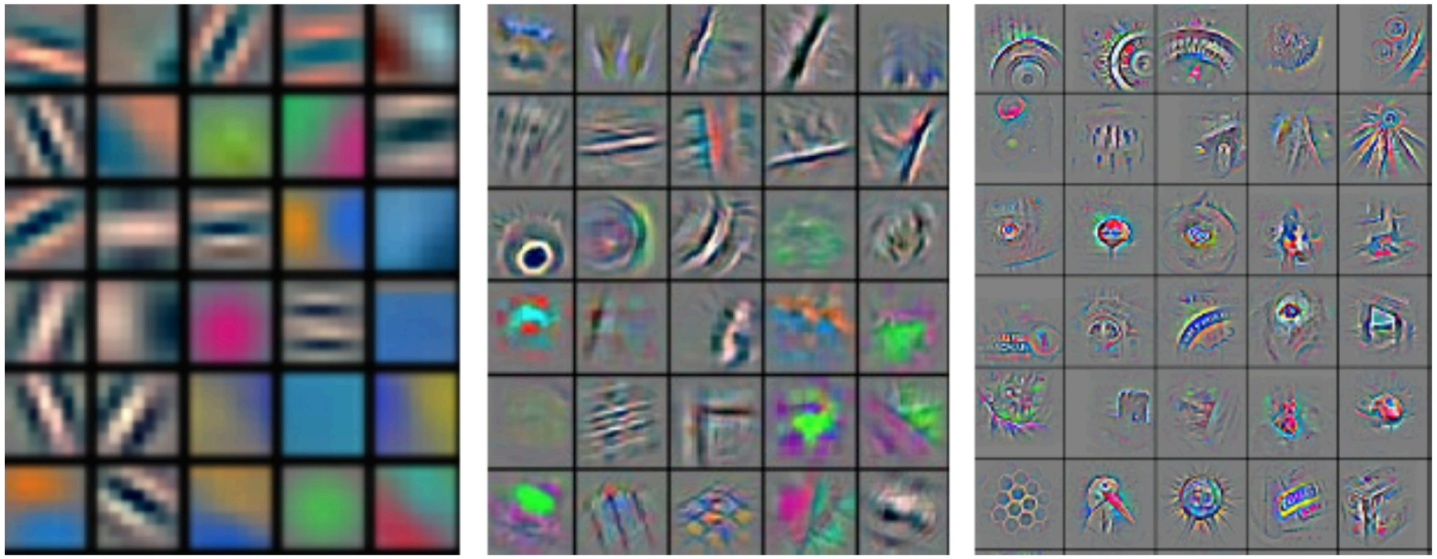
- Local stationarity $\Leftrightarrow$ Similar patches shared across the data domain.



Key properties

- Multi-scale: Simple structures combine to compose slightly more abstract structures, and so on, in a hierarchical way.

- Also inspired by brain visual primary cortex (V1 and V2 neurons).

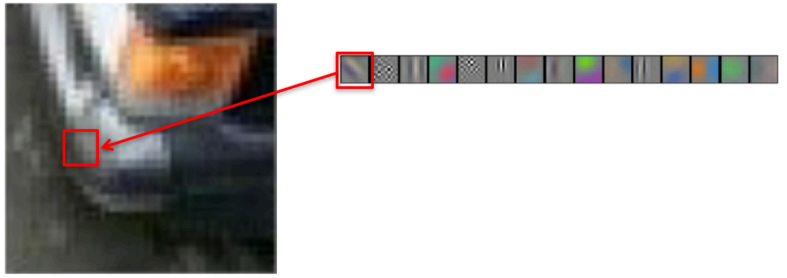


Features learned by a ConvNet become increasingly complex at deeper layers.[9]

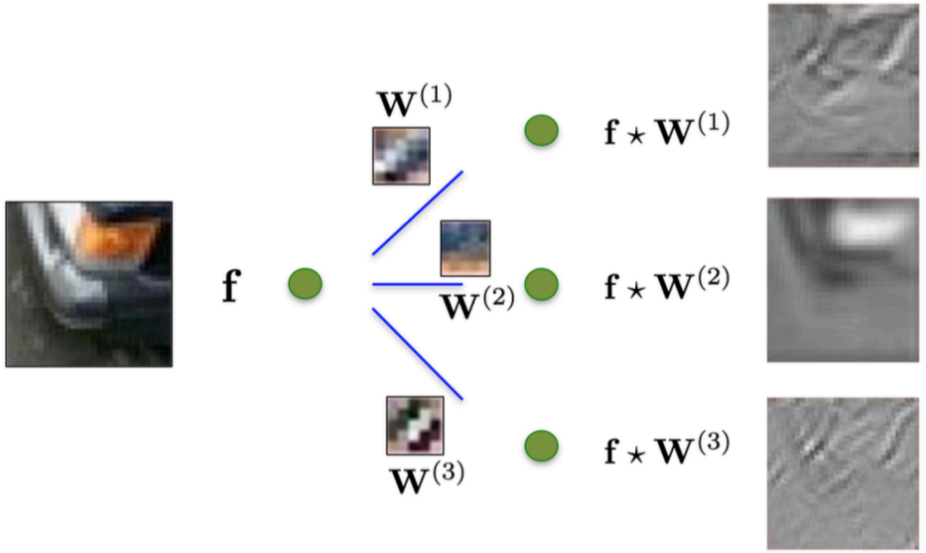
[9] Zeiler, Fergus 2013

Implementation

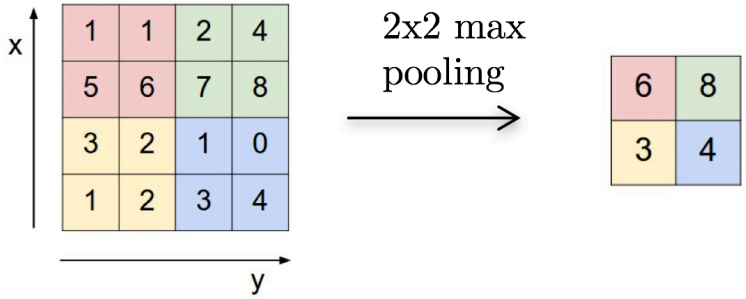
- Locality: compact support kernels
  $\Rightarrow$ O(1) parameters per filter.



- Stationarity: convolutional operators
  $\Rightarrow$ O($n$log$n$) in general (FFT) and
  O(n) for compact kernels.



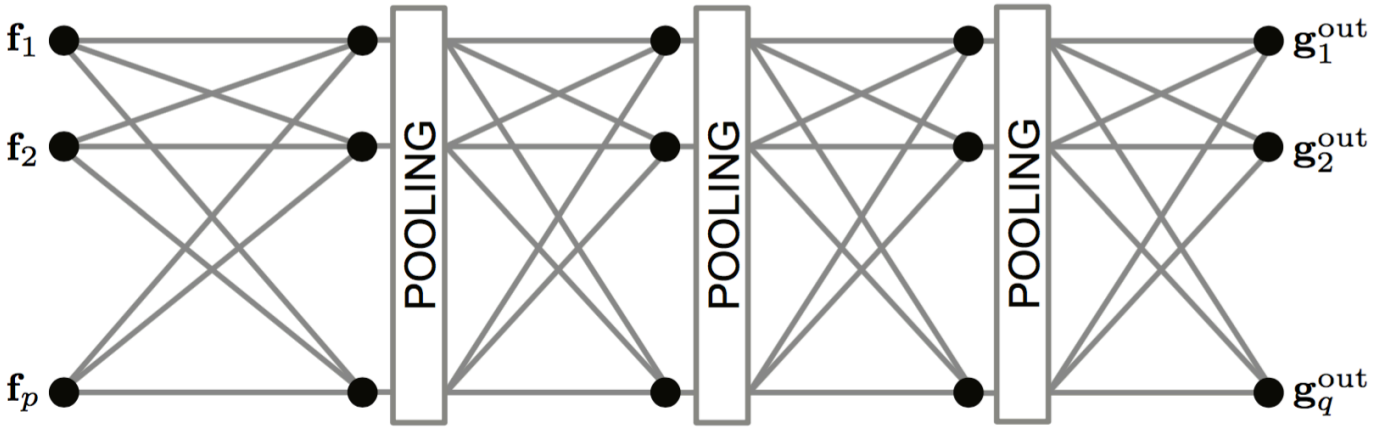
- Multi-scale: downsampling +
  pooling $\Rightarrow$ O(n)



Compositional features

$\mathbf{f}_l$ = l-th image feature (R,G,B channels), $\dim(\mathbf{f}_l) = n \times 1$

$\mathbf{g}_l^{(k)}$ = l-th feature map, $\dim(\mathbf{g}_l^{(k)}) = n_l^{(k)} \times 1$



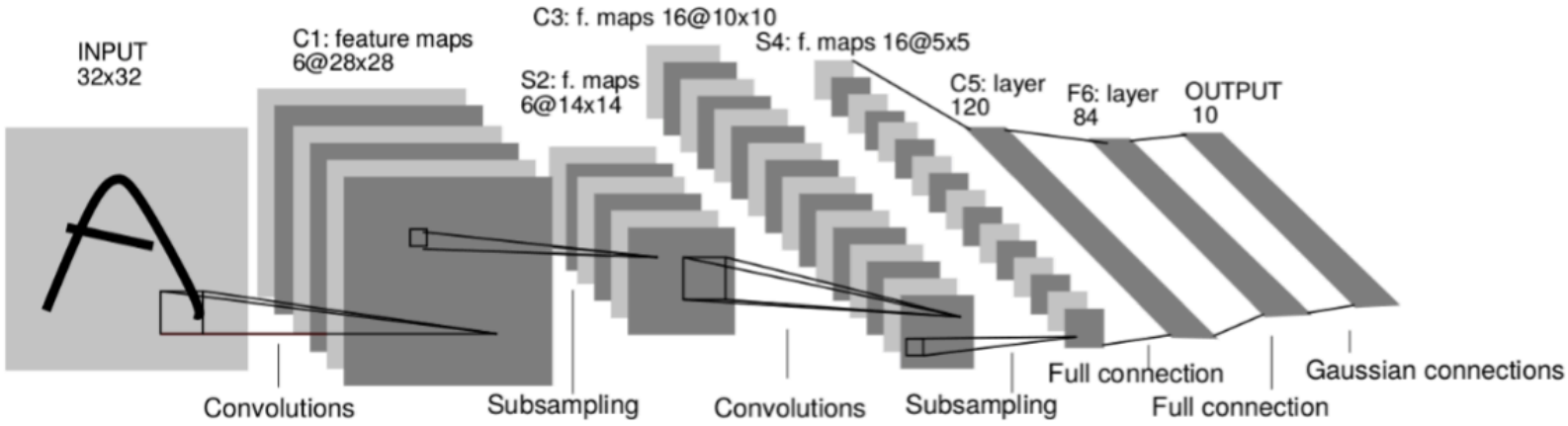
Compositional features consist of multiple convolutional + pooling layers.

Linear layer $$\mathbf{g}_l^{(k)} = \xi\left(\sum_{l'=1}^{q_{k-1}} \mathbf{W}_{l,l'}^{(k)} \star \xi\left(\sum_{l'=1}^{q_{k-2}} \mathbf{W}_{l,l'}^{(k-1)} \star \xi\left(\cdots \mathbf{f}_{l'}\right)\right)\right)$$

Activation, e.g. $\xi(x) = \max\{x, 0\}$ rectified linear unit (ReLU)

Pooling $$\mathbf{g}_l^{(k)}(x) = \|\mathbf{g}_l^{(k-1)}(x') : x' \in \mathcal{N}(x)\|_p \quad p = 1, 2, \text{ or } \infty$$

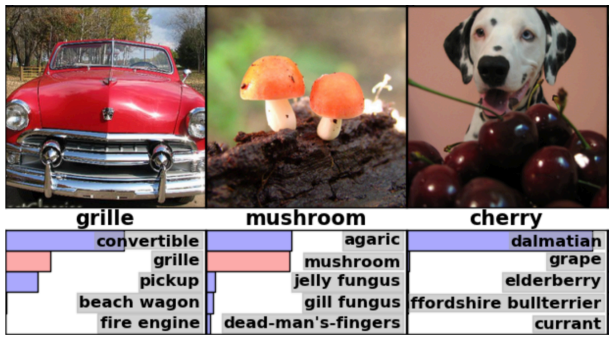
ConvNets[1]



- ☺ Filters localized in space (Locality)
- ☺ Convolutional filters (Stationarity)
- ☺ Multiple layers (Multi-scale)
- ☺ O(1) parameters per filter (independent of input image size n)
- ☺ O(n) complexity per layer (filtering done in the spatial domain)

[1] LeCun et al. 1998

ConvNets in computer vision



Image/object recognition

[Krizhevsky-Sutskever-Hinton'12]

Image inpainting

[Pathak-Efros-etal'16]

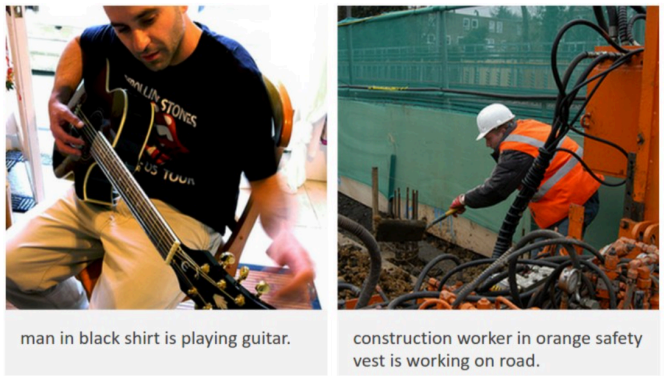


Image captioning

[Karpathy-Li'15]

Self-driving cars

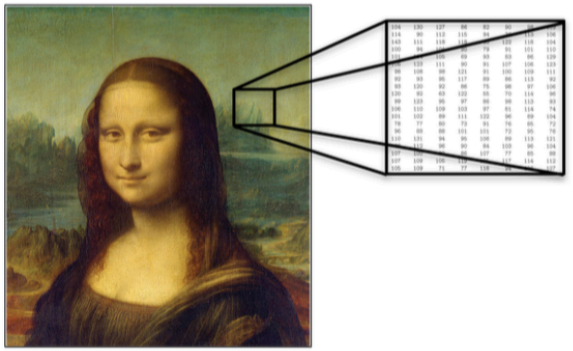
⇒ Highly successful - ConvNets is the Swiss knife of Computer Vision!

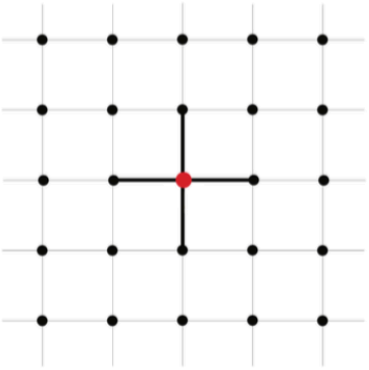
Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- **Convolutional Neural Networks**
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Data domain for ConvNets

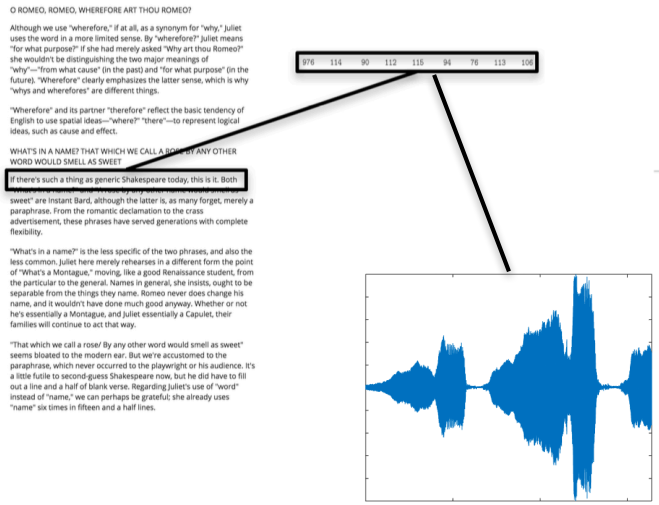
- Image, volume, video: 2D, 3D, 2D+1 Euclidean domains





2D grids

- Sentence, word, sound: 1D Euclidean domain

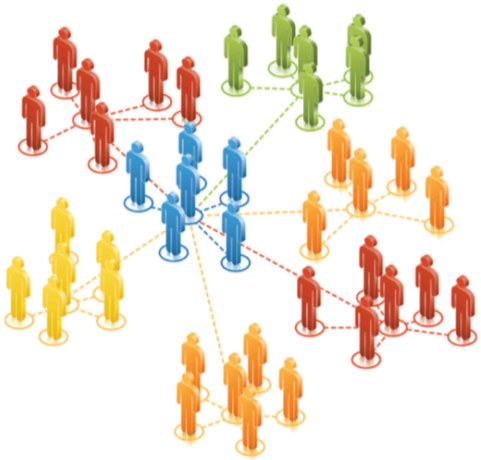


1D grid

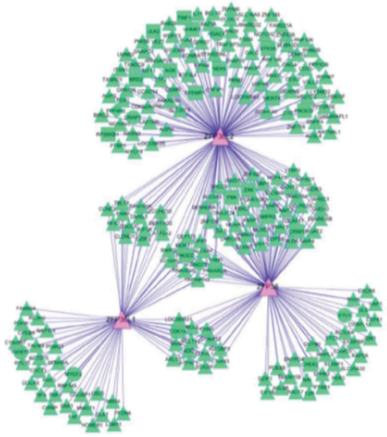
- These domains have nice regular spatial structures.

  $\Rightarrow$ All CNN operations are math well defined and fast (convolution, pooling).

Non-Euclidean data

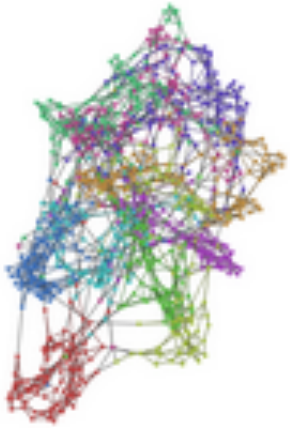


Social networks

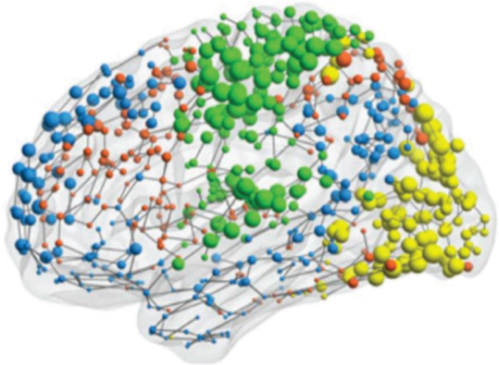


Regulatory networks

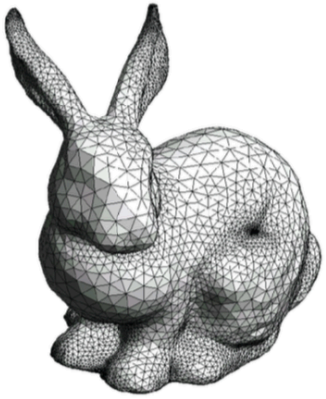
=



Graphs/
Networks



Functional networks



3D shapes

- Also NLP, physics, social science, communication networks, etc.

Challenges of graph deep learning

- Extend neural network techniques to graph-structured data.
- Assumption: Non-Euclidean data are locally stationary and manifest hierarchical structures.
- How to define compositionality on graphs? (convolution and pooling on graphs)
- How to make them fast? (linear complexity)

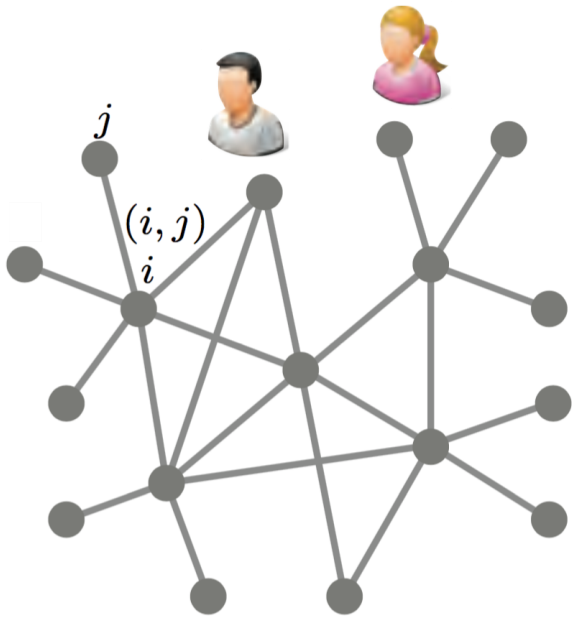
Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- **Spectral Graph Theory**
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Graphs[10]

- Graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
- Vertices $\mathcal{V} = \{1, \ldots, n\}$
- Edges $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$
- Vertex weights $b_i > 0$ for $i \in \mathcal{V}$
- Edge weights $a_{ij} \geq 0$ for $(i, j) \in \mathcal{E}$
- Vertex fields $L^2(\mathcal{V}) = \{f : \mathcal{V} \to \mathbb{R}^h\}$
  Represented as $\mathbf{f} = (f_1, \ldots, f_n)$
- Hilbert space with inner product

$$\langle f, g \rangle_{L^2(\mathcal{V})} = \sum_{i \in \mathcal{V}} a_i f_i g_i$$

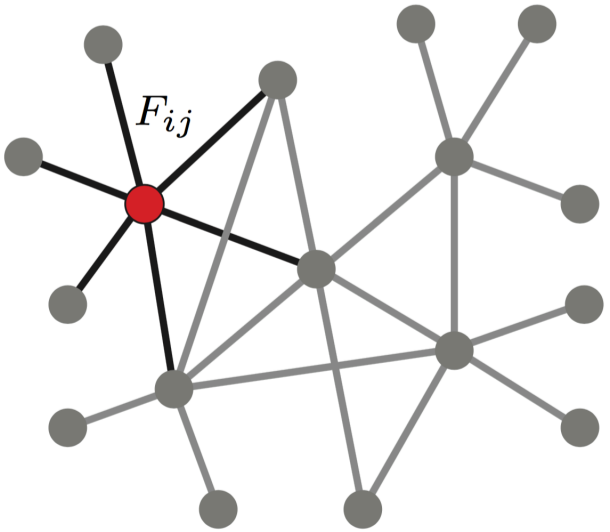


[10] Chung 1994

Calculus on graphs

- **Gradient** operator $\nabla : L^2(\mathcal{V}) \to L^2(\mathcal{E})$

$$(\nabla f)_{ij} = \sqrt{a_{ij}}(f_i - f_j)$$



- **Divergence** operator $\mathrm{div} : L^2(\mathcal{E}) \to L^2(\mathcal{V})$

$$(\mathrm{div}F)_i = \frac{1}{b_i} \sum_{j:(i,j)\in\mathcal{E}} \sqrt{a_{ij}}(F_{ij} - F_{ji})$$

adjoint to the gradient operator

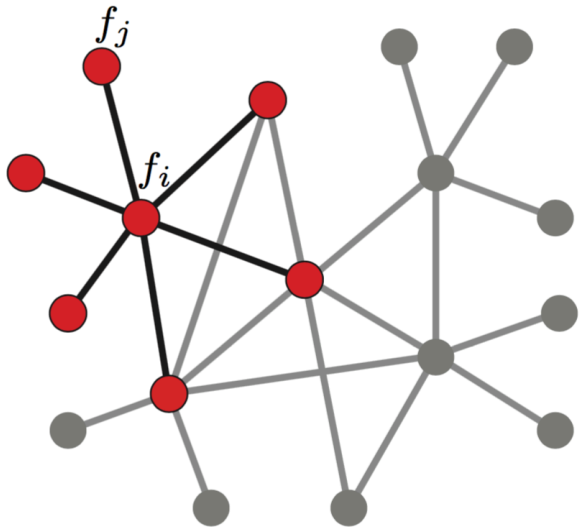
$$\langle F, \nabla f \rangle_{L^2(\mathcal{E})} = \langle \nabla^* F, f \rangle_{L^2(\mathcal{V})} = \langle -\mathrm{div}F, f \rangle_{L^2(\mathcal{V})}$$

Graph Laplacian

- Laplacian operator $\Delta : L^2(\mathcal{V}) \to L^2(\mathcal{V})$

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

difference between f and its local average (2^{nd} derivative on graphs)



- Core operator in spectral graph theory.
- Represented as a positive semi-definite $n \times n$ matrix
  - Unnormalized Laplacian $\mathbf{\Delta} = \mathbf{D} - \mathbf{A}$
  - Normalized Laplacian $\mathbf{\Delta} = \mathbf{I} - \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{-1/2}$
  - Random walk Laplacian $\mathbf{\Delta} = \mathbf{I} - \mathbf{D}^{-1}\mathbf{A}$

where $\mathbf{A} = (a_{ij})$ and $\mathbf{D} = \text{diag}(\sum_{j\neq i} a_{ij})$

Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- **Spectral Graph Theory**
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Spectral decomposition

- A Laplacian of a graph of n vertices admits n eigenvectors

$$\boldsymbol{\Delta}\boldsymbol{\phi}_k = \lambda_k \boldsymbol{\phi}_k, \qquad k = 1, 2, \ldots$$

  - Eigenvectors are real and orthonormal $\langle \phi_k, \phi_{k'} \rangle_{L^2(\mathcal{V})} = \delta_{kk'}$
    (due to self-adjointness)
  - Eigenvalues are non-negative $0 = \lambda_1 \leq \lambda_2 \leq \ldots \leq \lambda_n$
    (due to positive-semidefiniteness)

- Eigendecomposition of a graph Laplacian

$$\boldsymbol{\Delta} = \boldsymbol{\Phi}^T \boldsymbol{\Lambda} \boldsymbol{\Phi}$$

$$\text{where } \boldsymbol{\Phi} = (\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_n) \text{ and } \boldsymbol{\Lambda} = \text{diag}(\lambda_1, \ldots, \lambda_n)$$

Interpretation

- Find the smoothest orthonormal basis $\mathbf{\Phi} = (\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_n)$ on a graph

$$\min_{\boldsymbol{\phi}_k} E_{\text{Dir}}(\boldsymbol{\phi}_k) \quad \text{s.t.} \quad \|\boldsymbol{\phi}_k\| = 1, \quad k = 2, 3, \dots n$$
$$\boldsymbol{\phi}_k \perp \text{span}\{\boldsymbol{\phi}_1, \dots, \boldsymbol{\phi}_{k-1}\}$$

where E_{Dir} is the Dirichlet energy = measure of smoothness of a function

$$E_{\text{Dir}}(\mathbf{f}) = \mathbf{f}^\top \mathbf{\Delta} \mathbf{f}$$

- Solution: first n Laplacian eigenvectors

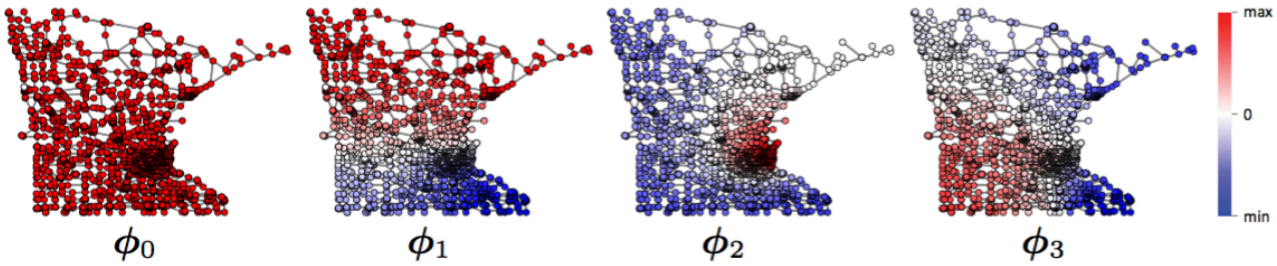
$$\min_{\mathbf{\Phi} \in \mathbb{R}^{n \times n}} \underbrace{\text{trace}(\mathbf{\Phi}^\top \mathbf{\Delta} \mathbf{\Phi})}_{\|\mathbf{\Phi}\|_{\mathcal{G}} \text{ Dirichlet norm}} \quad \text{s.t.} \quad \mathbf{\Phi}^\top \mathbf{\Phi} = \mathbf{I}$$

Laplacian eigenvectors

- Euclidean domain:

First eigenvectors of 1D Euclidean Laplacian = standard Fourier basis

- Graph domain:



First Laplacian eigenvectors of a graph

Lap eigenvectors related to graph geometry
(s.a. communities, hubs, etc), spectral clustering[10]

[10] Von Luxburg 2007

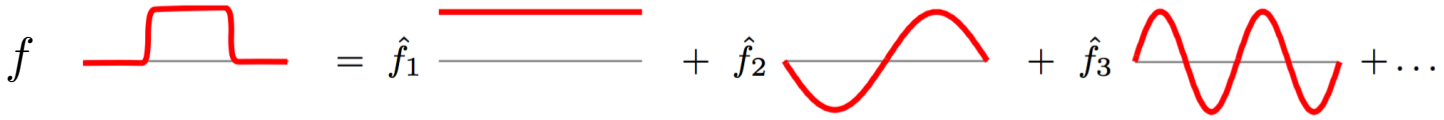
Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- **Spectral Graph Theory**
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Fourier analysis on Euclidean spaces

- **A function** $f : [-\pi, \pi] \to \mathbb{R}$ **can be written as** Fourier series

$$f(x) = \sum_{k \geq 0} \underbrace{\frac{1}{2\pi} \int_{-\pi}^{\pi} f(x') e^{-ikx'} dx'}_{\hat{f}_k = \langle f, e^{-ikx} \rangle_{L^2([-\pi,\pi])}} e^{-ikx}$$



- **Fourier basis** e^{-ikx} = Laplace-Beltrami eigenfunctions:

$$-\Delta \phi_k = k^2 \phi_k$$

$$\begin{array}{rcl} \phi_k & = & \text{Fourier mode} \\ k & = & \text{frequency of Fourier mode} \end{array}$$

Fourier analysis on graphs[11]

- A function $f : \mathcal{V} \to \mathbb{R}$ can be written as Fourier series

$$f_i = \sum_{k=1}^{n} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})}}_{\hat{f}_k} \phi_{k,i}$$

- $\hat{f}_k$ is the k-th graph Fourier coefficient.
- In matrix-vector notation, with the n x n Fourier matrix $\mathbf{\Phi} = [\phi_1, ..., \phi_n]$

$$\hat{\mathbf{f}} = \mathbf{\Phi}^\top \mathbf{f} \quad \text{and} \quad \mathbf{f} = \mathbf{\Phi}\hat{\mathbf{f}}$$

Fourier transform

Inverse Fourier transform

- Graph Fourier basis ϕ_k = Laplacian eigenvectors :

$$\mathbf{\Delta}\phi_k = \lambda_k \phi_k$$

ϕ_k = graph Fourier mode

λ_k = (square) frequency

[11] Hammond, Vandergheynst, Gribonval, 2011

Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- **Spectral Graph Theory**
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Convolution on Euclidean spaces

- Given two functions $f, g : [-\pi, \pi] \to \mathbb{R}$ their convolution is a function

$$(f \star g)(x) = \int_{-\pi}^{\pi} f(x')g(x - x')dx'$$

  - Shift-invariance: $f(x - x_0) \star g(x) = (f \star g)(x - x_0)$
  - Convolution operator commutes with Laplacian: $(\Delta f) \star g = \Delta(f \star g)$
  - Convolution theorem: Convolution can be computed in the Fourier domain as

$$\widehat{(f \star g)} = \hat{f} \cdot \hat{g}$$

  - Efficient computation using FFT: $O(n \log n)$

Convolution on discrete Euclidean spaces

- **Convolution of two vectors** $\mathbf{f} = (f_1, \dots, f_n)^\top$ and $\mathbf{g} = (g_1, \dots, g_n)^\top$

$$(\mathbf{f} \star \mathbf{g})_i = \sum_m g_{(i-m) \bmod n} \cdot f_m$$

$$\mathbf{f} \star \mathbf{g} = \underbrace{\begin{bmatrix} g_1 & g_2 & \cdots & \cdots & g_n \\ g_n & g_1 & g_2 & \cdots & g_{n-1} \\ \vdots & \vdots & \ddots & \ddots & \vdots \\ g_3 & g_4 & \cdots & g_1 & g_2 \\ g_2 & g_3 & \cdots & \cdots & g_1 \end{bmatrix}}_{\text{Circulant matrix}} \begin{bmatrix} f_1 \\ \vdots \\ f_n \end{bmatrix}$$

diagonalised by Fourier basis (Toeplitz)

$$= \mathbf{\Phi} \begin{bmatrix} \hat{g}_1 & & \\ & \ddots & \\ & & \hat{g}_n \end{bmatrix} \mathbf{\Phi}^\top \mathbf{f} = \mathbf{\Phi}\left(\mathbf{\Phi}^\top \mathbf{g} \circ \mathbf{\Phi}^\top \mathbf{f}\right)$$

Convolution on graphs[11]

- Spectral convolution of $f, g \in L^2(\mathcal{V})$ can be defined by analogy

$$(f \star g)_i = \underbrace{\sum_{k \geq 1} \underbrace{\langle f, \phi_k \rangle_{L^2(\mathcal{V})} \langle g, \phi_k \rangle_{L^2(\mathcal{V})}}_{\text{product in the Fourier domain}} \phi_{k,i}}_{\text{inverse Fourier transform}}$$

- In matrix-vector notation

$$\begin{aligned} \mathbf{f} \star \mathbf{g} &= \boldsymbol{\Phi} (\boldsymbol{\Phi}^\top \mathbf{g} \circ \boldsymbol{\Phi}^\top \mathbf{f}) \\ &= \underbrace{\boldsymbol{\Phi} \operatorname{diag}(\hat{g}_1, \dots, \hat{g}_n) \boldsymbol{\Phi}^\top}_{\mathbf{G}} \mathbf{f} \\ &= \boldsymbol{\Phi} \hat{g}(\boldsymbol{\Lambda}) \boldsymbol{\Phi}^\top \mathbf{f} = \hat{g}(\boldsymbol{\Phi} \boldsymbol{\Lambda} \boldsymbol{\Phi}^\top) \mathbf{f} = \hat{g}(\boldsymbol{\Delta}) \mathbf{f} \end{aligned}$$

  - Not shift-invariant! (G has no circulant structure)
  - Commutes with Laplacian: $\mathbf{G}\boldsymbol{\Delta}\mathbf{f} = \boldsymbol{\Delta}\mathbf{G}\mathbf{f}$
  - Filter coefficients depend on basis $\phi_1, \dots, \phi_n$
  - Expensive computation (no FFT): $O(n^2)$

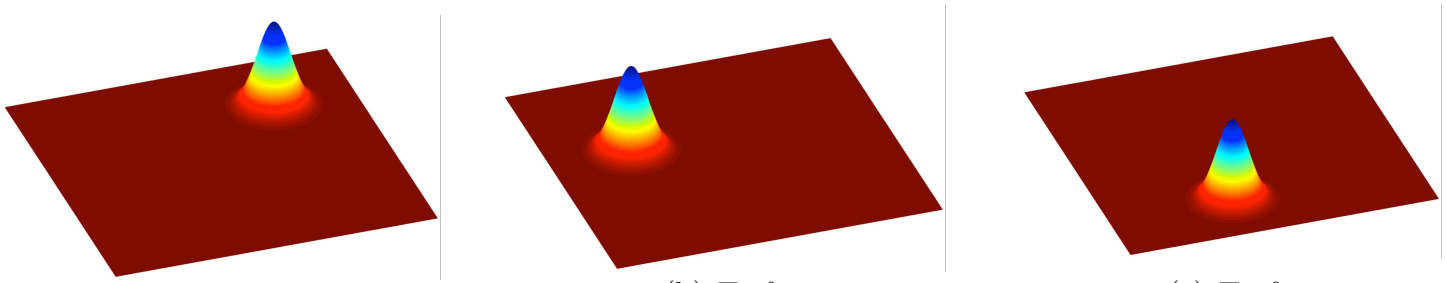
[11] Hammond, Vandergheynst, Gribonval, 2011

No shift invariance on graphs

- A signal f on graph can be translated to vertex i:

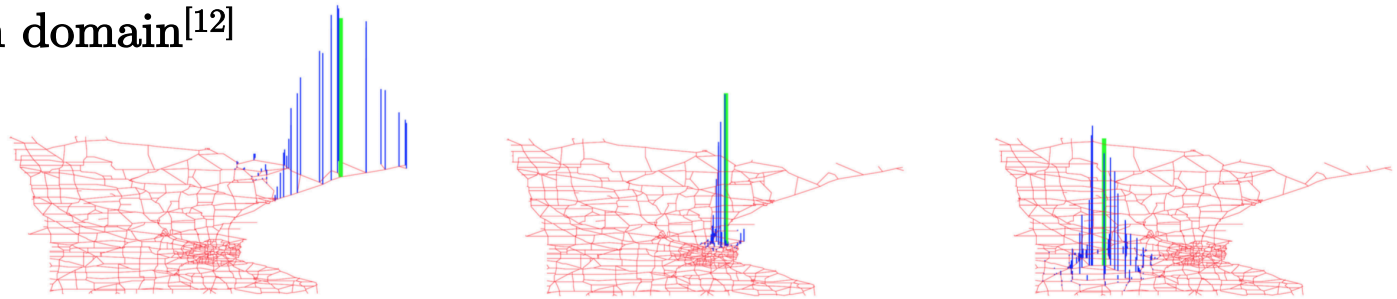
$$T_i \mathbf{f} = \mathbf{f} \star \boldsymbol{\delta}_i$$

- Euclidean domain



(a) $T_s f$ (b) $T_{s'} f$ (c) $T_{s''} f$

- Graph domain[12]



(d) $T_i f$ (e) $T_{i'} f$ (f) $T_{i''} f$

[12] Shuman et al. 2016

Outline

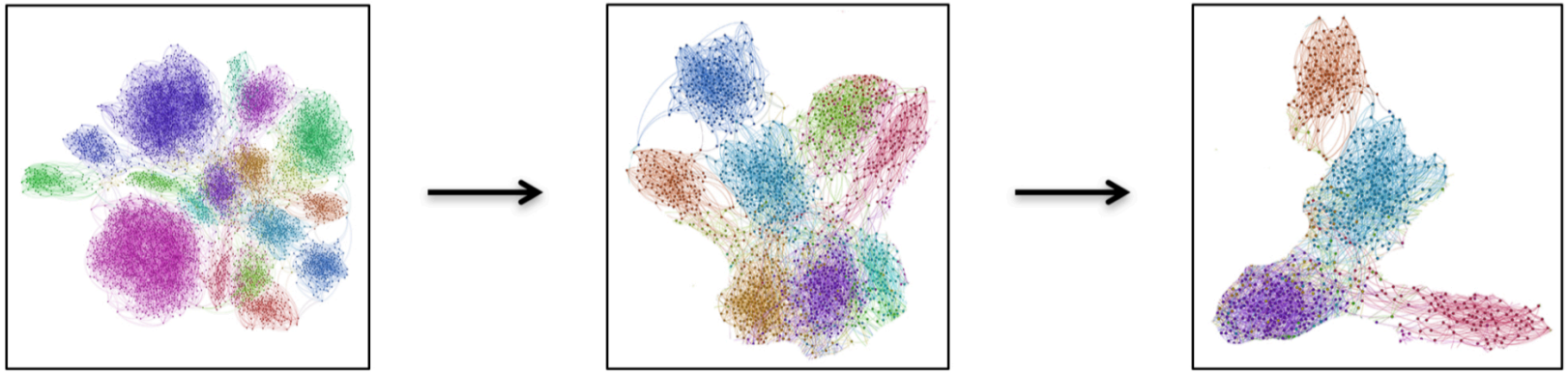
- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- **Spectral Graph Theory**
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Graph coarsening for graph pooling

- Goals:
  - Pool similar local features (max pooling).
  - Series of pooling layers create invariance to global geometric deformations.

- Challenges:
  - Design a multi-scale coarsening algorithm that preserves non-linear graph structures.
  - How to make graph pooling fast?

Graph coarsening = graph partitioning

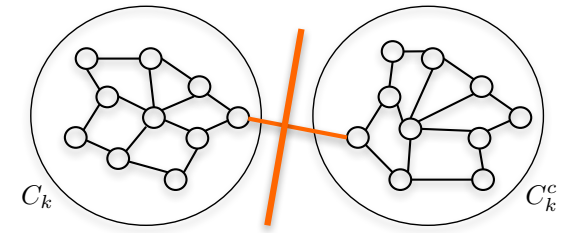


- Graph partitioning decomposes G into smaller meaningful clusters.
- NP-hard ⇒ Approximation

Balanced cuts

- **Powerful combinatorial graph partitioning models:**
  - **Normalized cut**[13]

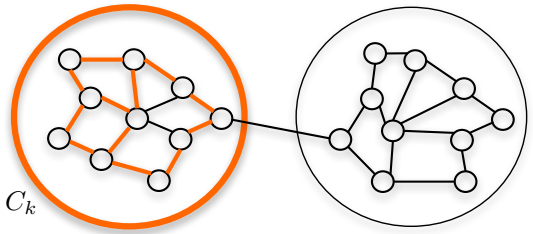
$$\min_{C_1,\dots,C_K} \sum_{k=1}^{K} \frac{\mathrm{Cut}(C_k, C_k^c)}{\mathrm{Vol}(C_k)}$$



Partitioning by min edge cuts.

  - **Normalized association**

$$\max_{C_1,\dots,C_K} \sum_{k=1}^{K} \frac{\mathrm{Assoc}(C_k)}{\mathrm{Vol}(C_k)}$$



Partitioning by max vertex matching.

where $\mathrm{Cut}(A,B) := \sum_{i\in A, j\in B} a_{ij}$, $\mathrm{Assoc}(A) := \sum_{i\in A, i\in B} a_{ij}$,
$\mathrm{Vol}(A) := \sum_{i\in A, j\in B} d_i$, and $d_i := \sum_{j\in V} a_{ij}$.

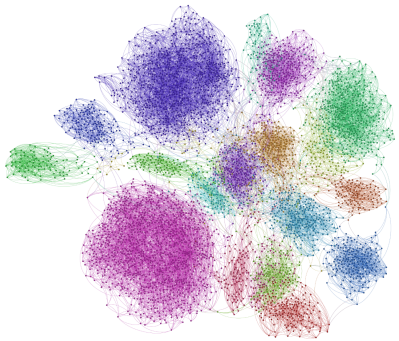
- **Both models are equivalent, but lead to different algorithms.**

[13] Shi, Malik, 2000

Balanced cuts

- Balanced cuts are NP-hard ⇒ most popular approximation techniques focus on linear spectral relaxation (eigenproblem with global solution).

- Graph geometry are generally not linear ⇒ Graclus[14] algorithm computes non-linear clusters that locally maximize the Normalized Association.

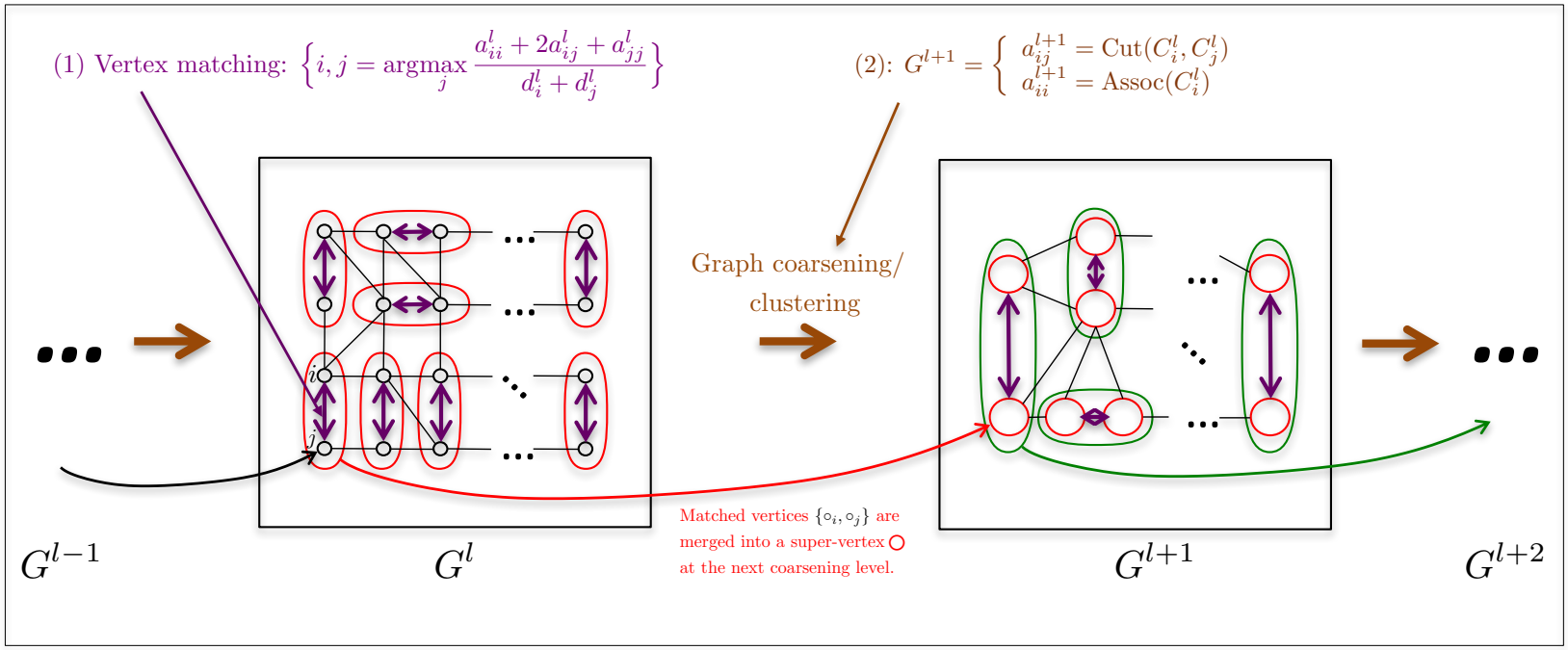


- Graclus algorithm offers a control of the coarsening ratio of ≈ 2 (like image grid) using heavy-edge matching[15].

[14] Dhillon, Guan, Kulis 2007
[15] Karypis, Kumar 1995

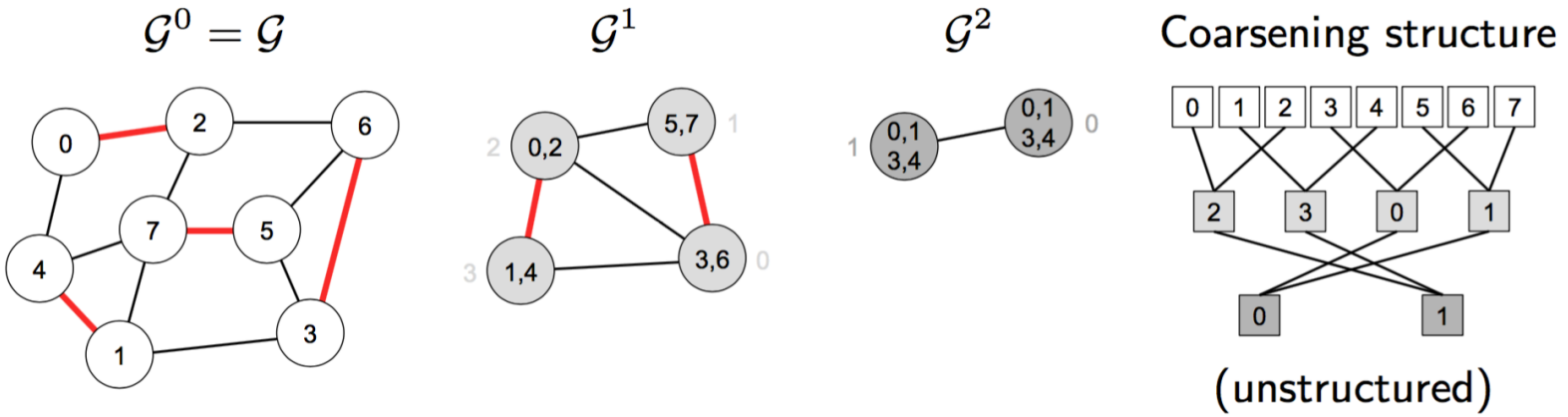
Heavy-Edge Matching (HEM)

- HEM proceeds by two successive steps, vertex matching and graph coarsening (that guarantees a local solution of Norm assoc):



Unstructured pooling

- Sequence of coarsened graphs produced by HEM:

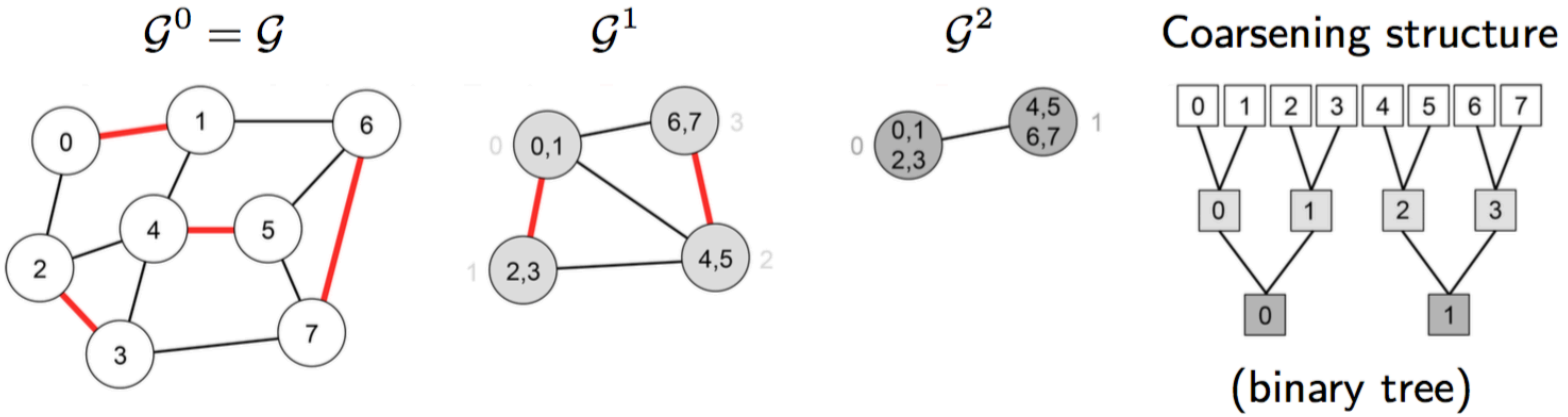


☹ Stores a table of indices for graph and all its coarsened versions

☹ Computationally inefficient

Fast graph pooling[18]

- Structured pooling: Arrangement of the node indexing such that adjacent nodes are hierarchically merged at the next coarser level.



☺ As efficient as 1D-Euclidean grid pooling.

[18] Defferrard, Bresson, Vandergheynst 2016

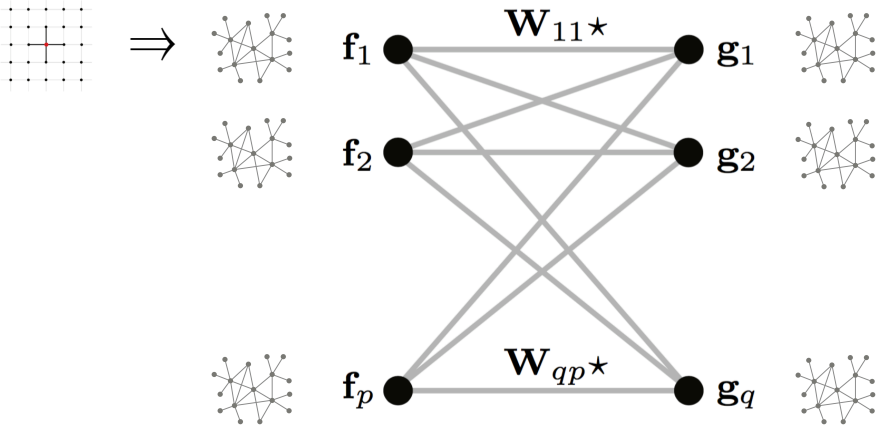
Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- **Spectral ConvNets**
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Vanilla spectral graph ConvNets[16]

- Graph convolutional layer

$$\begin{aligned} \mathbf{f}_l &= l\text{-th data feature on graphs, } \dim(\mathbf{f}_l) = n \times 1 \\ \mathbf{g}_l &= l\text{-th feature map, } \dim(\mathbf{g}_l) = n \times 1 \end{aligned}$$



Conv. layer $\quad \mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{W}_{l,l'} \star \mathbf{f}_{l'}\right)$

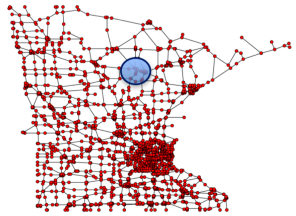
Activation, e.g. $\quad \xi(x) = \max\{x, 0\}$ $\quad$ rectified linear unit (ReLU)

[16] Bruna, Zaremba, Szlam, LeCun 2014

Spectral graph convolution

- Convolutional layer in the spatial domain:

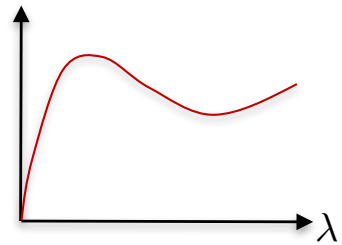
$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{W}_{l,l'} \star \mathbf{f}_{l'}\right),$$



where $\mathbf{W}_{l,l'}$ = matrix of graph spatial filter,

can also be expressed in the spectral domain $(\text{using } \mathbf{g} \star \mathbf{f} = \mathbf{\Phi}\, \hat{g}(\mathbf{\Lambda})\mathbf{\Phi}^\top \mathbf{f})$

$$\mathbf{g}_l = \xi\left(\sum_{l'=1}^{p} \mathbf{\Phi}\hat{\mathbf{W}}_{l,l'}\mathbf{\Phi}^\top \mathbf{f}_{l'}\right),$$



where $\hat{\mathbf{W}}_{l,l'}$ = n x n diagonal matrix of graph spectral filter.

We will denote the spectral filter without the hat symbol, i.e. $\mathbf{W}_{l,l'}$

Vanilla spectral graph ConvNets[16]

- Series of spectral convolutional layers

$$\mathbf{g}_l^{(k)} = \xi \left(\sum_{l'=1}^{q^{(k-1)}} \mathbf{\Phi} \mathbf{W}_{l,l'}^{(k)} \mathbf{\Phi}^\top \mathbf{g}_{l'}^{(k-1)} \right),$$

with spectral coefficients $\mathbf{W}_{l,l'}^{(k)}$ to be learned at each layer.

☺ First spectral graph CNN architecture

☹ No guarantee of spatial localization of filters

☹ $O(n)$ parameters per layer

☹ $O(n^2)$ computation of forward and inverse Fourier transforms $\phi, \phi^\top$ (no FFT on graphs)

☹ Filters are basis-dependent ⇒ does not generalize across graphs

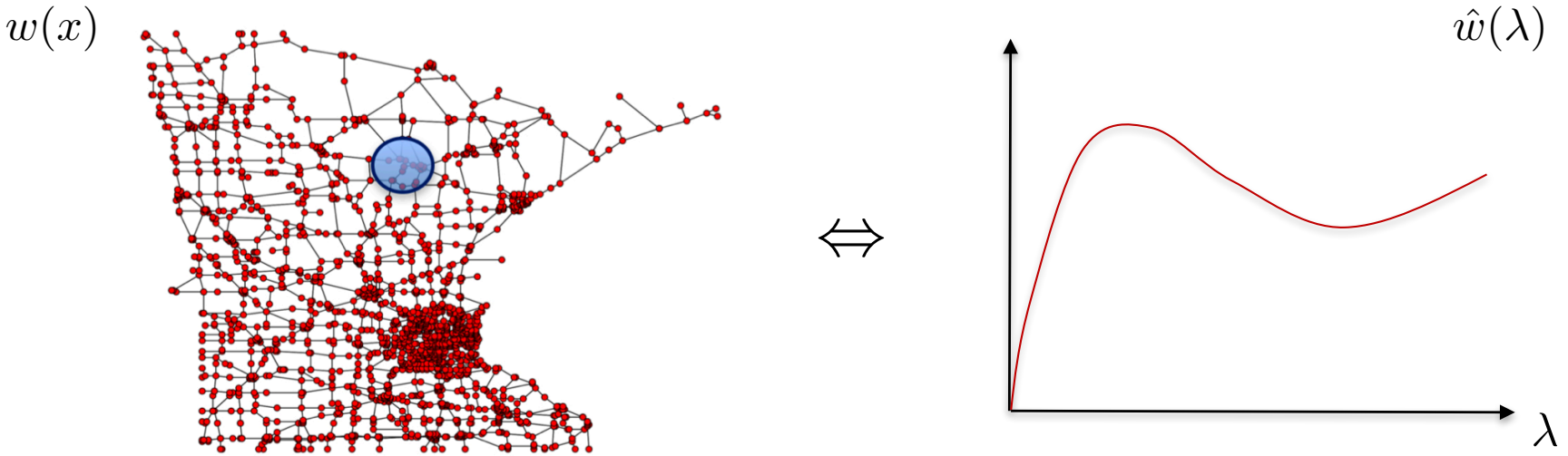
[16] Bruna, Zaremba, Szlam, LeCun 2014

Spatial localization and spectral smoothness[17]

- In the Euclidean setting (by Parseval's identity)

$$\int_{-\infty}^{+\infty} |x|^{2k} |w(x)|^2 dx = \int_{-\infty}^{+\infty} \left| \frac{\partial^k \hat{w}(\lambda)}{\partial \lambda^k} \right|^2 d\lambda$$

$\Rightarrow$ Localization in space = smoothness in frequency domain



Spatial localization $\Leftrightarrow$ Smooth spectral filter function

[17] Henaff, Bruna, LeCun 2015

Smooth parametric spectral filter

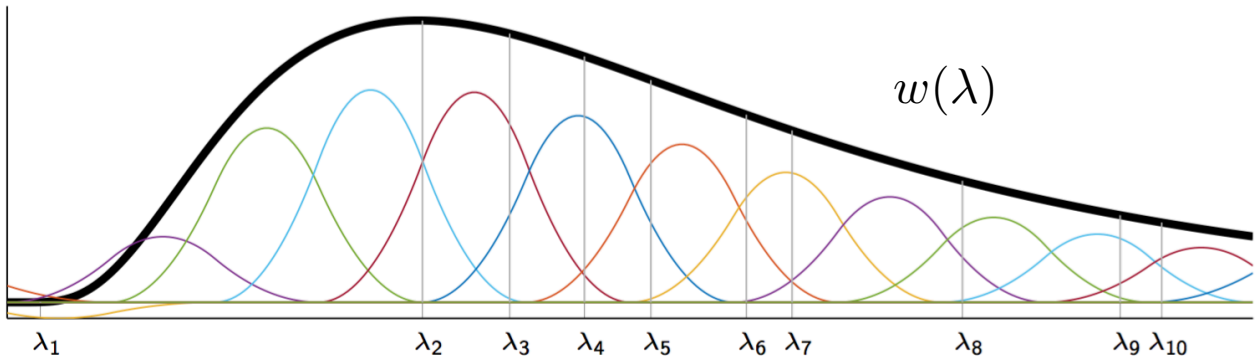
- **Parametrize the smooth spectral filter function $w_(\lambda)$ with a linear combination of smooth kernel functions $\beta_1(\lambda), \ldots, \beta_r(\lambda)$, e.g. splines[17]**

$$w_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda)$$

$$\Downarrow$$

$$w_{\boldsymbol{\alpha}}(\lambda_i) = \sum_{j=1}^{r} \alpha_j \beta_j(\lambda_i) = (\mathbf{B}\boldsymbol{\alpha})_i \quad \Longrightarrow \quad \mathbf{W} = \mathrm{Diag}(\mathbf{B}\boldsymbol{\alpha})$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters



[17] (Litman, Bronstein, 2014); Henaff, Bruna, LeCun 2015

Smooth parametric spectral filter

- Graph convolutional layer: Apply the spectral filter w to a feature signal $\mathbf{f}$

$$
\begin{aligned}
w(\boldsymbol{\Delta})\mathbf{f} &= \boldsymbol{\Phi} w(\boldsymbol{\Lambda}) \boldsymbol{\Phi}^\top \mathbf{f} \\
&= \boldsymbol{\Phi} \begin{pmatrix} w(\lambda_1) & & \\ & \ddots & \\ & & w(\lambda_n) \end{pmatrix} \boldsymbol{\Phi}^\top \mathbf{f}
\end{aligned}
$$

with $\boldsymbol{\alpha}$ learnable parameters

$$
w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \boldsymbol{\Phi} \begin{pmatrix} w_{\boldsymbol{\alpha}}(\lambda_1) & & \\ & \ddots & \\ & & w_{\boldsymbol{\alpha}}(\lambda_n) \end{pmatrix} \boldsymbol{\Phi}^\top \mathbf{f}
$$

SplineNets[17]

- Series of spectral convolutional layers

$$\mathbf{g}_l^{(k)} = \xi \left(\sum_{l'=1}^{q^{(k-1)}} \mathbf{\Phi} \mathbf{W}_{l,l'}^{(k)} \mathbf{\Phi}^\top \mathbf{g}_{l'}^{(k-1)} \right),$$

with smooth spectral parametric coefficients $\mathbf{W}_{l,l'}^{(k)}$ to be learned at each layer.

☺ Fast-decaying filters in space

☺ O(1) parameters per layer

☹ O(n^2) computation of forward and inverse Fourier transforms $\phi, \phi^\top$ (no FFT on graphs)

☹ Filters are basis-dependent ⇒ does not generalize across graphs

[17] Henaff, Bruna, LeCun 2015

Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- **Spectral ConvNets**
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Spectral polynomial filters[18]

- Represent smooth spectral functions with polynomials of Laplacian eigenvalues

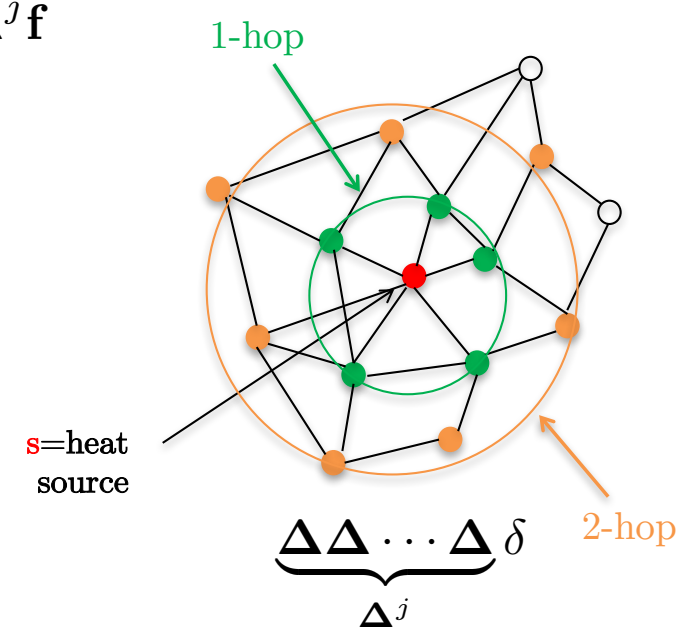
$$w_{\boldsymbol{\alpha}}(\lambda) = \sum_{j=0}^{r} \alpha_j \lambda^j$$

where $\boldsymbol{\alpha} = (\alpha_1, \ldots, \alpha_r)^\top$ is the vector of filter parameters.

- Convolutional layer: Apply spectral filter to feature signal f

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \textstyle\sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

- Key observation: Each Laplacian operation increases the support of a function by 1-hop ⇒ Exact control the size of Laplacian-based filters.



[18] Defferrard, Bresson, Vandergheynst 2016

Linear complexity

- Application of the filter to a feature signal f

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

- Denote $\mathbf{X}_0 = \mathbf{f}$ and define $\mathbf{X}_1 = \boldsymbol{\Delta}\mathbf{X}_0 = \boldsymbol{\Delta}\mathbf{f}$ and the sequence $\mathbf{X}_j = \boldsymbol{\Delta}\mathbf{X}_{j-1}$

$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \mathbf{X}_j$$

- Two important observations:

  1. *No* need to compute the eigendecomposition of the Laplacian (ϕ,Λ).

  2. Observe that $\{X_j\}$ are generated by multiplication of a sparse matrix and a vector $\Rightarrow$ Complexity is $O(Er)$=$O(n)$ for sparse graphs.

- Graph convolutional layers are GPU friendly.

Spectral graph ConvNets with polynomial filters

- Series of spectral convolutional layers

$$\mathbf{g}_l^{(k)} = \xi\left(\sum_{l'=1}^{q^{(k-1)}} \mathbf{\Phi}\mathbf{W}_{l,l'}^{(k)}\mathbf{\Phi}^\top \mathbf{g}_{l'}^{(k-1)}\right),$$

with spectral polynomial coefficients $\mathbf{W}_{l,l'}^{(k)}$ to be learned at each layer.

- ☺ Filters are exactly localized in r-hops support
- ☺ O(1) parameters per layer
- ☺ No computation of ϕ, $\phi^\top \Rightarrow$ O(n) computational complexity (assuming sparsely-connected graphs)
- ☹ Unstable under coefficients perturbation (hard to optimize)
- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs

[18] Defferrard, Bresson, Vandergheynst 2016

Chebyshev polynomials

- Graph convolution with (non-orthogonal) monomial basis $1, x, x^2, x^3, \cdots$

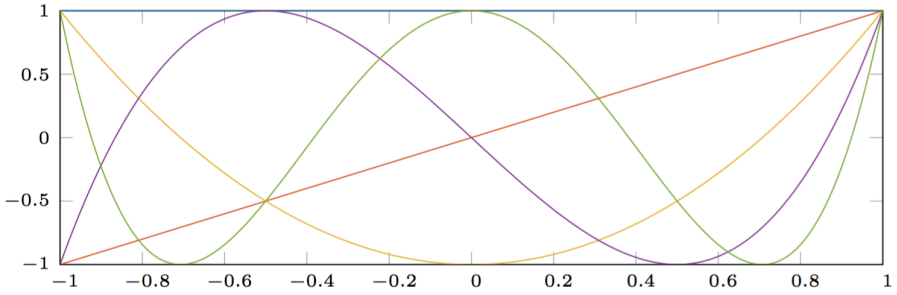
$$w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \boldsymbol{\Delta}^j \mathbf{f}$$

- Graph convolution with (orthogonal) Chebyshev polynomials

$$w_{\boldsymbol{\alpha}}(\tilde{\boldsymbol{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\boldsymbol{\Delta}})\mathbf{f}$$

- Orthonormal on $L^2([-1,+1])$ w.r.t. $\langle f, g \rangle = \int_{-1}^{+1} f(\tilde{\lambda}) g(\tilde{\lambda}) \frac{d\tilde{\lambda}}{\sqrt{1-\tilde{\lambda}^2}}$

- Stable under perturbation of coefficients



T_0, T_1, T_2, T_3

ChebNets[18]

- Application of the filter with the scaled Laplacian $\tilde{\mathbf{\Delta}} = 2\lambda_n^{-1}\mathbf{\Delta} - \mathbf{I}$

$$w_{\boldsymbol{\alpha}}(\tilde{\mathbf{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j T_j(\tilde{\mathbf{\Delta}})\mathbf{f} = \sum_{j=0}^{r} \alpha_j \mathbf{X}^{(j)}$$

with

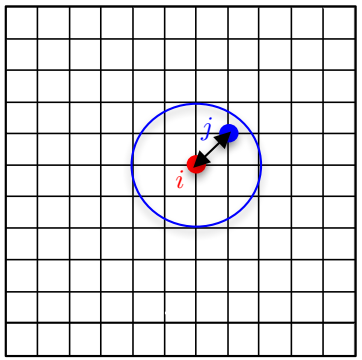
$$\begin{aligned} \mathbf{X}^{(j)} &= T_j(\tilde{\mathbf{\Delta}})\mathbf{f} \\ &= 2\tilde{\mathbf{\Delta}}\mathbf{X}^{(j-1)} - \mathbf{X}^{(j-2)}, \qquad \mathbf{X}^{(0)} = \mathbf{f}, \quad \mathbf{X}^{(1)} = \tilde{\mathbf{\Delta}}\mathbf{f} \end{aligned}$$

- ☺ Filters are exactly localized in r-hops support
- ☺ O(1) parameters per layer
- ☺ No computation of ϕ, $\phi^\top \Rightarrow$ O(n) computational complexity (assuming sparsely-connected graphs)
- ☺ Stable under coefficients perturbation
- ☹ Filters are basis-dependent $\Rightarrow$ does not generalize across graphs

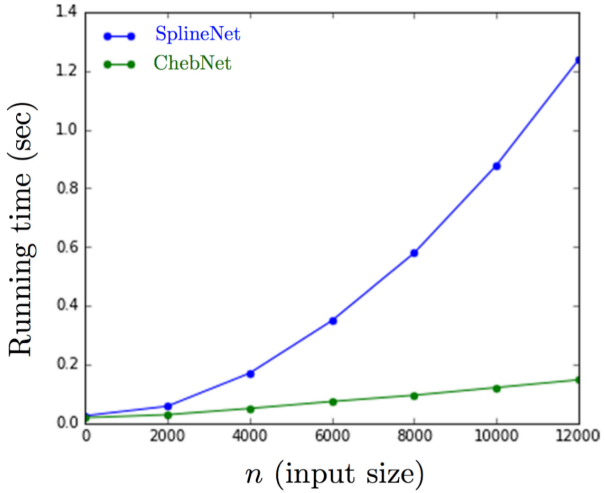
[18] Defferrard, Bresson, Vandergheynst 2016

Numerical experiments

Graph: a 8-NN graph of the Euclidean grid



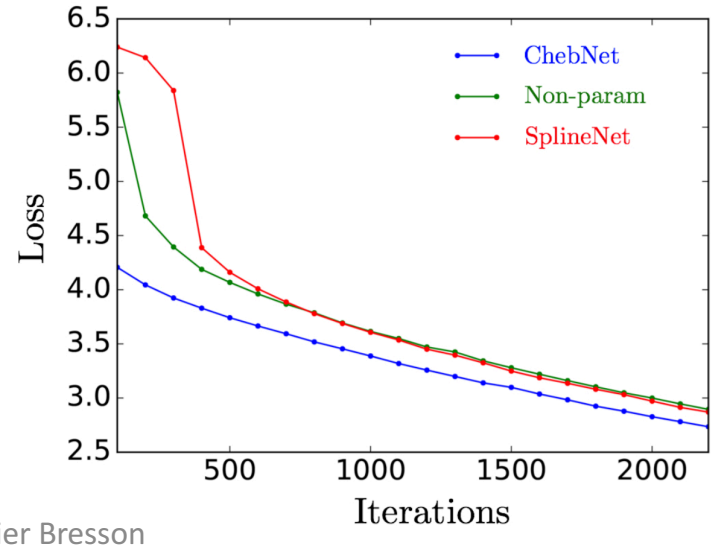
- Running time

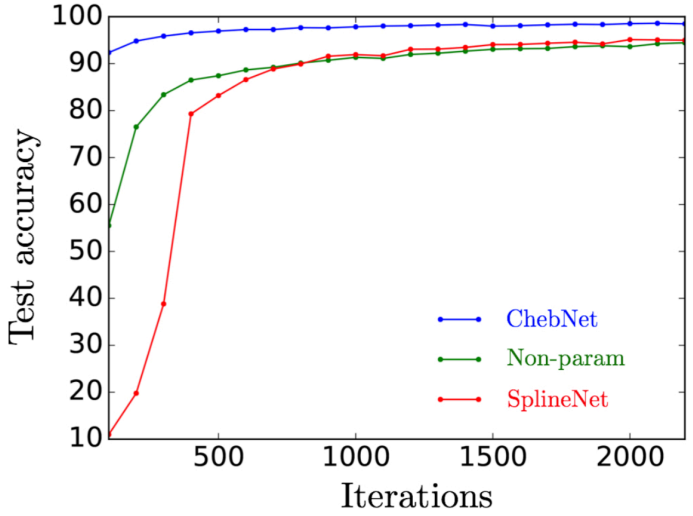


- Accuracy

Model	Order	Accuracy
LeNet5	-	99.33%
SplineNet	25	97.75%
ChebNet	25	99.14%

- Optimization





Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- **Spectral ConvNets**
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Graph convolutional nets[19]: simplified ChebNets

- Use Chebychev polynomials of degree $r=2$ and assume $\lambda_n \approx 2$

$$\begin{aligned} w_{\boldsymbol{\alpha}}(\boldsymbol{\Delta})\mathbf{f} &= \alpha_0\mathbf{f} + \alpha_1(\boldsymbol{\Delta} - \mathbf{I})\mathbf{f} \\ &= \alpha_0\mathbf{f} - \alpha_1\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}\mathbf{f} \end{aligned}$$

- Further constrain $\alpha = \alpha_0 = -\alpha_1$ to obtain a single-parameter filter

$$w_\alpha(\boldsymbol{\Delta})\mathbf{f} = \alpha(\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2})\mathbf{f}$$

- Caveat: The eigenvalues of $\mathbf{I} + \mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$ are now in [0,2]
  $\Rightarrow$ repeated application of the filter results in numerical instability

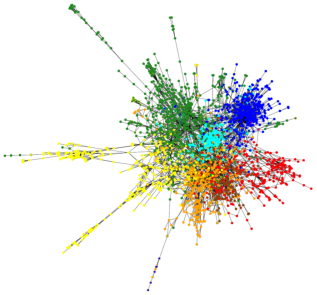
- Fix: Apply a renormalization

$$w_\alpha(\boldsymbol{\Delta})\mathbf{f} = \alpha\tilde{\mathbf{D}}^{-1/2}\tilde{\mathbf{W}}\tilde{\mathbf{D}}^{-1/2}\mathbf{f}$$

$$\text{with } \tilde{\mathbf{W}} = \mathbf{W} + \mathbf{I} \text{ and } \tilde{\mathbf{D}} = \text{diag}(\textstyle\sum_{j\neq i} \tilde{w}_{ij})$$

[19] Kipf, Welling 2016

Example: citation networks



Method	Cora[1]	PubMed[2]
Manifold Regularization[3]	59.5%	70.7%
Semidefinite Embedding[4]	59.0%	71.1%
Label Propagation[5]	68.0%	63.0%
DeepWalk[6]	67.2%	65.3%
Planetoid[7]	75.7%	77.2%
Graph Convolutional Net[8]	**81.59%**	**78.72%**

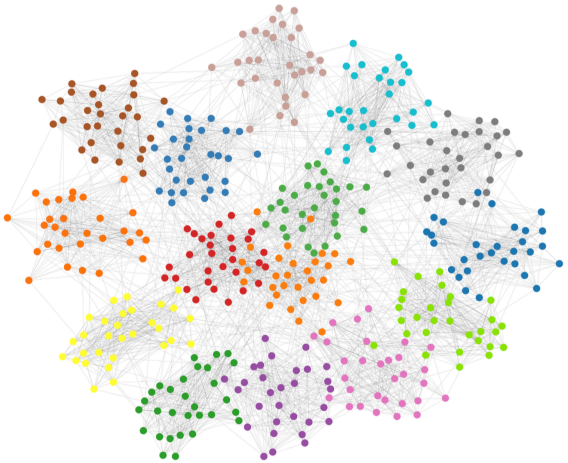
Classification accuracy of different methods on citation network datasets

Monti et al. 2016; data: [1,2]Sen et al. 2008; methods: [3]Belkin et al. 2006; [4]Weston et al. 2012; [5]Zhu et al. 2003; [6]Perozzi et al. 2014; [7]Yang et al. 2016; [8]Kipf, Welling 2016

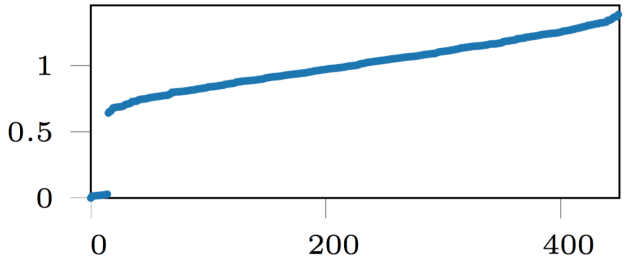
Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- **Spectral ConvNets**
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- Conclusion

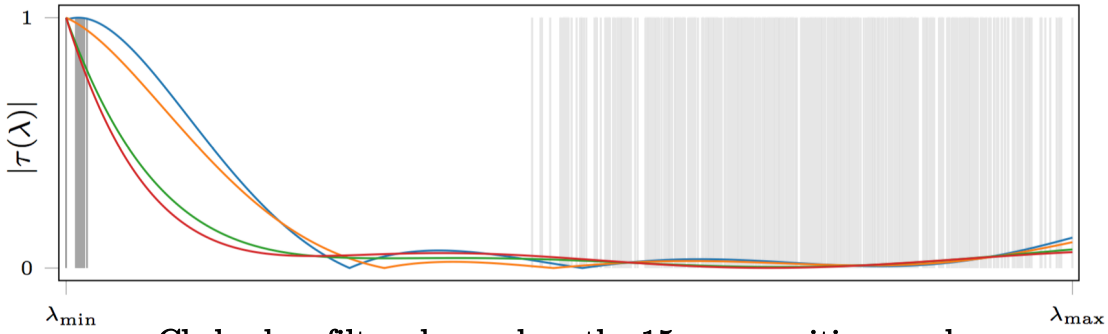
Community graphs



Synthetic graph with 15 communities



Normalized Laplacian eigenvalues



Chebyshev filters learned on the 15-communities graph

☹ ChebNet is unstable to produce filters with frequency bands of interest

Spectral graph ConvNets with Cayley polynomials[20]

- Cayley polynomials are a family of real-valued rational functions (w/ complex coefficients)

$$p(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j \Big(\underbrace{\frac{\lambda - i}{\lambda + i}}_{C(\lambda)} \Big)^j \Big\}$$

- Cayley transform

$$C(\lambda) = \tfrac{\lambda - i}{\lambda + i} \text{ is a smooth bijection from } \mathbb{R} \text{ to } e^{i\mathbb{R}} \setminus \{1\}$$

- Expressivity: Since $z^{-1} = \bar{z}$ for $z \in e^{i\mathbb{R}}$ and $2\mathrm{Re}\{z\} = z + \bar{z}$ we can rewrite $p(\lambda)$ as a real trigonometric polynomial w.r.t. $C(\lambda)$

$$\begin{aligned} p(\lambda) &= c_0 + \sum_{j=1}^{r} c_j \underbrace{C^j(\lambda)}_{e^{ij\omega_\lambda}} + \bar{c_j} \underbrace{C^{-j}(\lambda)}_{e^{-ij\omega_\lambda}} \\ &= 2 \sum_{j=1}^{r} \mathrm{Re}\{c_j\} \cos(j\omega_\lambda) - \mathrm{Im}\{c_j\} \sin(j\omega_\lambda) \end{aligned}$$

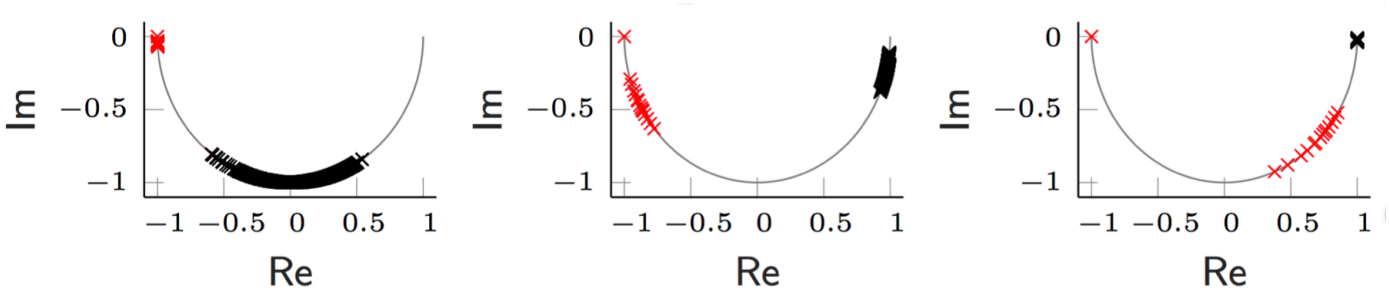
[20] (Cayley 1846); Levie, Monti, Bresson, Bronstein 2017

Spectral zoom[20]

- Applying Cayley transform to the scaled Laplacian $h\boldsymbol{\Delta}$

$$C(h\boldsymbol{\Delta}) = (h\boldsymbol{\Delta} - i\mathbf{I})(h\boldsymbol{\Delta} + i\mathbf{I})^{-1}$$

results in a non-linear transformation of the eigenvalues (spectral zoom)



Cayley transform $C(h\lambda)$ of the 15-communities graph Laplacian spectrum

[20] Levie, Monti, Bresson, Bronstein 2017

Fast inversion

- Application of Cayley filter $w(h\mathbf{\Delta})\mathbf{f}$ requires the solution of recursive linear system

$$\begin{aligned} \mathbf{y}_0 &= \mathbf{f} \\ (h\mathbf{\Delta} + i\mathbf{I})\mathbf{y}_j &= (h\mathbf{\Delta} - i\mathbf{I})\mathbf{y}_{j-1} \qquad j = 1, \dots, r \end{aligned}$$

- Exact (stable) solution: $\mathrm{O}(n^3)$ complexity

- Approximate solution using K Jacobi iterations

$$\begin{aligned} \tilde{\mathbf{y}}_j^{(k+1)} &= \mathbf{J}\tilde{\mathbf{y}}_j^{(k)} + \mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})(h\mathbf{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \\ \tilde{\mathbf{y}}_j^{(0)} &= \mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})(h\mathbf{\Delta} - i\mathbf{I})\tilde{\mathbf{y}}_{j-1} \end{aligned}$$

with $\mathbf{J} = -\mathrm{Diag}^{-1}(h\mathbf{\Delta} + i\mathbf{I})\mathrm{Off}(h\mathbf{\Delta} + i\mathbf{I})$

- Application of the approximate Cayley filter

$$\sum_{j=0}^{r} c_j \tilde{\mathbf{y}}_j \approx w(h\mathbf{\Delta})\mathbf{f}$$

Complexity: $\mathrm{O}(ErK) = \mathrm{O}(n)$

CayleyNets[30]

- Represent spectral transfer function as a Cayley polynomial of order r

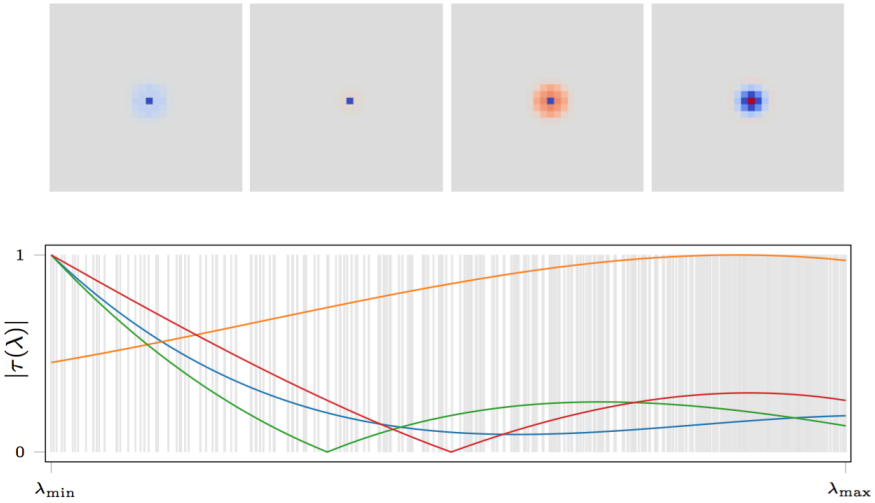
$$w_{\mathbf{c},h}(\lambda) = c_0 + 2\mathrm{Re}\Big\{ \sum_{j=1}^{r} c_j (h\lambda - i)^j (h\lambda + i)^{-j} \Big\}$$

where the filter parameters are the vector of complex coefficients $\mathbf{c} = (c_0, \dots, c_r)^\top$ and the spectral zoom h.

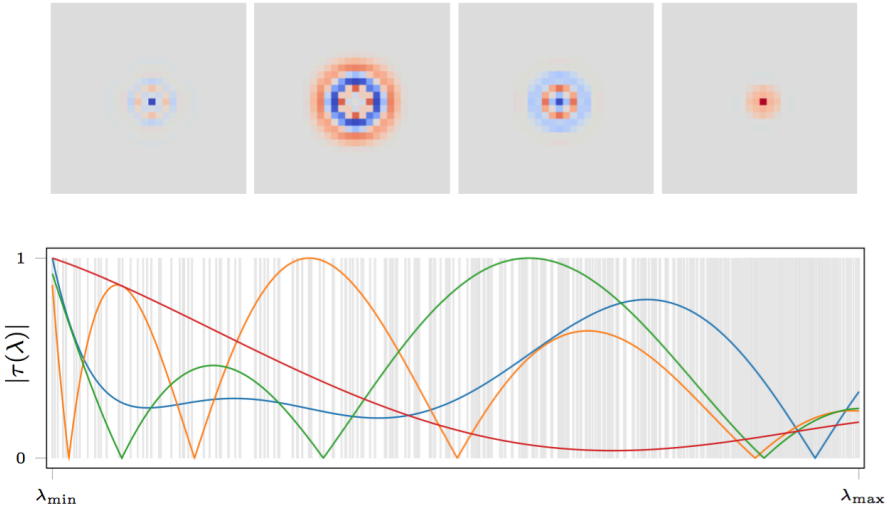
- ☺ Filters have guaranteed exponential spatial decay
- ☺ O(1) parameters per layer
- ☺ O(n) computational complexity with Jacobi approximate inversion (assuming sparsely-connected graph)
- ☺ Spectral zoom property allowing to better localize in frequency
- ☺ Richer class of filters than Chebyshev for the same order
- ☹ Filters are basis-dependent ⇒ does not generalize across graphs

[20] Levie, Monti, Bresson, Bronstein 2017

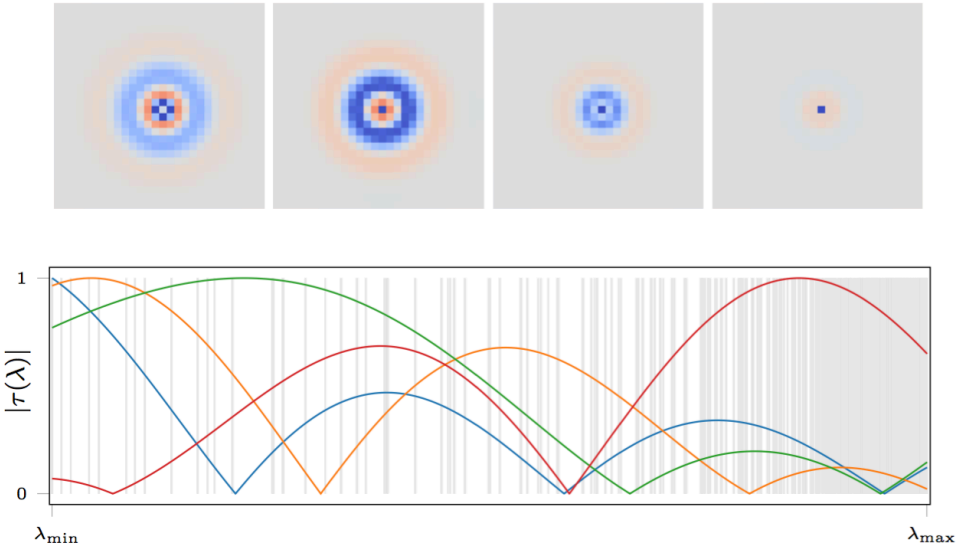
Chebyshev vs Cayley



Example of Chebyshev filters (order $r = 3$) on Euclidean grid

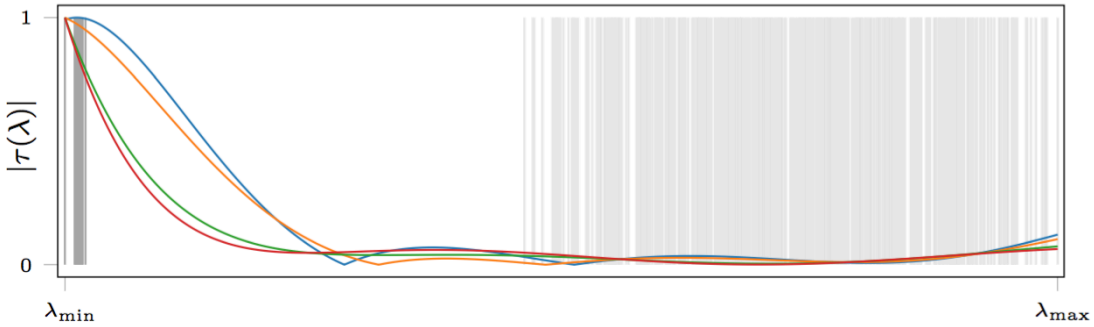


Example of Chebyshev filters (order $r = 7$) on Euclidean grid



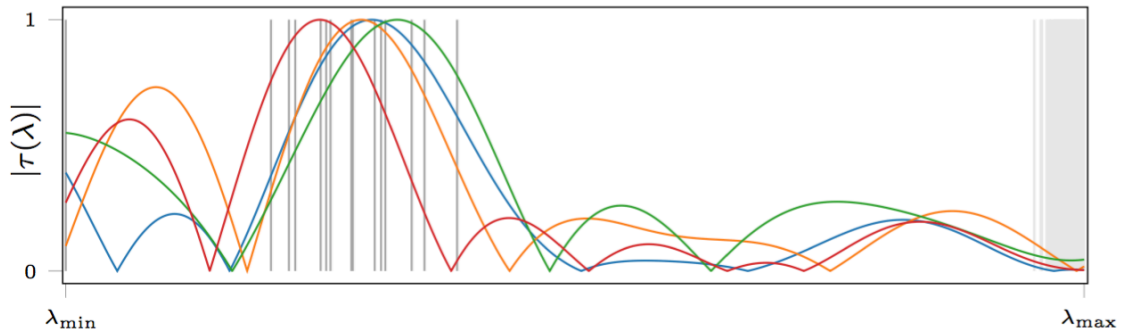
Example of Cayley filters (order $r = 3$) on Euclidean grid

Chebyshev vs Cayley: community graphs example



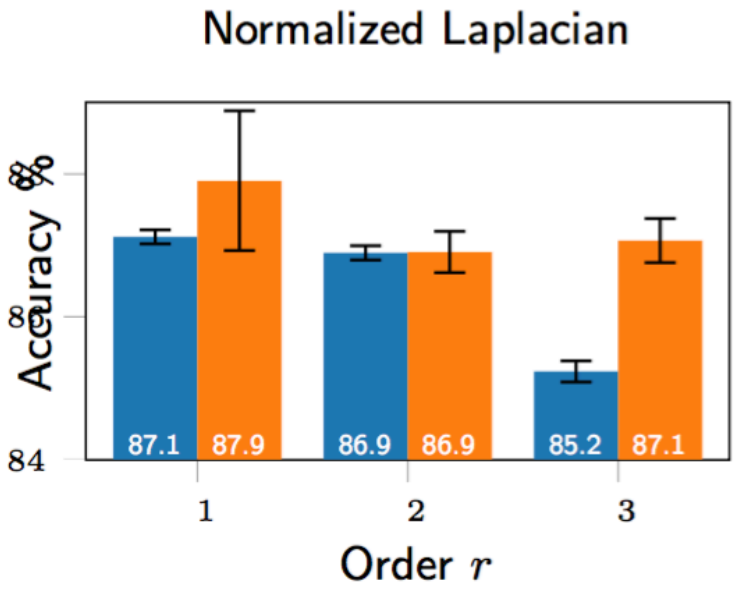
Example of Chebyshev filters learned on the 15-communities graph

Synthetic graph with 15 communities



Example of Cayley filters learned on the 15-communities graph

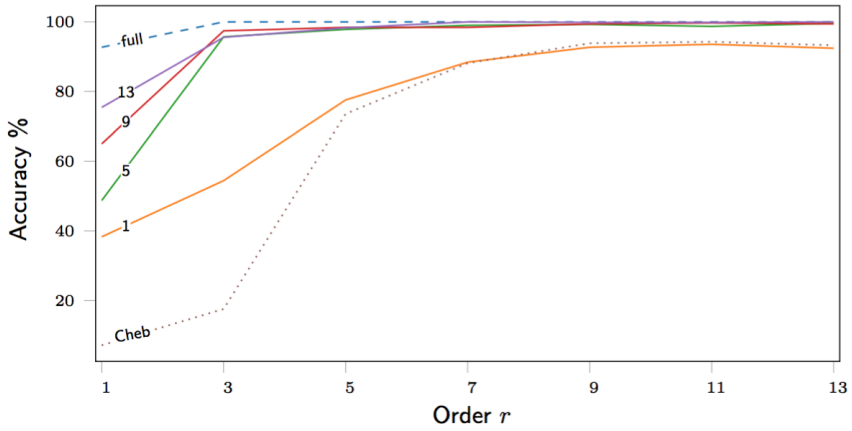
Chebyshev vs Cayley: Cora example



Accuracy of ChebNet (blue) and CayleyNet (orange) on Cora dataset

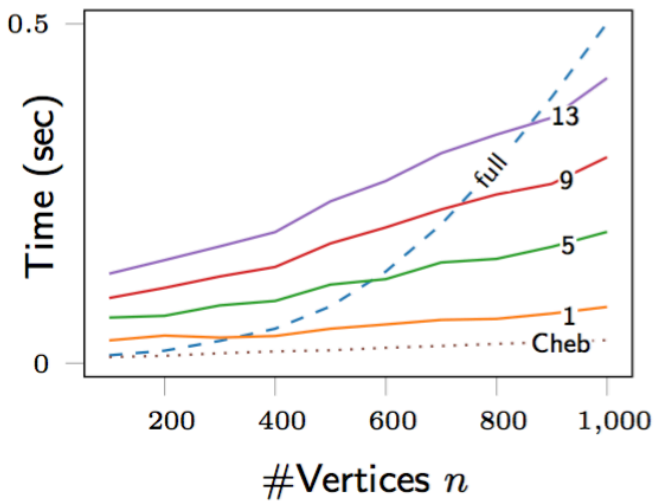
Approximate inversion

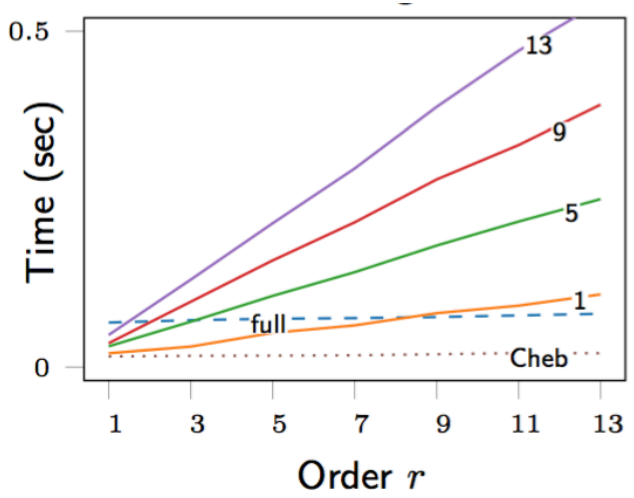
- Accuracy of approximate inversion



Community detection accuracy of CayleyNet using approximate Jacobi inversion on the synthetic 15-community graph

- Speed of approximate inversion

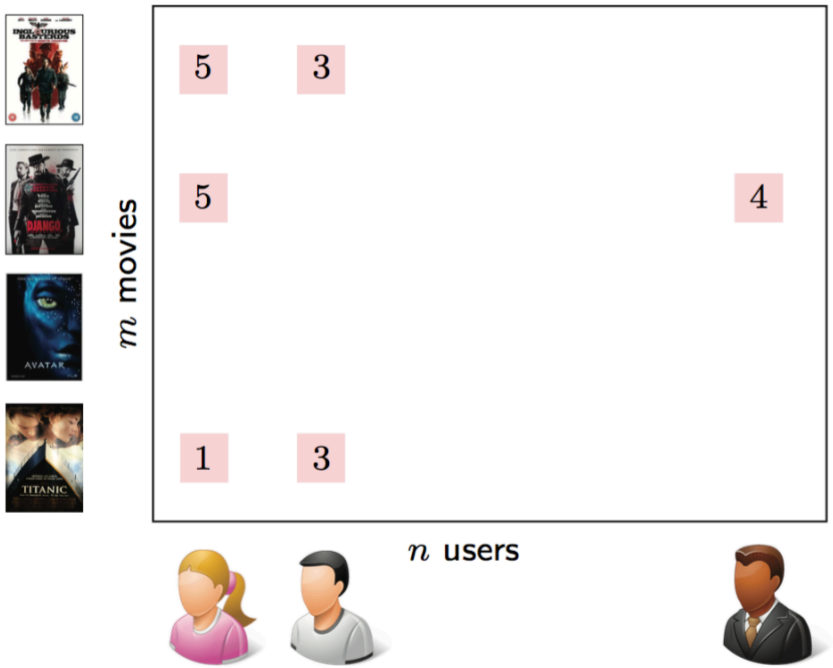




Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- **Spectral ConvNets on Multiple Graphs**
  - *Recommender Systems* [NIPS'17]*
- Conclusion

Matrix completion - Netflix challenge



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \text{rank}(\mathbf{X}) \quad \text{s.t.} \quad x_{ij} = a_{ij} \;\; \forall ij \in \Omega$$

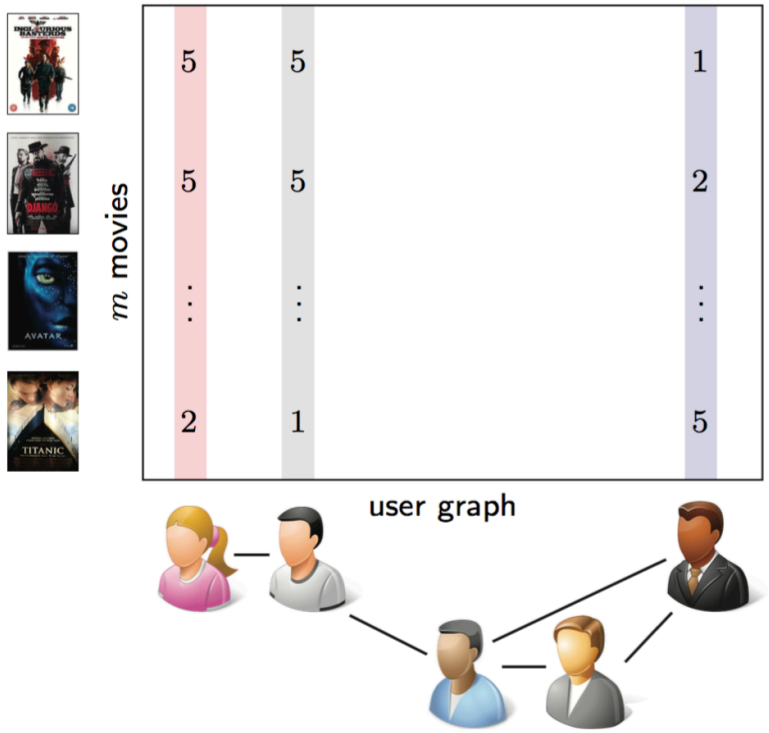
$$\Downarrow \text{ Convex relaxation}^{[21]}$$

$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \|\mathbf{X}\|_* \quad \text{s.t.} \quad x_{ij} = a_{ij} \;\; \forall ij \in \Omega$$

$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \|\mathbf{X}\|_* + \mu\|\mathbf{\Omega} \circ (\mathbf{X} - \mathbf{A})\|_{\text{F}}^2$$

[21] Candes et al. 2008

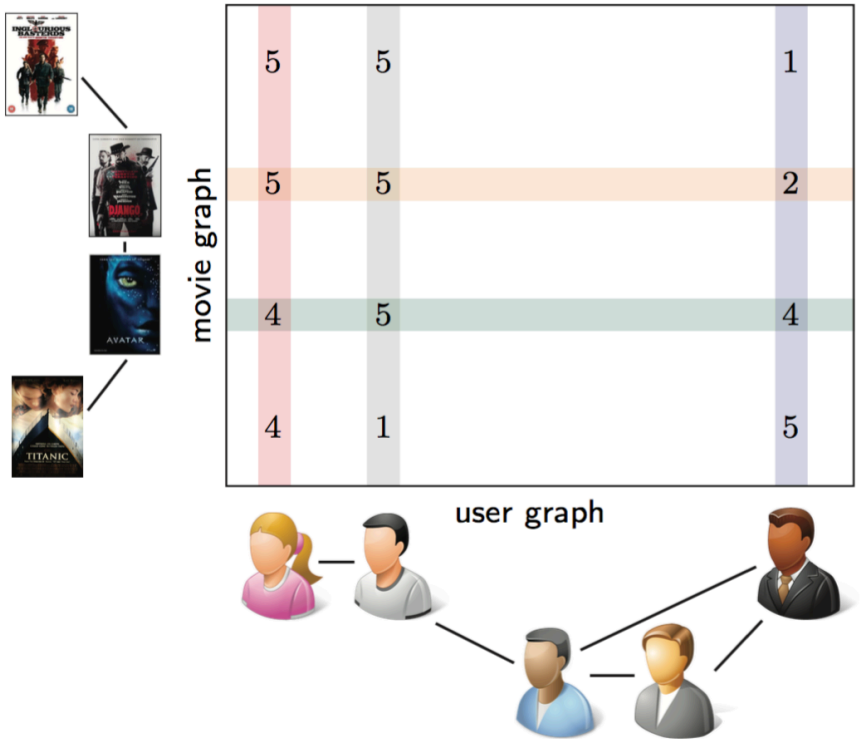
Matrix completion on single graph[22]



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \|\mathbf{X}\|_* + \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2 + \mu_{\mathrm{c}}\underbrace{\mathrm{tr}(\mathbf{X}\mathbf{\Delta}_{\mathrm{c}}\mathbf{X}^\top)}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{c}}}^2}$$

[22] Kalofolias, Bresson, Bronstein, Vandergheynst, 2014

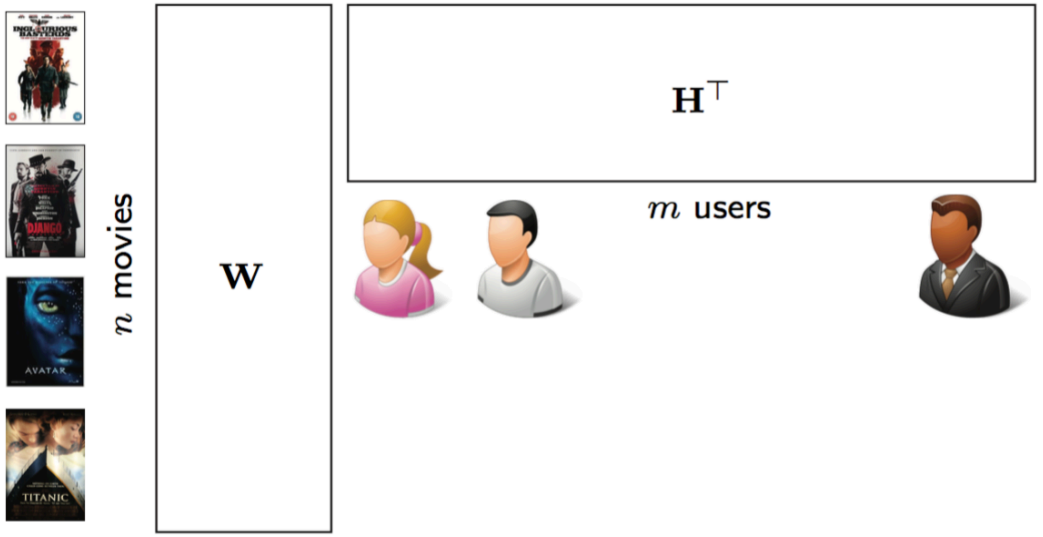
Matrix completion on multiple graphs[22]



$$\min_{\mathbf{X}\in\mathbb{R}^{m\times n}} \quad \|\mathbf{X}\|_* + \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2 + \mu_{\mathrm{c}}\underbrace{\mathrm{tr}(\mathbf{X}\mathbf{\Delta}_{\mathrm{c}}\mathbf{X}^\top)}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{c}}}^2} + \mu_{\mathrm{r}}\underbrace{\mathrm{tr}(\mathbf{X}^\top\mathbf{\Delta}_{\mathrm{r}}\mathbf{X})}_{\|\mathbf{X}\|_{\mathcal{G}_{\mathrm{r}}}^2}$$

[22] Kalofolias, Bresson, Bronstein, Vandergheynst, 2014

Factorized matrix completion models[23]



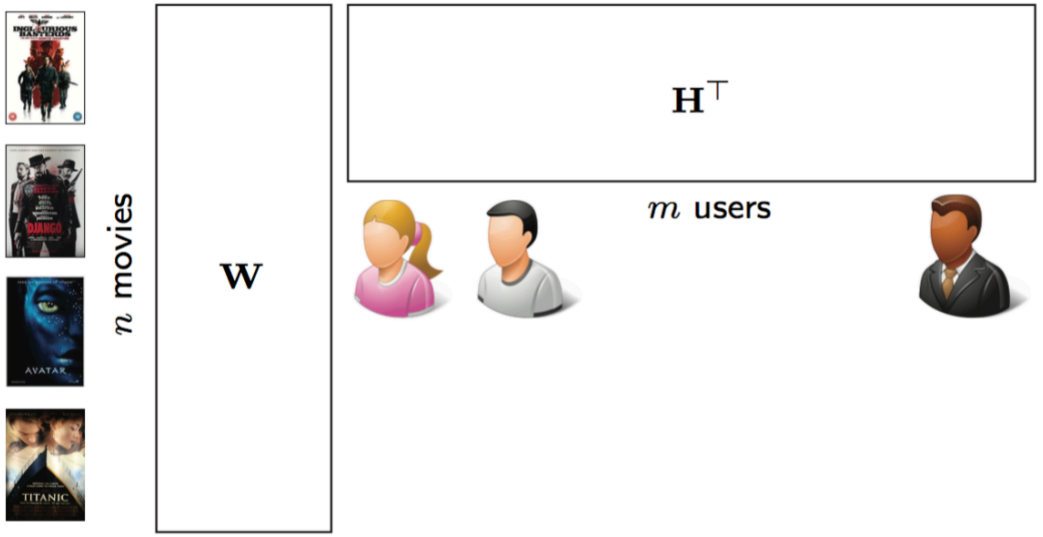
$$\min_{\substack{\mathbf{W}\in\mathbb{R}^{m\times s}\\ \mathbf{H}\in\mathbb{R}^{n\times s}}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\|\mathbf{W}\|_{\mathrm{F}}^2+\mu_{\mathrm{r}}\|\mathbf{H}\|_{\mathrm{F}}^2$$

☺ $\mathrm{O}(n+m)$ variables instead of $\mathrm{O}(nm)$

☺ $\mathrm{rank}(\mathrm{X}) = \mathrm{rank}(\mathrm{WH}^{\mathrm{T}}) \leq s$ by construction

[23] Srebro, Rennie, Jaakkola 2004

Factorized matrix completion on graphs[24]



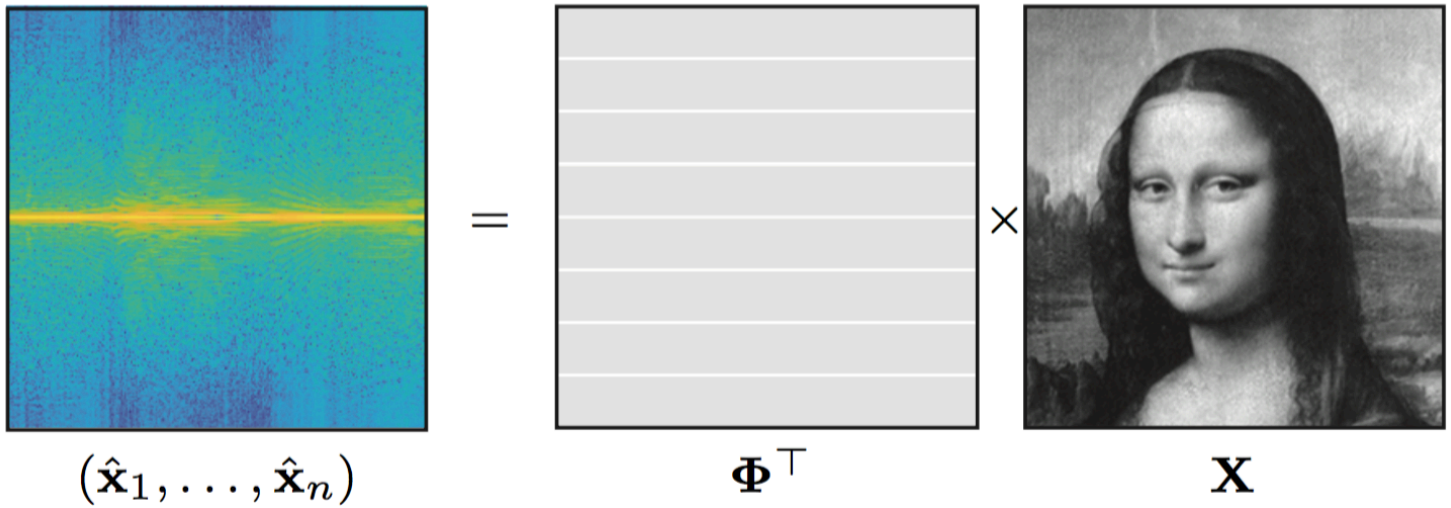
$$\min_{\substack{\mathbf{W}\in\mathbb{R}^{m\times s}\\ \mathbf{H}\in\mathbb{R}^{n\times s}}} \quad \mu\|\mathbf{\Omega}\circ(\mathbf{X}-\mathbf{A})\|_{\mathrm{F}}^2+\mu_{\mathrm{c}}\mathrm{tr}(\mathbf{H}^\top\mathbf{\Delta}_{\mathrm{c}}\mathbf{H})+\mu_{\mathrm{r}}\mathrm{tr}(\mathbf{W}^\top\mathbf{\Delta}_{\mathrm{r}}\mathbf{W})$$

☺ $\mathrm{O}(n+m)$ variables instead of $\mathrm{O}(nm)$

☺ $\mathrm{rank}(\mathrm{X}) = \mathrm{rank}(\mathrm{WH}^{\mathrm{T}}) \leq s$ by construction

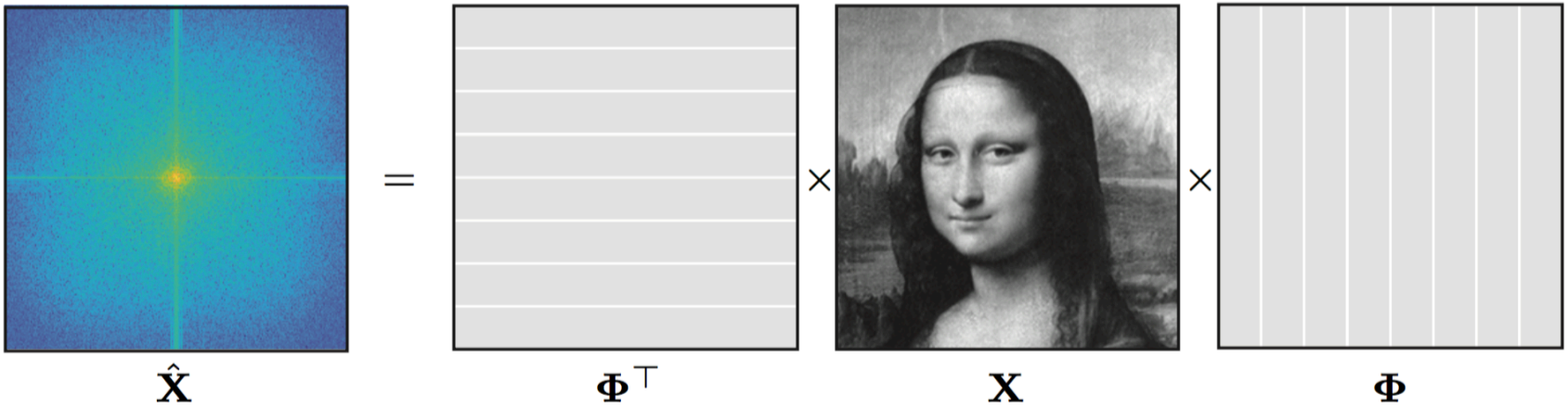
[24] Rao et al. 2015

1D Fourier transform



Column-wise transform

2D Fourier transform



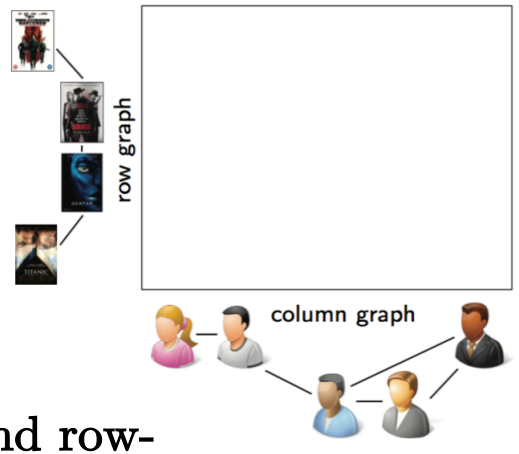
Column-wise transform + Row-wise transform = 2D transform

From 1D signal processing to 2D image processing

Multi-graph Fourier transform and convolution[25]

- Multi-graph Fourier transform

$$\hat{\mathbf{X}} = \mathbf{\Phi}_{\mathrm{r}}^{\top} \mathbf{X} \mathbf{\Phi}_{\mathrm{c}}$$



where $\mathbf{\Phi}_{\mathrm{c}}$, $\mathbf{\Phi}_{\mathrm{r}}$ are the eigenvectors of the column- and row-graph Laplacians $\mathbf{\Delta}_{\mathrm{c}}$, $\mathbf{\Delta}_{\mathrm{r}}$, respectively.

- Multi-graph spectral convolution

$$\begin{aligned} \mathbf{X} \star \mathbf{Y} &= \mathbf{\Phi}_{\mathrm{r}}((\mathbf{\Phi}_{\mathrm{r}}^{\top} \mathbf{X} \mathbf{\Phi}_{\mathrm{c}}) \circ (\mathbf{\Phi}_{\mathrm{r}}^{\top} \mathbf{Y} \mathbf{\Phi}_{\mathrm{c}})) \mathbf{\Phi}_{\mathrm{c}}^{\top} \\ &= \mathbf{\Phi}_{\mathrm{r}}(\hat{\mathbf{X}} \circ \hat{\mathbf{Y}}) \mathbf{\Phi}_{\mathrm{c}}^{\top} \end{aligned}$$

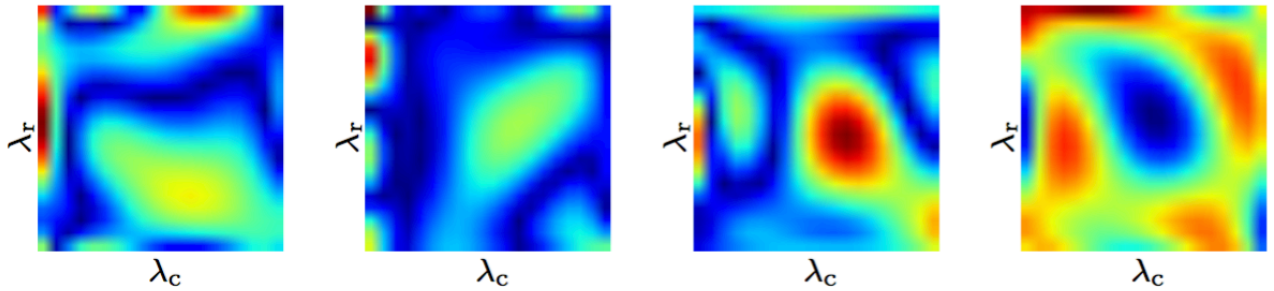
[25] Monti, Bresson, Bronstein 2017

Multi-graph spectral filters

- Parametrize the multi-graph filter as a smooth function of column- and row frequencies w/ e.g. bivariate Chebyshev polynomial

$$w_{\mathbf{\Theta}}(\tilde{\lambda}_{\mathrm{c}}, \tilde{\lambda}_{\mathrm{r}}) = \sum_{j,j'=1}^{r} \theta_{jj'} T_j(\tilde{\lambda}_{\mathrm{c}}) T_{j'}(\tilde{\lambda}_{\mathrm{r}})$$

where $\mathbf{\Theta} = (\theta_{jj'})$ is matrix of coefficients and $-1 \leq \tilde{\lambda}_{\mathrm{c}}, \tilde{\lambda}_{\mathrm{r}} \leq 1$



Examples of multi-graph spectral filters (shown is $|\tau(\tilde{\lambda}_{\mathrm{c}}, \tilde{\lambda}_{\mathrm{r}})|$)

- Multi-graph convolutional layer: Apply filter to matrix $\mathbf{X}$

$$\mathbf{Y} = \sum_{j,j'=1}^{r} \theta_{jj'} T_j(\tilde{\mathbf{\Delta}}_{\mathrm{c}}) \mathbf{X} T_{j'}(\tilde{\mathbf{\Delta}}_{\mathrm{r}})$$

where the scaled Laplacians $\tilde{\mathbf{\Delta}}_{\mathrm{c}} = 2\lambda_{\mathrm{c},n}^{-1} \mathbf{\Delta}_{\mathrm{c}} - \mathbf{I}$ and $\tilde{\mathbf{\Delta}}_{\mathrm{r}} = 2\lambda_{\mathrm{r},m}^{-1} \mathbf{\Delta}_{\mathrm{r}} - \mathbf{I}$

Multi-graph ConvNets[25]

- Multi-graph spectral filters

$$\mathbf{Y}_l^{(k)} = \xi\left(\sum_{l'=1}^{p}\sum_{j,j'=0}^{r}\theta_{jj'll'}T_j(\tilde{\mathbf{\Delta}}_{\mathrm{r}})\mathbf{Y}_{l'}^{(k-1)}T_{j'}(\tilde{\mathbf{\Delta}}_{\mathrm{c}})\right) \quad \begin{matrix} l=1,\dots,q \\ l'=1,\dots,p \end{matrix}$$

applied to p input channels (m x n matrices $\mathbf{Y}_1^{(k-1)},\dots,\mathbf{Y}_p^{(k-1)}$) and producing q output channels $\mathbf{Y}_1^{(k)},\dots,\mathbf{Y}_q^{(k)}$

☺ Filters have guaranteed r-hops support on both graphs

☺ O(1) parameters per layer

☹ O(nm) computational complexity (assuming sparse graphs)

[25] Monti, Bresson, Bronstein 2017

Factorized multi-graph ConvNets

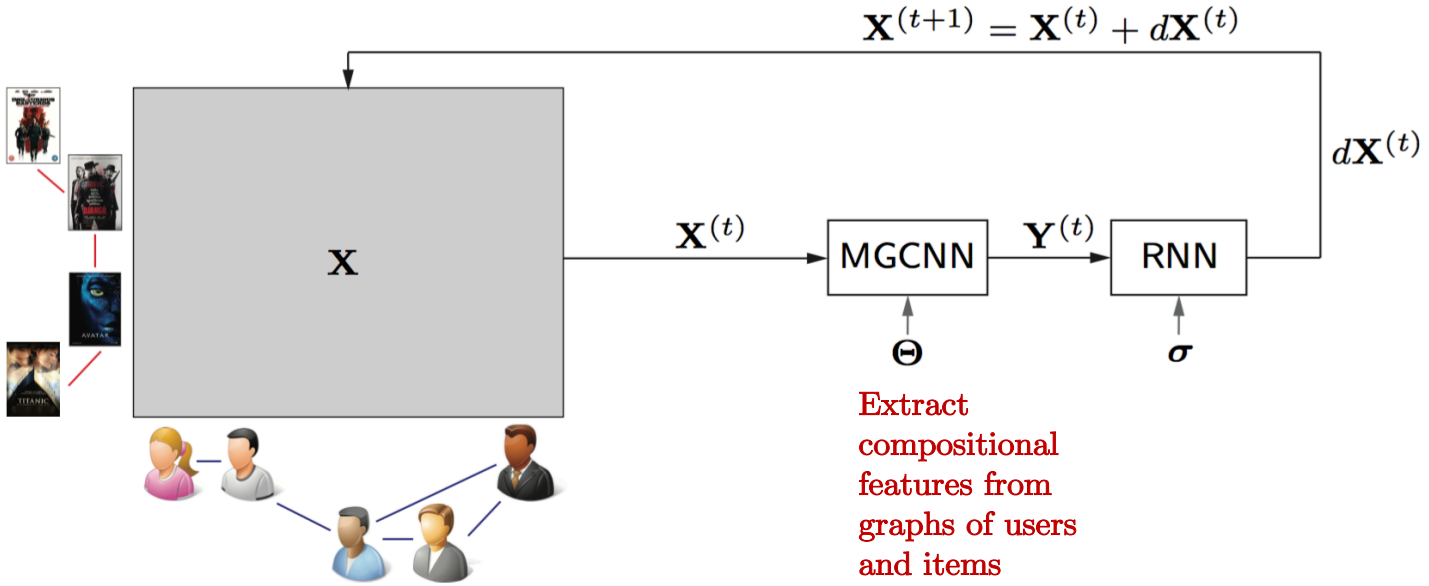
- Two spectral convolutional layers applied to each of the factors W, H

$$\mathbf{u}_l = \xi\left(\sum_{l'=1}^{p}\sum_{j=0}^{r}\theta_{\mathrm{r},jll'}T_j(\tilde{\mathbf{\Delta}}_{\mathrm{r}})\mathbf{w}_{l'}\right) \qquad \begin{array}{l} l=1,\dots,q \\ l'=1,\dots,p \end{array}$$

$$\mathbf{v}_l = \xi\left(\sum_{l'=1}^{p}\sum_{j'=0}^{r}\theta_{\mathrm{c},j'll'}T_{j'}(\tilde{\mathbf{\Delta}}_{\mathrm{c}})\mathbf{h}_{l'}\right)$$

- ☺ Filters have guaranteed r-hops support on both graphs
- ☺ O(1) parameters per layer
- ☺ O(n+m) computational complexity (assuming sparse graphs)
- ☹ May loose some coupling information

Matrix completion with Recurrent Multi-Graph ConvNets[25]



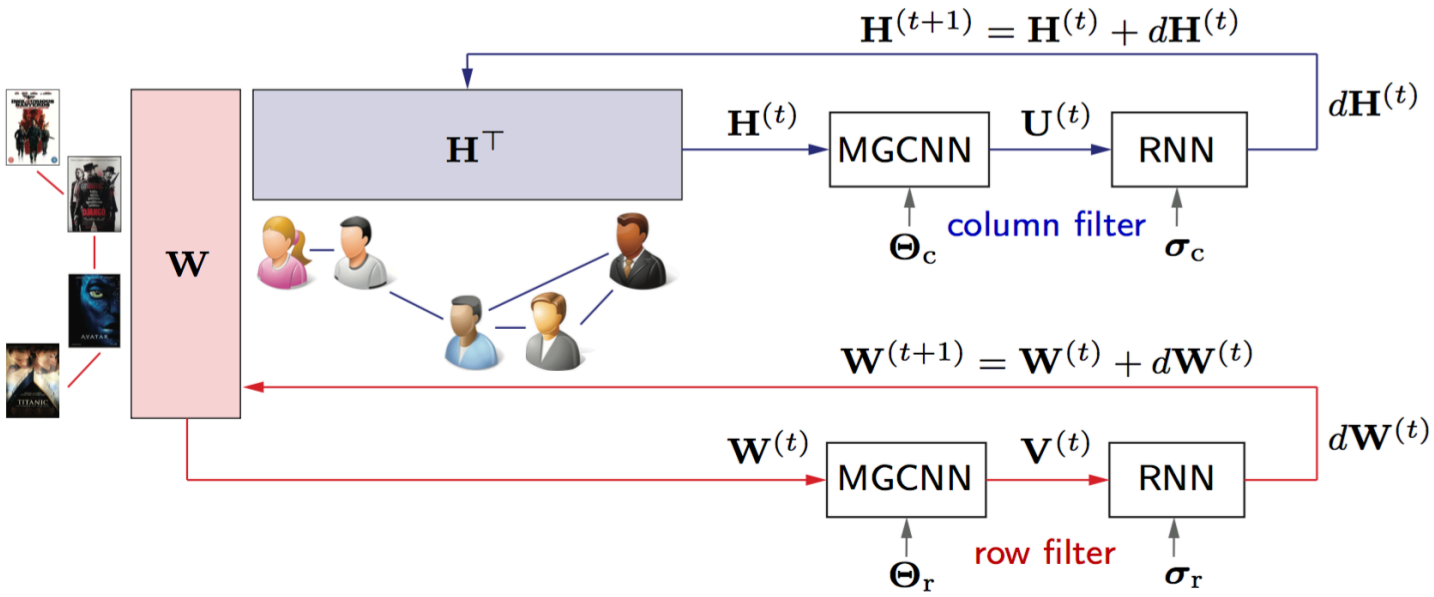
Recurrent multigraph CNN (RMCNN) architecture for matrix completion

$$\min_{\boldsymbol{\Theta},\boldsymbol{\sigma}} \|\mathbf{X}^{(T)}_{\boldsymbol{\Theta},\boldsymbol{\sigma}}\|^2_{\mathcal{G}_\mathrm{r}} + \|\mathbf{X}^{(T)}_{\boldsymbol{\Theta},\boldsymbol{\sigma}}\|^2_{\mathcal{G}_\mathrm{c}} + \frac{\mu}{2}\|\boldsymbol{\Omega} \circ (\mathbf{X}^{(T)}_{\boldsymbol{\Theta},\boldsymbol{\sigma}} - \mathbf{Y})\|^2_\mathrm{F}$$

Complexity: $\mathrm{O}(nm)$

[25] Monti, Bresson, Bronstein 2017

Factorized version



Separable recurrent multigraph CNN (sRMCNN) architecture
for matrix completion in factorized form

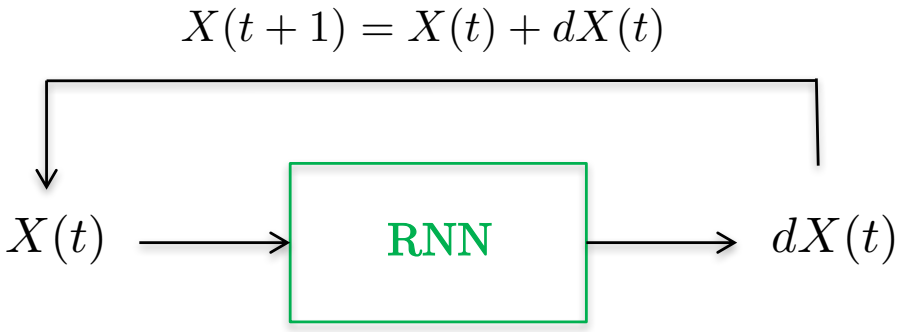
$$\min_{\boldsymbol{\theta}_\mathrm{r},\boldsymbol{\theta}_\mathrm{r},\boldsymbol{\sigma}_\mathrm{r},\boldsymbol{\sigma}_\mathrm{c}} \|\mathbf{W}^{(T)}_{\boldsymbol{\theta}_\mathrm{r},\boldsymbol{\sigma}_\mathrm{r}}\|^2_{\mathcal{G}_\mathrm{r}} + \|\mathbf{H}^{(T)}_{\boldsymbol{\theta}_\mathrm{c},\boldsymbol{\sigma}_\mathrm{c}}\|^2_{\mathcal{G}_\mathrm{c}} + \frac{\mu}{2}\|\boldsymbol{\Omega} \circ (\mathbf{W}^{(T)}_{\boldsymbol{\theta}_\mathrm{r},\boldsymbol{\sigma}_\mathrm{r}}(\mathbf{H}^{(T)}_{\boldsymbol{\theta}_\mathrm{c},\boldsymbol{\sigma}_\mathrm{c}})^\top - \mathbf{Y})\|^2_\mathrm{F}$$

Complexity: $\mathrm{O}(n+m)$

[25] Monti, Bresson, Bronstein 2017

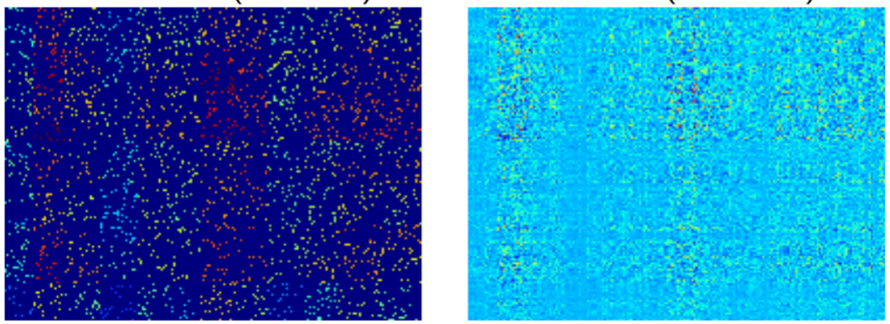
Incremental updates with RNNs

- The RNN/LSTM[26] cast the completion problem as non-linear diffusion process.
  - Replace multi-layer CNNs by simple RNN + diffusion
    ⇒ much less parameters to learn (reduce overfitting).
    ⇒ more favorable for highly sparse entries (better info diffusion).
  - Fix #parameters of the model independently of entry sparsity.
  - Allows to learn temporal dynamics of entries if such information is available.

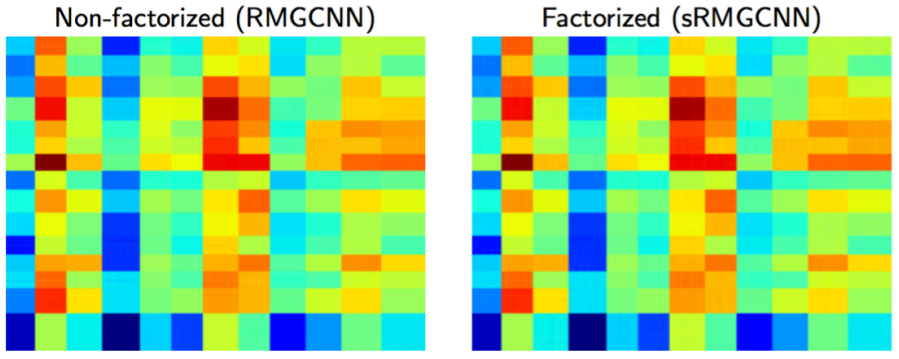


[26] Hochreiter, Schmidhuber 1997

Matrix completion methods comparison: synthetic



Matrix completion results on a synthetic dataset (initialization)



Matrix completion results on a synthetic dataset after $t = 10$ instances

Method	#Params	Complexity	RMSE
GMC[1]	mn	mn	0.3693
GRALS[2]	$m+n$	$m+n$	0.0114
sRMGCNN[3]	**1**	$\boldsymbol{m+n}$	**0.0106**
RMGCNN[3]	**1**	$\boldsymbol{mn}$	**0.0053**

Comparison of geometric matrix completion methods on synthetic data using both user+movie graphs

Methods: [1]Kalofolias et al. 2014; [2]Rao et al. 2015; [3]Monti, Bresson, Bronstein 2017

Matrix completion methods comparison: real

Method	MovieLens[1]	Flixster[2]	Douban[3]	Yahoo[4]
IMC[5]	1.653	–	–	–
GMC[6]	0.996	–	–	–
MC[7]	0.973	–	–	–
GRALS[8]	0.945	1.245	0.833	38.042
sRGCNN (Cheb)[9]	**0.929**	**0.926**	**0.801**	**22.415**
sRGCNN (Cayley)[10]	**0.922**	–	–	–

Accuracy (RMS error) of matrix completion methods on real data

Data: [1]Miller et al. 2003; [2]Jamali, Ester 2010; [3]Ma et al. 2011; [4]Dror et al. 2012
Methods: [5]Jain, Dhillon 2013; [6]Kalofolias et al. 2014; [7]Candès, Recht 2012; [8]Rao et al. 2015; [9]Monti, Bresson, Bronstein 2017; [10]Levie et al. 2017

Outline

- Deep Learning
  - *A brief history*
  - *High-dimensional learning*
- Convolutional Neural Networks
  - *Architecture*
  - *Non-Euclidean Data*
- Spectral Graph Theory
  - *Graphs and operators*
  - *Spectral decomposition*
  - *Fourier analysis*
  - *Convolution*
  - *Coarsening*
- Spectral ConvNets
  - *SplineNets*
  - *ChebNets* [NIPS'16]*
  - *GraphConvNets*
  - *CayleyNets**
- Spectral ConvNets on Multiple Graphs
  - *Recommender Systems* [NIPS'17]*
- **Conclusion**

Related works

- Classes of graph ConvNets:

  (1) Spectral approaches

  (2) Spatial approaches

- Spatial approaches
  - Geodesic CNN: Masci, Boscaini, Bronstein, Vandergheynst 2015
  - Anisotropic CNN: Boscaini, Masci, Rodola, Bronstein 2016
  - MoNet: Monti, Boscaini, Masci, Rodola, Svoboda, Bronstein 2017

- Graph Neural Networks:
  - Gori, Monfardini, Scarselli 2005
  - Li, Tarlow, Brockschmidt, Zemel 2015
  - Sukhbaatar, Szlam, Fergus 2016
  - Bruna, Li 2017

- Several other works

Conclusion

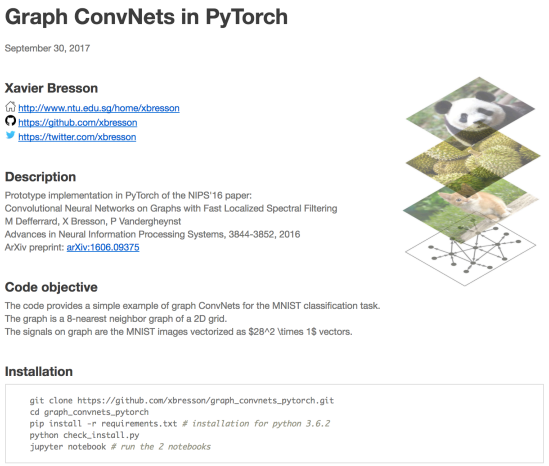
- Contributions:
  - Generalization of ConvNets to data on graphs (fixed)
  - Exact/tight localized filters on graphs
  - Linear complexity for sparse graphs
  - GPU implementation
  - Rich expressivity
  - Multiple graphs

- Several potential applications in network sciences.

Papers & code

- Papers
  - Convolutional neural networks on graphs with fast localized spectral filtering, M Defferrard, X Bresson, P Vandergheynst, NIPS, 2016, arXiv:1606.09375
  - Geometric matrix completion with recurrent multi-graph neural networks, F Monti, MM Bronstein, X Bresson, NIPS, 2017, arXiv:1704.06803
  - CayleyNets: Graph Convolutional Neural Networks with Complex Rational Spectral Filters, R Levie, F Monti, X Bresson, MM Bronstein, 2017, arXiv:1705.07664

- Code
  - https://github.com/xbresson/graph_convnets_pytorch



Tutorials

Computer Vision and Pattern Recognition (CVPR)

Geometric deep learning on graphs and manifolds

M. Bronstein, J. Bruna, A. Szlam, X. Bresson, Y. LeCun

July 21, 2017

Slides: https://www.dropbox.com/s/appafg1fumb6u7f/tutorial_CVPR17_DL_graphs.pdf?dl=0

Video will be posted on YouTube/ComputerVisionFoundation Videos

Neural Information Processing Systems (NIPS)

Geometric deep learning on graphs and manifolds

M. Bronstein, J. Bruna, A. Szlam, X. Bresson, Y. LeCun

Dec 4, 2017

Workshop

New Deep Learning Techniques

Feb 5-9, 2018

Organizers

Xavier Bresson, NTU

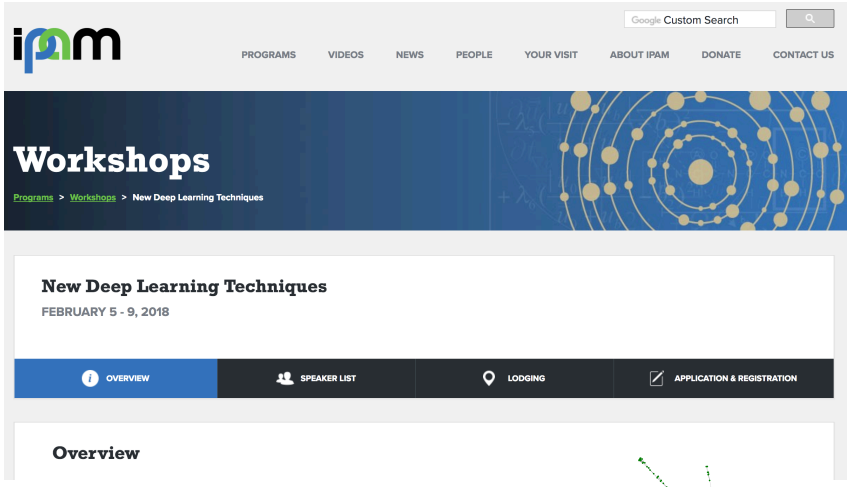
Michael Bronstein, USI/Intel

Joan Bruna, NYU/Berkeley

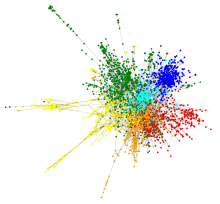
Yann LeCun, NYU/Facebook

Stanley Osher, UCLA

Arthur Szlam, Facebook



UCLA



http://www.ipam.ucla.edu/programs/workshops/new-deep-learning-techniques

Thank you

http://www.ntu.edu.sg/home/xbresson

https://github.com/xbresson

https://twitter.com/xbresson