CANDIDACY EXAM

Electrical Engineering Doctoral program (EDEE)

Deep Learning on Graphs

for Advanced Big Data Analysis

Student
Michaël Defferrard

Supervisor
Xavier Bresson

Advisor
Pierre Vandergheynst

EPFL LTS2 Laboratory

August 30, 2016

Introduction

- Objective: analyze and extract information for decision-making from large-scale and high-dimensional datasets
- Method: Deep Learning (DL), especially Convolutional Neural Networks (CNNs), on Graphs
- Fields: Deep Learning and Graph Signal Processing (GSP)

Motivation

- Important and growing class of data lies on irregular domains
  - Natural graphs / networks
  - Constructed (feature / data) graphs
- Modeling versatility: graphs model heterogeneous pairwise relationships
- Important problem: recent works, high demand
- Reproduce the breakthrough of DL beyond Computer Vision !

Problem

Formulate DL components on graphs (& discover alternatives)

Convolutional Neural Networks (CNNs)

- Localization: compact filters for low complexity
- Stationarity: translation invariance
- Compositionality: analysis with a filterbank

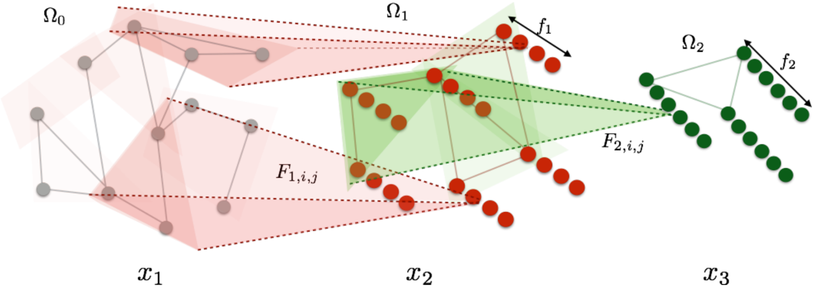
Challenges

- Generalize convolution, downsampling and pooling to graphs
- Evaluate the assumptions on graph signals

Local Receptive Fields

Gregor and LeCun 2010; Coates and Ng 2011; Bruna et al. 2013

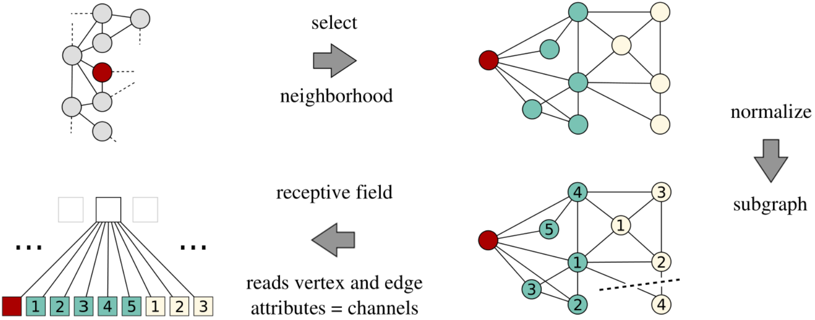
- Group features based upon similarity
  - Reduce the number of learned parameters
  - Can use graph adjacency matrix
- No weight-sharing / convolution / stationarity



Spatial approaches to Convolution on Graphs

Niepert, Ahmed, and Kutzkov 2016; Vialatte, Gripon, and Mercier 2016

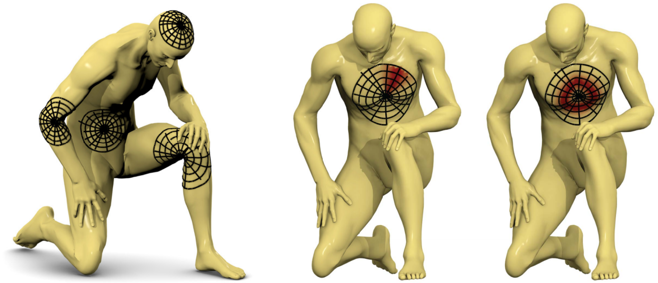
1. Define receptive field / neighborhood
2. Order nodes



Geodesic CNNs on Riemannian manifolds

Masci et al. 2015

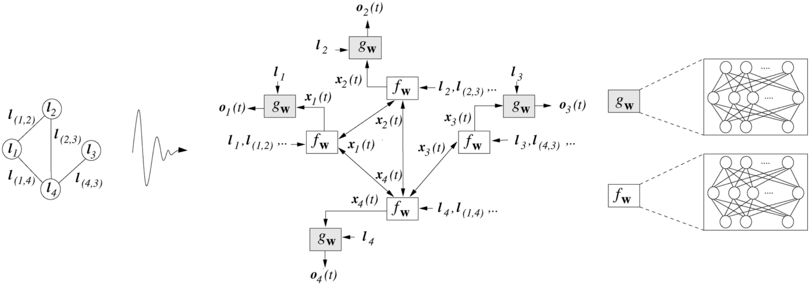
- Generalization of CNNs to non-Euclidean manifolds
- Local geodesic system of polar coordinates to extract patches
- Tailored for geometry analysis and processing



Graph Neural Networks (GNNs)

Scarselli et al. 2009

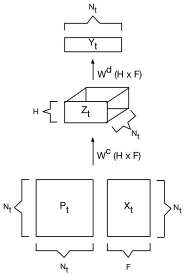
- Recurrent Neural Networks (RNNs) on Graphs
- Propagate node representations until convergence
- Representations used as features



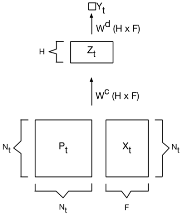
Diffusion-Convolutional Neural Networks (DCNNs)

Atwood and Towsley 2015

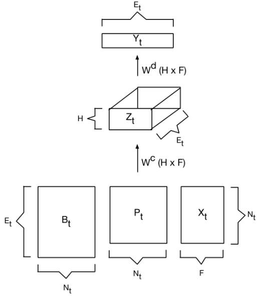
- Multiplication with powers (0 to H) of transition matrix
- Diffused features multiplied by weight vector of support H
- No pooling, followed by a fully connected layer



Node classification



Graph classification



Edge classification

Spectral Networks on Graphs

Bruna et al. 2013; Henaff, Bruna, and LeCun 2015

- First spectral definition
- Introduced a supervised graph estimation strategy
- Experiments on image recognition, text categorization and bioinformatics

- Spline filter parametrization
- Agglomerative method for coarsening

Further Work

Build on (Bruna et al. 2013) and (Henaff, Bruna, and LeCun 2015)

- Spectral formulation
- Computational complexity
- Localization
- Ad hoc coarsening & pooling

Performed Research

Proposed an efficient spectral generalization of CNNs to graphs

Main contributions

1. Spectral formulation
2. Strictly localized filters
3. Low computational complexity
4. Efficient pooling
5. Experimental results

Paper

"Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering" Defferrard, Bresson, and Vandergheynst 2016

- Accepted for publication at NIPS 2016
- Presented by Xavier at SUTD and University of Bergen

Peer Reviews

- "extend ... data driven, end-to-end learning with excellent learning complexity"
- "very clean, efficient parametrization [for] efficient learning and evaluation"
- "highly promising paper ... shows how to efficiently generalize the [convolution]"
- "the potential for significant impact is high"
- "new and upcoming area with only a few recent works"

Definitions

Chung 1997

- $\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$: undirected and connected graph
- $W \in \mathbb{R}^{n \times n}$: weighted adjacency matrix
- $D_{ii} = \sum_j W_{ij}$: diagonal degree matrix
- $x : \mathcal{V} \to \mathbb{R}$, $x \in \mathbb{R}^n$: graph signal
- $L = D - W \in \mathbb{R}^{n \times n}$: combinatorial graph Laplacian
- $L = I_n - D^{-1/2} W D^{-1/2}$: normalized graph Laplacian
- $L = U \Lambda U^T$, $U = [u_0, \ldots, u_{n-1}] \in \mathbb{R}^{n \times n}$: graph Fourier basis
- $\hat{x} = U^T x \in \mathbb{R}^n$: graph Fourier transform

Spectral Filtering of Graph Signals

$$y = g_\theta(L)x = g_\theta(U\Lambda U^T)x = Ug_\theta(\Lambda)U^T x$$

Non-parametric filter:

$$g_\theta(\Lambda) = \mathrm{diag}(\theta)$$

- Non-localized in vertex domain
- Learning complexity in $\mathcal{O}(n)$
- Computational complexity in $\mathcal{O}(n^2)$ (& memory)

Polynomial Parametrization for Localized Filters

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k$$

- Value at j of g_θ centered at i:
  $(g_\theta(L)\delta_i)_j = (g_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}$
- $d_{\mathcal{G}}(i,j) > K$ implies $(L^K)_{i,j} = 0$
  (Hammond, Vandergheynst, and Gribonval 2011, Lemma 5.2)

- K-localized
- Learning complexity in $\mathcal{O}(K)$
- Computational complexity in $\mathcal{O}(n^2)$

Recursive Formulation for Fast Filtering

$$g_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \quad \tilde{\Lambda} = 2\Lambda/\lambda_{max} - I_n$$

- Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$
- Filtering: $y = g_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x$
- Recurrence: $y = g_\theta(L)x = [\bar{x}_0, \ldots, \bar{x}_{K-1}]\theta$, $\bar{x}_k = T_k(\tilde{L})x = 2\tilde{L}\bar{x}_{k-1} - \bar{x}_{k-2}$ with $\bar{x}_0 = x$ and $\bar{x}_1 = \tilde{L}x$

- K-localized
- Learning complexity in $\mathcal{O}(K)$
- Computational complexity in $\mathcal{O}(K|\mathcal{E}|)$

Learning Filters

$$y_{s,j} = \sum_{i=1}^{F_{in}} g_{\theta_{i,j}}(L) x_{s,i} \in \mathbb{R}^n$$

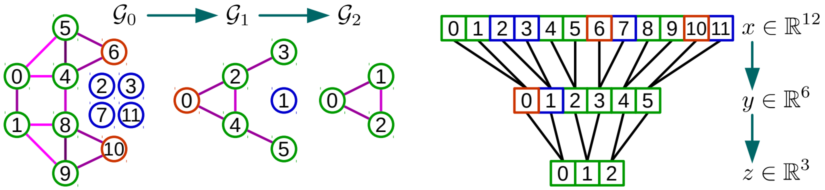
- $x_{s,i}$: feature map i of sample s
- $\theta_{i,j}$: trainable parameters
  ($F_{in} \times F_{out}$ vectors of Chebyshev coefficients)

Gradients for backpropagation:

- $\frac{\partial E}{\partial \theta_{i,j}} = \sum_{s=1}^{S} [\bar{x}_{s,i,0}, \ldots, \bar{x}_{s,i,K-1}]^T \frac{\partial E}{\partial y_{s,j}}$
- $\frac{\partial E}{\partial x_{s,i}} = \sum_{j=1}^{F_{out}} g_{\theta_{i,j}}(L) \frac{\partial E}{\partial y_{s,j}}$

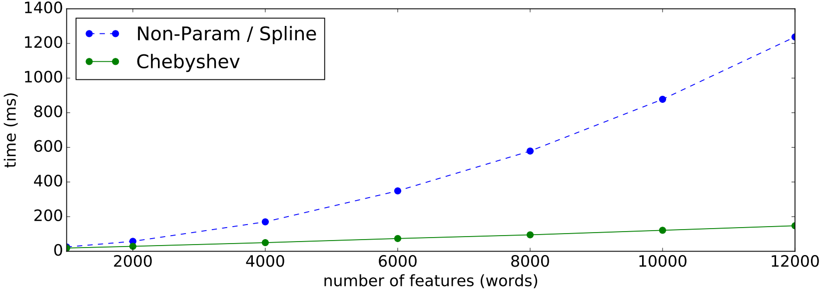
Overall cost of $\mathcal{O}(K|\mathcal{E}|F_{in}F_{out}S)$ operations

Coarsening & Pooling



- Coarsening: Graclus / Metis
  - Normalized cut minimization
- Pooling: as regular 1D signals
  - Satisfies parallel architectures like GPUs
- Activation: ReLU (or tanh, sigmoid)

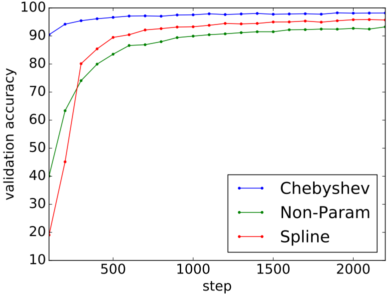
Training time (20NEWS)



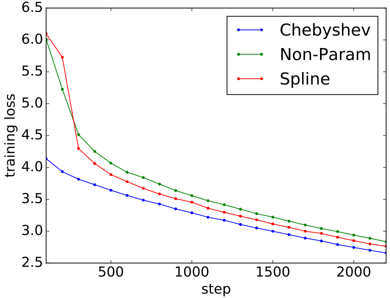
Make CNNs practical for graph signals !

Spline: $g_\theta(\Lambda) = B\theta$ (Bruna et al. 2013; Henaff, Bruna, and LeCun 2015)

Convergence (MNIST)



Validation accuracy



Training loss

Faster convergence !

Classification accuracy (MNIST)

Model	Architecture	Accuracy
Classical CNN	C32-P4-C64-P4-FC512	99.33
Proposed graph CNN	GC32-P4-GC64-P4-FC512	99.14

Table: Comparison to classical CNNs.

Comparable to classical CNNs and better than other parametrizations !

	Accuracy		
Architecture	Non-Param	Spline	Chebyshev
GC10	95.75	97.26	97.48
GC32-P4-GC64-P4-FC512	96.28	97.15	99.14

Table: Comparison between spectral filters, $K = 25$.

Further Research (1)

1. Numerical experiments on text documents
2. Alternative Parametrization
   - Polynomial of the Laplacian
   - Krylov subspace methods
3. Graph Coarsening
   - Contraction-based schemes
   - Kron reduction
   - Algebraic Multigrid methods (AMG)
   - Multi-level label propagation
   - Multi-level graph embedding
   - Spectral clustering

Further Research (2)

4. Local Stationarity: verify the statistical assumptions
5. Initialization & Optimization
6. Filter Transfer
7. Anisotropic Filters
8. Supervised Graph Estimation
9. Time-varying Data
10. Comparison of all methods
11. Applications
    - Rotation invariance for Computer Vision
    - Topic Categorization on Wikipedia
    - Collaborate for social & biological sciences

Thanks

Feedbacks? Questions?