Deep Learning on Graphs
Learning beyond Euclidean Data

Michaël Defferrard
École Polytechnique Fédérale de Lausanne (EPFL)

Joint work with
Youngjoo Seo (EPFL)
Xavier Bresson (NTU)
Pierre Vandergheynst (EPFL)
Structured data

Majority of data is naturally unstructured. But can be structured by graphs.

Why structure data?

- To incorporate additional information.
- To exploit spatial correlations.
- To decrease learning complexity by making geometric assumptions.

Data structured by Euclidean grids.

- 1D: sound, time-series.
- 2D: images.
- 3D: video, hyper-spectral images.
Non-Euclidean data: natural graphs

Modeling versatility: graphs model heterogeneous pairwise relationships.

Examples of irregular / graph-structured data:

- Social networks: Facebook, Twitter.
- Biological networks: genes, molecules, brain connectivity.
- Infrastructure networks: energy, transportation, Internet, telephony.
Non-Euclidean data: constructed graphs

Sample graph

- Semi-supervised learning.
- Incorporate external information.

Feature graph

- Reduce computations.
- Incorporate external information.

Problems: signals, nodes or graphs classification (regression).
Using the structure

Extrinsic: embed the graph in an Euclidean space.
  - Each node is represented by a vector.
  - Use that embedding as additional features for a fully connected NN.
  - Use a convolutional NN in the embedding space.
    Possibly very high-dimensional!

Intrinsic: a Neural Net defined on graphically structured data.
  - Exploit geometric structure for computational efficiency.
  - Starting point: ConvNet, an intrinsic formulation for Euclidean grids.
ConvNets are ubiquitous
LeCun, Bengio, and Hinton 2015

First developed for Computer Vision
- Object recognition
- Image captioning
- Image inpainting

Spreading outside CV
- Natural language processing
- Audio: sound & voice
- Autonomous agents (playing Atari or Go)
Why are ConvNets good?

ConvNets are extremely efficient at extracting meaningful statistical patterns in large-scale and high-dimensional datasets.

They exploit the underlying geometric structure in the data.

Statistical assumptions

- **Localization**: compact filters for low complexity.
- **Stationarity**: translation invariance.
- **Compositionality**: analysis with a filterbank.
- **Multi-scale**: hierarchical features extracted by multiple layers.
ConvNets: architecture

Ingredients

1. Convolution (local)
2. Non-linearity (point-wise)
3. Down-sampling (global / local)
4. Pooling (local)
ConvNets: feature extraction
Zeiler and Fergus 2014

Figure: Features extracted from ImageNet.
Developed for data lying on Euclidean grids

All operations are well defined and computationally efficient:

1. Convolution $\rightarrow$ filter translation or fast Fourier transform (FFT).
2. Down-sampling $\rightarrow$ pick one pixel out of $n$.
3. Non-linearity $\rightarrow$ point-wise operation.
4. Pooling $\rightarrow$ summarize the receptive field.

Image (2D)  Video (3D)  Sound (1D)
ConvNets on graphs

Graphs vs Euclidean grids

- Irregular sampling.
- Weighted edges.
- No orientation (in general).

Challenges

1. Formulate convolution and down-sampling on graphs.
2. Make them efficient!
1. Define receptive field / neighborhood.
2. Order nodes, i.e. give an orientation.
ConvNets on graphs: spectral approach
Bruna, Zaremba, Szlam, and LeCun 2014; Henaff, Bruna, and LeCun 2015

- Spectral graph theory for convolution on graphs.
- Balanced cut model for graph coarsening (sub-sampling).
Definitions: graph
Chung 1997

\( G = (\mathcal{V}, \mathcal{E}, W) \): undirected and connected graph

- \( \mathcal{V} \): set of \( |\mathcal{V}| = n \) vertices
- \( \mathcal{E} \): set of edges
- \( W \in \mathbb{R}^{n \times n} \): weighted adjacency matrix
- \( D_{ii} = \sum_{j} W_{ij} \): diagonal degree matrix

Graph Laplacians (core operator to spectral graph theory):
- \( L = D - W \in \mathbb{R}^{n \times n} \): combinatorial
- \( L = I_n - D^{-1/2}WD^{-1/2} \): normalized
Definitions: graph Fourier transform
Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

$L$ is symmetric and positive semidefinite $\rightarrow L = U\Lambda U^T$ (EVD)

- Graph Fourier basis $U = [u_0, \ldots, u_{n-1}] \in \mathbb{R}^{n \times n}$

- Graph “frequencies” $\Lambda = \begin{bmatrix} \lambda_1 & 0 \\ \vdots & \ddots \\ 0 & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n}$

Graph Fourier Transform
1. Graph signal $x: \mathcal{V} \rightarrow \mathbb{R}$ seen as $x \in \mathbb{R}^n$
2. Transform: $\hat{x} = \mathcal{F}_G \{x\} = U^T x \in \mathbb{R}^n$
3. Inverse: $x = U\hat{x} = UU^T x = x$
Definitions: convolution on graphs
Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

Convolution theorem:

\[ x \ast_g g = U (U^T g \odot U^T x) \]
\[ = U (\hat{g} \odot U^T x) \]

Conveniently written as:

\[ x \ast_g g = U \begin{bmatrix} \hat{g}(\lambda_1) & 0 \\ \vdots & \ddots \\ 0 & \hat{g}(\lambda_n) \end{bmatrix} U^T x \]
\[ = U \hat{g}(\Lambda) U^T x \]
\[ = \hat{g}(L)x \]
Spectral filtering of graph signals

\[ y = \hat{g}_\theta(L)x = U\hat{g}_\theta(\Lambda)U^T x \]

Non-parametric filter:

\[ \hat{g}_\theta(\Lambda) = \text{diag}(\theta), \quad \theta \in \mathbb{R}^n \]

- Non-localized in vertex domain
- Learning complexity in \( \mathcal{O}(n) \)
- Computational complexity in \( \mathcal{O}(n^2) \) (\& memory)
Polynomial parametrization for localized filters
Shuman, Ricaud, and Vandergheynst 2016

\[ \hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k \Lambda^k, \quad \theta \in \mathbb{R}^K \]

- Value at \( j \) of \( g_\theta \) centered at \( i \): \((\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j} = \sum_k \theta_k (L^k)_{i,j}\)
- \( d_G(i, j) > K \) 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 \( \mathcal{O}(n^2) \)
Filter localization

Shuman, Ricaud, and Vandergheynst 2016

**Figure:** Localization on regular Euclidean grid.

**Figure:** Localization on graph with $(\hat{g}_\theta(L)\delta_i)_j = (\hat{g}_\theta(L))_{i,j}$. 
Recursive formulation for fast filtering
Hammond, Vandergeynst, and Gribonval 2011

\[ \hat{g}_\theta(\Lambda) = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{\Lambda}), \quad \tilde{\Lambda} = 2\Lambda/\lambda_{\text{max}} - I_n \]

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

- \( K \)-localized
- Learning complexity in \( \mathcal{O}(K) \)
- Computational complexity in \( \mathcal{O}(K|\mathcal{E}|) \) (same as classical ConvNets!)
Learning filters
Defferrard, Bresson, and Vandergheynst 2016

\[ y_{s,j} = \sum_{i=1}^{F_{in}} \hat{g}_{\theta_i,j}(L)x_{s,i} \in \mathbb{R}^n \]

- \( x_{s,i} \): 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
Defferrard, Bresson, and Vandergheynst 2016

- **Coarsening**: Graclus / Metis
  - Greedy node merging.
  - Very fast!

- **Pooling**: as regular 1D signals
  - Binary tree structured coarsened graphs.
  - Satisfies parallel architectures like GPUs.

- **Activation**: ReLU, LeakyReLU, maxout, tanh, sigmoid.
Graph ConvNet architecture
Defferrard, Bresson, and Vandergheynst 2016

Input graph signals
e.g. bags of words

Feature extraction
Convolutional layers

Classification
Fully connected layers

Output signals
e.g. labels

Graph signal filtering
1. Convolution
2. Non-linear activation

Graph coarsening
3. Sub-sampling
4. Pooling
Applications

- Semi-supervised learning
  [Kipf and Welling 2016; Manessi, Rozza, and Manzo 2017]
- Quantum Chemistry
  [Duvenaud et al. 2015; Gilmer, Schoenholz, Riley, Vinyals, and Dahl 2017]
- High Energy Physics
- Community detection [Bruna and Li 2017]
- Brain analysis
  [Ktena et al. 2017; Parisot et al. 2017; Anirudh and Thiagarajan 2017]
- Matrix completion for recommendation
  [Monti, Bronstein, and Bresson 2017]
- Neural machine translation
  [Bastings, Titov, Aziz, Marcheggiani, and Sima’an 2017]
- Link prediction and entity classification in knowledge bases
  [Schlichtkrull et al. 2017]
Time Series

- **Sensors**: temperature, wind, pressure, body signals, etc.
- **Stock market**
- **Text** (series of discrete symbols, i.e. words)
- **Network activity**: energy, transportation, communication, brain
Recurrent Neural Networks & LSTM

Figures by Colah, 2015
Recurrent Graph Convolutional Network
Seo, Defferrard, Bresson, and Vandergheynst 2016

1D signals

- $h_t = \tanh(W_x x_t + W_h h_{t-1})$
- $y_t = W h_t$
- State stored in hidden units

Graph signals

- $h_t = \tanh(W_x \ast_G x_t + W_h \ast_G h_{t-1})$
- $y_t = W \ast_G h_t$
- State stored locally on the nodes

- Graph filtering $x$ as $y = [\bar{x}_0, \ldots, \bar{x}_{K-1}] \theta$ is a weighted sum of diffused versions $\bar{x}$ of $x$.
- Data exchanged locally around the $K$-neighborhood.
- Reduces to independent signals if $K = 1$ or graph has no edge.
Real data: Wikipedia

Goal: structured times series forecasting

- Anomaly / event detection
- Regulation & Control
- Generative process understanding

Wikipedia network & signals

- Nodes: articles
- Edges: hyper-links
- Signals: number of hits per hour
Structured Time Series

Charlie_Hebdo (2251390)

Charlie_Hebdo_shooting (44969225)
Real data (english wikipedia) is massive and hard
- 4.9M nodes
- 294M edges
- 6k time samples
- 760 GiB raw data
- dynamic: edges and nodes are added and removed

Example pruning: nodes with mean activity higher than 100 per hour
- 10k nodes
- 560k edges
- 6k time samples
- static: assume last hyper-link graph
Conclusion

- Graph are versatile tools to structure real data.
- Neural networks are the most effective ML algorithm today.

**Deep Learning is coming to Graph Signal Processing**

Further research

- Transfer between graphs / dynamic graphs
- Combine time & vertex domains with a joint transform
- Multi-scale approaches: both in time and vertex


Code: https://github.com/mdeff/cnn_graph

Thanks Questions?

PS: next year, we’re organizing the GSP workshop at EPFL!
SUPERVISED CLASSIFICATION OF NEUROIMAGING DATA USING GRAPH SIGNAL PROCESSING

Mathilde Ménoret
Nicolas Farrugia
Bastien Pasdeloup
Vincent Gripon

CMU Workshop on Graph Signal Processing

June 1st 2017
The application of Graph Theory for neuroimaging is now widespread in the neuroimaging community.
Laplacian eigenvectors of a graph based on brain structure (gray and white matter) can predict spontaneous brain activity (Atasoy, Donelly & Pearson, 2017, Nature communications)
Laplacian eigenvectors can be interpreted functionally (Margulies et al. 2016, PNAS, here termed “connectivity gradients”) Therefore GSP appears as an ideal framework to analyse, predict and interpret brain activity.
Graph Frequencies + PCA (Huang et al. 2016): statistical analysis on a motor learning task and fMRI data.

Dimensionality reduction (Rui et al. 2016) by learning subspaces on graph frequencies on MEG data.

Denoising and dimensionality reduction on EEG data using graph laplacian – SPHARA (Graichen et al. 2015)

Low rank estimation / denoising (Liu et al. 2016)

Measure derived from graph signal smoothness combined with graph modularity (Modular Dirichlet Energy) to analyse an EEG experiment (Smith et al. 2017)
OUR CONTRIBUTION

► Goal: use GSP for supervised classification of brain activity (fMRI)

► Key Questions
  - Can GSP improve classification accuracy?
  - What is the influence of the graph?
  - Can GSP be exploited for dimensionality reduction in the context of brain data?

► Experiments
  - Simulated fMRI data
  - Haxby dataset

All results are presented here using classification with linear support vector machines.

This work is currently available as preprint:
Menoret, Farrugia, Pasdeloup & Gripon
Brain parcellation to create ROIs (clusters)
Geometry : baricenter of ROIs
We simulated the activation of 6 areas as spheres whose center corresponded to the MNI coordinates of brain areas known to be involved in visual processing of Houses and Face:

- L/R Primary Visual Cortex V1 (similar activation for both conditions),
- L/R Fusiform Face Area (stronger activation for faces)
- L/R Parahippocampal Place area (stronger activation for houses).

Template MNI / Baseline data: averaged “rest” from Haxby
GSP using 6 types of graph

Geometric graphs:
- Fully connected geometric graph: \( w_{i,j} = \text{distance}(i,j) \)
- Geometric:
  \[
  \text{if distance}(i,j) < \alpha: \quad w_{i,j} = \text{distance}(i,j) \\
  \text{else} \quad w_{i,j} = 0
  \]

Functional graphs \textbf{(based on rest data)}:
- Covariance matrix \( w_{i,j} = \text{cov}(i,j) \)
- Kalofolias method (estimates graph on which observed signals are smooth)

Mixed functional and geometric graphs:
- Semilocal:
  \[
  \text{if distance}(i,j) < \alpha: \quad w_{i,j} = \text{cov}(i,j) \\
  \text{else} \quad w_{i,j} = 0
  \]
- Graph associating both distance between brain areas and connectivity measures (« fundis »):
  \[
  w_{i,j} = \exp\left(-\frac{1 - \| \text{correlation}(i,j) \|}{2\theta}\right) \cdot \exp\left(-\frac{\text{distance}(i,j)^2}{2\sigma}\right)
  \]
Dimensionality reduction:

- Principal Component Analysis
- Independent Component Analysis
- **Graph sampling (GS)**: adapted from Puy et al. 2016: calculate the graph weighted coherence for a band frequency of interest (high pass, low pass) and extract the k nodes where the signal energy is the most concentrated
- ANOVA- select k-best:
  - fMRI data
  - Data projected in the graph space (Graph Fourier Transform)
RESULTS OVERVIEW – SIMULATED DATA

<table>
<thead>
<tr>
<th>Graph Types</th>
<th>GFT LF</th>
<th>GFT HF</th>
<th>GFT ANOVA</th>
<th>GS LF</th>
<th>GS HF</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Difficult</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full</td>
<td>54.8%</td>
<td>51.1%</td>
<td>66.0%</td>
<td>52.0%</td>
<td>51.3%</td>
</tr>
<tr>
<td>Geometric</td>
<td>56.7%</td>
<td>64.8%</td>
<td>64.8%</td>
<td>50.5%</td>
<td>65.2%</td>
</tr>
<tr>
<td>lCorrelation</td>
<td>52.4%</td>
<td>66.8%</td>
<td>64.7%</td>
<td>50.9%</td>
<td>60.3%</td>
</tr>
<tr>
<td>lCovariance</td>
<td>52.4%</td>
<td>67.6%</td>
<td>65.2%</td>
<td>51.2%</td>
<td>66.2%</td>
</tr>
<tr>
<td>Kalofolias</td>
<td>61.6%</td>
<td>51.9%</td>
<td>65.9%</td>
<td>61.6%</td>
<td>51.9%</td>
</tr>
<tr>
<td>Semilocal</td>
<td>53.8%</td>
<td>69.5%</td>
<td>65.6%</td>
<td>50.3%</td>
<td>72.5%*</td>
</tr>
<tr>
<td>Fundis</td>
<td>54.9%</td>
<td>64.2%</td>
<td>65.1%</td>
<td>49.7%</td>
<td>62.8%</td>
</tr>
<tr>
<td><strong>Easy</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Full</td>
<td>65.1%</td>
<td>60.0%</td>
<td>88.9%</td>
<td>49.6%</td>
<td>60.6%</td>
</tr>
<tr>
<td>Geometric</td>
<td>71.3%</td>
<td>79.4%</td>
<td>86.0%</td>
<td>57.8%</td>
<td>79.1%</td>
</tr>
<tr>
<td>lCorrelation</td>
<td>59.5%</td>
<td>86.5%</td>
<td>86.9%</td>
<td>53.5%</td>
<td>75.2%</td>
</tr>
<tr>
<td>lCovariance</td>
<td>57.2%</td>
<td>87.0%</td>
<td>88.8%</td>
<td>52.8%</td>
<td>84.8%</td>
</tr>
<tr>
<td>Kalofolias</td>
<td>88.2%</td>
<td>54.2%</td>
<td>89.4%</td>
<td>87.5%</td>
<td>52.6%</td>
</tr>
<tr>
<td>Semilocal</td>
<td>61.3%</td>
<td>87.9%</td>
<td>88.6%</td>
<td>52.3%</td>
<td>90.9%*</td>
</tr>
<tr>
<td>Fundis</td>
<td>67.4%</td>
<td>77.5%</td>
<td>86.7%</td>
<td>52.4%</td>
<td>77.5%</td>
</tr>
</tbody>
</table>

-> next results are shown only for the semilocal graph
RESULTS SIMULATION – SEMILOCAL GRAPH

Classification accuracy

Keeping 50 dimensions
Fig. 1. Classification performance for the *Semilocal* graph depending on the number of dimensions. Comparison of the graph sampling method (gray) and the graph K-best (black) for the two groups: Easy and Difficult.
RESULTS USING SIMULATED DATA

Contrast Condition 1 – Condition 2

SVM weights after GS

SVM weight after ANOVA feature selection
TABLE II

COMPARISON OF GRAPH SAMPLING (SEMILOCAL GRAPH), PCA, ICA AND ANOVA. CLASSIFICATION ACCURACY WITH 50 COMPONENTS FOR THE SIMULATED AND HAXBY DATASETS.

<table>
<thead>
<tr>
<th>Method</th>
<th>Simulation</th>
<th></th>
<th>Haxby</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Easy</td>
<td>Difficult</td>
<td>Face-House</td>
<td>Cat-Face</td>
</tr>
<tr>
<td>PCA</td>
<td>88.8%</td>
<td>65.5%</td>
<td>82.7%</td>
<td>63.6%</td>
</tr>
<tr>
<td>ICA</td>
<td>90.2%</td>
<td>65.3%</td>
<td>84.4%</td>
<td>67.0%</td>
</tr>
<tr>
<td>ANOVA</td>
<td>92.1%</td>
<td>67.3%</td>
<td>85.5%</td>
<td>65.5%</td>
</tr>
<tr>
<td>Graph sampling</td>
<td>90.9%</td>
<td>72.5%</td>
<td>88.2%</td>
<td>69.0%</td>
</tr>
</tbody>
</table>
RESULTS ON HAXBY DATASET

Dim: 50

Contrast House-Face

SVM weight after Graph Sampling (Semilocal)
Dimensionality reduction combined with GSP is a promising avenue

Classification
- On simulated data, GSP using semi-local graph and graph sampling yields significant performance improvements in a difficult scenario
- On a real dataset: when keeping few dimensions, GSP + semilocal and graph sampling may improve classification accuracy

Perspectives
- Use a structural graph from dMRI?
- Role of classification technique? (linear SVC used so far)
- Optimal transport using Riemannian geometry?
Coded computation for speeding up graph analyses

Yaoqing Yang, Pulkit Grover, and Soummya Kar

ECE, CMU
Paper appears on arXiv tomorrow
Motivation: GSP is computationally expensive!

- Graph sizes are large.
  - Facebook: $10^9$ nodes, $10^{12}$ edges.
  - Human brain connectome: $10^{11}$ nodes, $10^{14}$ edges.
  - Google searches $10^{13}$ webpages, $10^{11}$ times per month (2013).
- People & systems are impatient.
  - We have computational deadlines!
- Saturation of Moore’s law $\Rightarrow$ massive parallelization
Motivation: GSP is computationally expensive!

- Graph sizes are large.
  - Facebook: $10^9$ nodes, $10^{12}$ edges.
  - Human brain connectome: $10^{11}$ nodes, $10^{14}$ edges.
  - Google searches $10^{13}$ webpages, $10^{11}$ times per month (2013).
- People & systems are impatient.
  - We have computational deadlines!
- Saturation of Moore’s law $\Rightarrow$ massive parallelization
- This talk: how to address unpredictable variability of “workers” in highly parallel graph analyses.
- Main message: error-correcting codes $>$ replication.
  ... shown theoretically, and experimentally.
Some processors unpredictably slower than others.
A core problem in parallelization: how to deal with this unpredictable variability?
Replication: a strategy to address stragglng

Is replication optimal use of redundant processors? What if straggling processor not known in advance?
Classical coded computation

Encoder

r₁ → r₁, r₂ → r₂ → r₁+r₂ → Processor1 → X₁*, Processor2 (slow) → X₂* → Processor3 → X₁*+X₂* → Decoder

wait for two fast processors
Classical coded computation

Variations of coded computing makes several problems resilient to faults/straggling:

- Matrix-vector/matrix-matrix products [Huang, Abraham'84][Pippenger'85]...[Yang, Grover, Kar’14,’16][Lee et al.’16][Dutta, Cadambe, Grover’16][Tandon et al.’16][Reisizadehmobarakeh et al. ’17]
- Logistic regression [Yang, Grover, Kar’16]
- Convolutions [Yang, Grover, Kar’16][Dutta, Cadambe, Grover’17]
- ... graph analyses?
What graph analyses could coding speed up?

Coding works well with algorithms that rely on linear computations

► Personalized PageRank
► Graph Signal Sampling and Recovery [Narang, Gadde, Sanou, Ortega ’13][Chen, Varma, Sandryhaila, Kovacevic ’15][Marques, Segarra, Leus, Ribeiro ’16]
► Graph filtering

Graph analyses are a new class of problems for coded computing: they tend to iteratively converge to the actual solution.
Coded graph analytics through an example problem

- Personalized PageRank, Random Walk with Restart [Haveliwala ’02]

\[ x = (1-d)Ax + dr \]

- For simplicity, we focus on power-iteration method [Page, Brin, Motwani, Winograd ’99]

\[ x^{(l+1)} = (1 - d)Ax^{(l)} + dr. \]
Coded graph analytics through an example problem

- Personalized PageRank, Random Walk with Restart [Haveliwala ’02]

\[ x = (1-d)Ax + dr \]

- For simplicity, we focus on power-iteration method [Page, Brin, Motwani, Winograd ’99]

\[ x^{(l+1)} = (1 - d)Ax^{(l)} + dr. \]

- **Problem formulation**: Solve \( k \) personalized PageRank problems using \( P > k \) processors.
Power-iterations converge to PageRank solution

\[ x^{(l+1)} = (1 - d)Ax^{(l)} + dr. \]

Converges to \( x^* \) satisfying \( x^* = (1 - d)Ax^* + dr. \)

Subtracting, \( e^{(l+1)} = (1 - d)Ae^{(l)} \), where \( e^{(l)} = x^{(l)} - x^* \).
Classical coded computation applied to personalized PageRank

- Initialize (Encoding)

\[ [s_1, \ldots, s_P] = [r_1, \ldots, r_k] \cdot G_{k \times P}. \]

- Parallel Computing:
  \( l_i \) power iterations at the \( i \)-th worker with input \( s_i \)

\[ Y_{N \times P}^{(T_{dl})} = [y_1^{(l_1)}, \ldots, y_P^{(l_P)}]. \]

- Post Processing (Decoding) Matrix inversion on fastest \( k \) processors.

\[ \hat{X}^\top = \tilde{G}^{-1}(Y^{(T_{dl})})^\top. \]
Classical coded computation on for personalized PageRank: errors blow up!

Decoding: $\hat{X}^T = \tilde{G}^{-1}(Y^{(T_{dl})})^T$.

- e.g. 120 processors; 100 PageRank problems.
- Decode using fastest 100 processors.
- Decoding matrix is ill-conditioned w.h.p. $\Rightarrow$ errors are blown up at small deadlines!
Addressing error-blowup issue in coded computation

Main idea: Gap from convergence can be viewed as “additive noise”

Classical coded computation

Proposed coded method
Proposed algorithm: weighted combination of processor outputs

- Initialize (Encoding)

\[
[s_1, \ldots, s_P] = [r_1, \ldots, r_k] \cdot G.
\]

- Parallel Computing:
  \( l_i \) power iterations at the \( i \)-th worker with input \( s_i \)

\[
Y_{N \times P}^{(T_{dl})} = \begin{bmatrix} y_1^{(l_1)} & \cdots & y_P^{(l_P)} \end{bmatrix}.
\]

- Post Processing (Decoding)

\[
\hat{X}^\top = (G\Lambda^{-1}G^\top)^{-1}G\Lambda^{-1}(Y^{(T_{dl})})^\top.
\]

Similar to the “weighted least-square” solution.
The diagonal weight matrix $\Lambda$

$$\Lambda = \text{diag}[\text{trace}(C(l_1)), \ldots, \text{trace}(C(l_P))],$$

where the matrices $C(l_i), i = 1, \ldots, P$ are defined as

$$C(l_i) = B^{l_i} C_E (B^T)^{l_i}.$$

- $l_i = \text{number of iterations at the } i\text{-th processor}$
- $B = \text{“contraction” matrix in the iterative algorithm}$
- $C_E = \text{error covariance at the 0-th iteration}$
- $\text{trace}(C(l_i)) = \text{MSE at the } i\text{-th processor}$
Coded computing provably outperforms repetition

For iid speeds of computation:

**Theorem 1**: For fixed deadline \( T_{dl} \), as \( P \to \infty \),

\[ \text{Coded MSE} < \text{Replication MSE} \]

as long as the squared error at each processor satisfies a heavy-tail distribution property (with randomness arising from number of iterations).

**Theorem 2**: For fixed \( P \), as \( T_{dl} \to \infty \), the error exponents satisfy \textbf{coded} \( \gtrsim \) \textbf{replication} = \textbf{uncoded}, i.e., \textbf{the coded PageRank error drops fastest}.

\[ ErrExp := \lim_{T_{dl} \to \infty} -\frac{1}{T_{dl}} \log MSE(T_{dl}) \]
Experiments on CMU Clusters: coded PageRank has a higher error-exponent

Coding beats competing strategies by a factor of $10^4$!

Parameters: $P=120$ processors for $k=100$ PageRank problems.
Google plus graph: $N=107,614$ nodes, 13,673,453 edges.
Twitter graph: $N=81,306$ nodes, 1,768,149 edges.
Complexity analysis: Encoding/decoding complexity negligible compared to per-processor computation

- Encoding/decoding complexity \( O(PkN) \)
- Parallel computing \( O(N^\alpha l) \) with \( \alpha \approx 1.68 \) for social network [Leskovec, Kleinberg, Faloutsos, ’05]
- Encoding/decoding < per-processor computing (\( N \) is large and \( k, P \) are small)
- Communication cost is small \( O(N) \) per-processor
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.

- Coding speeds up (linear) GSP on multiple processors
  - outperforms replication by an arbitrarily large factor.
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.
- Coding speeds up (linear) GSP on multiple processors
  - outperforms replication by an arbitrarily large factor.
- GSP introduces a novel issue in coded computation
  - computation slowly converges to the solution
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.

- Coding speeds up (linear) GSP on multiple processors
  - outperforms replication by an arbitrarily large factor.
- GSP introduces a novel issue in coded computation
  - computation slowly converges to the solution
- Technique presented for PageRank, but extends to linear inverse problems, graph filtering, interpolation, etc.
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.

- Coding speeds up (linear) GSP on multiple processors
  - outperforms replication by an arbitrarily large factor.
- GSP introduces a novel issue in coded computation
  - computation slowly converges to the solution
- Technique presented for PageRank, but extends to linear inverse problems, graph filtering, interpolation, etc.
- Ongoing work: parallelizing single graph computation on multiple processors. Needs deeper understanding of communication overhead.
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.

- Coding speeds up (linear) GSP on multiple processors
  - outperforms replication by an arbitrarily large factor.
- GSP introduces a novel issue in coded computation
  - computation slowly converges to the solution
- Technique presented for PageRank, but extends to linear inverse problems, graph filtering, interpolation, etc.
- Ongoing work: parallelizing single graph computation on multiple processors. Needs deeper understanding of communication overhead.

Future work:
Do you have a graph problem that needs to be processed faster?
Summary and ongoing work

Analogy with comm systems: diversity improves resilience.

- Coding speeds up (linear) GSP on multiple processors
  - outperforms replication by an arbitrarily large factor.
- GSP introduces a novel issue in coded computation
  - computation slowly converges to the solution
- Technique presented for PageRank, but extends to linear inverse problems, graph filtering, interpolation, etc.
- Ongoing work: parallelizing single graph computation on multiple processors. Needs deeper understanding of communication overhead.

Future work:
Do you have a graph problem that needs to be processed faster?

Thank you!
Quickest Search for Local Structures in Random Graphs

Javad Heydari     Ali Tajer

Rensselaer Polytechnic Institute
Salient Anomalies

*rare* and at the same time *significant* outliers in large datasets

**fraud detection**
- > 1.5 B CCs

**spectrum sensing**
- > 300 GHz

**outage detection**
- > 500,000 Miles

**query search**
- > 10 B pages

**isolated effects**
(e.g., CC frauds)

**network-wide effects**
(e.g., line outage in power grids)
How to develop *accurate* and *fast* algorithms for recovering rare events from noisy high-dimensional data?
Example: Network-wide Effects

IEEE 39 bus power grid model

outage

surveillance systems

presence of an object changes the correlation structure

outage in a line leads to change in correlation structure
anomaly detection in networks can be viewed as detecting and localizing correlation structures

\[ \mathcal{N}(0, I_n) \quad \text{versus} \quad \mathcal{N}(0, \Sigma) \]
Networked Data

a network of $n$ interconnected entities
Networked Data

a network of $n$ interconnected entities
Networked Data

a network of \( n \) interconnected entities
Networked Data

a network of $n$ interconnected entities

generations of random fields $\{X_1, \ldots, X_n\}$ are inter-dependent

$H_0 : \{X_1, \ldots, X_n\} \sim P$ vs. $H_1 : \{X_1, \ldots, X_n\} \not\sim P$
Quickest Correlation Detection

**Question:**
minimum number of samples to determine if \( \{X_1, \ldots, X_n\} \sim \mathbb{P} \)?

coupled sensing & decision making:
- What is the optimal path for sampling the graph?
- What is the optimal time to stop sampling?
**Quickest Decision**

minimize # of samples  s.t.  controlled reliability

\[
\inf \mathbb{E}\{\# \text{ of samples}\}
\]

s.t.  \( P_0^1 \leq \alpha \)

\( P_1^0 \leq \beta \)

**Q1:** what is the optimal stopping time \( \tau \)?

**Q2:** which node should be sampled at time \( t \leq \tau \)?
Special Case:

Quickest Search for One Node
**Quickest Search for One Node**

- network is *normal* if $\forall i: \mathcal{X}_i \sim F_0$
- network is *anomalous* if $\exists i: \mathcal{X}_i \sim F_1$

**Goal:** quickest sequential search for identifying one sequence $\sim F_1$
Sequential Search

- \( t = 1 \): initiate by measuring \( \mathcal{X}^1 \rightarrow Y_1 \)
- \( t = 2 \): depending on the measurement at \( t = 2 \) : \( \rightarrow Y_2 \)
- \( t = \tau \): form a confident decision
Sampling Strategy

take one measurement at-a-time

Q1: what is the stopping time?

Q2: when a sequence should be discarded?

Q3: which sequence should be measured next?
Quickest Detection

- inherent tension between decision reliability and delay

\[
\text{optimize } \text{ delay (quality)} \quad \text{s.t.} \quad \text{controlled quality (delay)}
\]

- Bayesian integrated cost:

\[
\inf \mathbb{P}(\mathcal{X}^{s_\tau} \sim F_1) + c \cdot \mathbb{E}[\tau]
\]

- \(\mathcal{F}_t\): filtration generated up to time \(t\) from \(\{Y_1, Y_2, \ldots, Y_t\}\)

Decision at time \(t\):

1. \(C_1(\mathcal{F}_t)\): cost of observation
2. \(C_2(\mathcal{F}_t)\): cost of exploration
3. \(C_3(\mathcal{F}_t)\): cost of detection
optimal decision at time $t$: action with the minimal cost-to-go

<table>
<thead>
<tr>
<th>Optimal Stopping Time</th>
<th>Optimal Switching Rule</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Terminate sampling and declare a decision if</strong></td>
<td><strong>Discard the current sequence and switch to a new one</strong></td>
</tr>
<tr>
<td>cost of detection $\leq \min{\text{cost of observation}, \text{cost of exploration}}$</td>
<td>cost of exploration $\leq$ cost of observation</td>
</tr>
</tbody>
</table>
Intuition From A Special Case

linear search over line graphs

\[ \mathbb{P}(X_i \sim F_1 \mid X_{i-1} \sim F_0) = \epsilon_0 \]
\[ \mathbb{P}(X_i \sim F_1 \mid X_{i-1} \sim F_1) = \epsilon_1 \]

positive dependency, i.e., \( \epsilon_1 \geq \epsilon_0 \)

optimal rules

negative dependency, i.e., \( \epsilon_1 < \epsilon_0 \)

asymptotically optimal rules
Positive Dependency

optimal decision at time $t$: action with the minimal cost-to-go

- three non-overlapping regions
- in each region one cost is strictly dominant

1. $C_1(\mathcal{F}_t)$: cost of observation
2. $C_2(\mathcal{F}_t)$: cost of exploration
3. $C_3(\mathcal{F}_t)$: cost of stopping

$$\frac{\epsilon_0}{1 - \epsilon_1 + \epsilon_0}$$
\[ \pi_t \]

switch to a new sequence

\[ \pi^*_U \]

stop

\[ \frac{\epsilon}{1 - (\epsilon_1 - \epsilon_0)} \]

\[ t \]
Negative Dependency

$C_1$ (observation) and $C_2$ (exploration) might intersect more (odd number)

- counter-intuitive
- no closed-form characterization of the intersections
- previous algorithm is **asymptotically** optimal
Generalization

Quickest Search for Correlation Structures
Quickest Decision

\begin{align*}
\text{minimize delay} & \quad \text{s.t.} \quad \text{controlled reliability} \\
\inf & \quad \mathbb{E}\{\# \text{ of samples}\} \\
\text{s.t.} & \quad \mathbb{P}_{0}^{1} \leq \alpha \\
& \quad \mathbb{P}_{0}^{0} \leq \beta
\end{align*}

**Q1:** what is the optimal stopping time $\tau$?

**Q2:** which node should be sampled at time $t \leq \tau$?
Existing Art: Chernoff Rule

relevant literature (recent):

- multiple hypothesis testing (Nitinawarat, et al. 2013)
- active hypothesis testing (Naghshvar & Javidi 2013)
- search over correlated data (Heydari & Tajer 2014)

intuition (from controlled sensing):

- select the hypothesis with the highest likelihood
- take the action (select the node) that maximizes the information content

implications:

- start with sampling the most densely connected node
- devise a metric for density of connection
at time $t$ assign two measures to node $i$:

if $H_0$ is true: \[ D_i^0(t) = D_{KL} \left( f_0(\cdot; i \mid \text{past}) , f_1(\cdot; i) \right) \]

if $H_1$ is true: \[ D_i^1(t) = D_{KL} \left( f_1(\cdot; i \mid \text{past}) , f_0(\cdot; i) \right) \]
Sampling Strategy: Chernoff Rule

at time $t$ assign two measures to node $i$:

if $H_0$ is true: \[ D^i_0(t) = D_{KL} \left( f_0(\cdot; i \mid \text{past}) , f_1(\cdot; i) \right) \]

if $H_1$ is true: \[ D^i_1(t) = D_{KL} \left( f_1(\cdot; i \mid \text{past}) , f_0(\cdot; i) \right) \]

\[ \psi(t) = \begin{cases} 
\arg \max_{i \in \varphi_t} D^i_0(t) & \text{if } \pi_{t-1} < \frac{1}{2} \\
\arg \max_{i \in \varphi_t} D^i_1(t) & \text{if } \pi_{t-1} \geq \frac{1}{2} 
\end{cases} \]

index of source to be sampled at $t$

posterior at $t-1$ that $H_1$ is true
Sampling Strategy: Chernoff Rule

Example: text a Gaussian MRF versus independence.

covariance matrix of the GMRF $\Sigma_{ij} = \rho_{ij}$

$$D^i_1(t) = \sum_{j \in \psi^{t-1}} \log \frac{1}{1 - \rho_{ij}^2} + \frac{\rho_{ij}^2}{1 - \rho_{ij}^2} (Y_{tj}^2 - 1).$$

increasing in $\rho_{ij}$

at time $t$ sample node $\arg \max_i D^i_1(t)$

$H_j$ is more likely

- # of neighbors
- correlation level
**Sampling Strategy: Chernoff Rule**

Example: text a Gaussian MRF versus independence.

The covariance matrix of the GMRF $\Sigma_{ij} = \rho_{ij}$

$$D_1^i(t) = \sum_{j \in \psi^{t-1}} \log \frac{1}{1 - \rho_{ij}^2} + \frac{\rho_{ij}^2}{1 - \rho_{ij}^2} (Y_{tj}^2 - 1).$$

**Shortcoming:**
sampling does not depart a disconnected subgraph until all nodes exhausted
Sampling Strategy: Chernoff Rule

Example: text a Gaussian MRF versus independence.

covariance matrix of the GMRF $\Sigma_{ij} = \rho_{ij}$

$$D_1^i(t) = \sum_{j \in \psi^{t-1}} \log \frac{1}{1 - \rho_{ij}^2} + \frac{\rho_{ij}^2}{1 - \rho_{ij}^2} (Y_{t_j} - 1).$$

Shortcoming:
sampling does not depart a disconnected subgraph until all nodes exhausted

Reason:
impacts of future decisions not incorporated
Optimal Path Selection Rule

\[ D^i_1(t) = D_{KL} \left( f_1(\cdot; i\mid \text{past}) , f_0(\cdot; i) \right) \]

change to:

\[ D^i_1(t, S) = \frac{1}{|S|} \ D_{KL} \left( f_1(\cdot; S\mid \text{past}) , f_0(\cdot; S) \right) \]

all possible unobserved sequences of nodes that contain \( i \)

\[ \psi(t) = \begin{cases} 
\arg \max_{i, S} D^i_0(t, S) & \text{if } \pi_{t-1} < \frac{1}{2} \\
\arg \max_{i, S} D^i_1(t, S) & \text{if } \pi_{t-1} \geq \frac{1}{2} 
\end{cases} \]
Optimal Path Selection Rule

\[ D^i_1(t) = D_{KL}\left( f_1(\cdot; i \mid \text{past}) , f_0(\cdot; i) \right) \]

change to

\[ D^i_1(t, S) = \frac{1}{|S|} D_{KL}\left( f_1(\cdot; S \mid \text{past}) , f_0(\cdot; S) \right) \]

all possible unobserved sequences of nodes that contain \( i \)

**Challenge:** Identifying the optimal node:
- involves identifying the path with the optimal perspective
- has exponential search complexity
Key Observation

\[
\arg \max_{i, S} D^i_0(t, S) = \arg \max_{i, T} D^i_0(t, T)
\]

all possible sequences of unobserved nodes that contain \(i\)

all possible sequences of unobserved neighbors of \(i\)

Implication:

significant reduction in search complexity
The proposed selection rule followed by likelihood ratio test as the stopping rule is asymptotically optimal.

Likelihood ratio: \[ \Lambda_t = \log \frac{f_1(Y_t; \psi_t)}{\prod_{s=1}^{t} f_0(Y_s; \psi(s))} \]

Stopping rule: \[ \tau = \inf \left\{ t : \Lambda_t \notin (\gamma_L, \gamma_U) \right\}, \]

\( \gamma_L \) and \( \gamma_U \) are selected to control error rates below \( \alpha \) and \( \beta \).
Anomaly Detection in Power Grid
Localizing Line Outages

**Line Outage:** failure of transmission lines due to system faults or natural failures

\[ \theta_1 \Rightarrow (\hat{\theta}_1^t, v_1^t) \]
\[ \theta_i \Rightarrow (\hat{\theta}_i^t, v_i^t) \]
\[ \theta_2 = (p_2^t, v_2^t) \]
\[ \theta_3 \rightarrow \hat{\theta}_3 \]
\[ \theta_4 \rightarrow \hat{\theta}_4 \]

**Goals:**

1. **\( \theta_i \):** Collected (measurement) data to determine whether an outage exists.
2. **\( \hat{\theta}_i \):** Anomalous measurements from bus \( i \) post-event.
3. If an outage is deemed to exist, localize it.
Correlation Model

\( x_{ij} \) \{ reactance of the line connecting buses \( i \) and \( j \) \}

\( r_{ij} \triangleq \frac{\beta_{ij}}{r_{ij}} \)

\( \sum_{j \in \psi_{t-1}} \log \frac{1}{1 - r_{ij}^2} + \sum_{j \in S^i_t} \log \frac{1}{1 - r_{ij}^2} \)

measure assigned to bus \( i \) at time \( t \)

correlation structure

data

IEEE 14-bus model

\( v_3 = |v_3| \angle \theta_3 \)

resistance between buses 1 and 2
Outage Localization Performance

- IEEE standard 118-bus model
- We use measurements from only 50 PMUs
Outage Localization Performance

- IEEE standard 118-bus model
Computational Complexity

- IEEE standard 118-bus model

<table>
<thead>
<tr>
<th>Number of measurements</th>
<th>30</th>
<th>50</th>
<th>70</th>
<th>90</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{DA}(sec)$</td>
<td>0.2774</td>
<td>0.8069</td>
<td>1.4681</td>
<td>2.0817</td>
</tr>
<tr>
<td>$t_{ES}(sec)$</td>
<td>6.4415</td>
<td>14.4312</td>
<td>20.8759</td>
<td>23.3983</td>
</tr>
<tr>
<td>$t_{ES}/t_{DA}$</td>
<td>23.2</td>
<td>17.9</td>
<td>14.2</td>
<td>11.2</td>
</tr>
</tbody>
</table>

$$\arg \max_{i,S} D_0^i(t, S) = \arg \max_{i,T} D_0^i(t, T)$$

- all possible sequences of unobserved nodes that contain $i$
- all possible sequences of unobserved neighbors of $i$
Scalability

- IEEE standard 2383-bus model
Decentralized Fusion of Low-Pass Weighted Particle Clouds

Michael Rabbat

Joint with: Chon Wang Chao (McGill)
and Stephane Blouin (DRDC)

McGill
Motivation

A collection of sensors $v \in \mathcal{V}$ collaborate to track the state $x_t$ of an evolving process.

- State dynamic model:
  \[ x_t = f(x_{t-1}) + \xi_t \]

- Observation model at sensor $v$:
  \[ z_{t,v} = h_{t,v}(x_t) + \zeta_{t,v} \]

Each sensor gathers local observations, runs a local tracking algorithm.

Sensors communicate to synchronize, improve accuracy.
Particle Filters in One Slide

Approximate posterior as $\hat{p}(x_t|z_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - \hat{x}_t^{(i)})$
Particle Filters in One Slide

Approximate posterior as $\hat{p}(x_t|z_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - \hat{x}_t^{(i)})$
Approximate posterior as $\hat{p}(x_t | z_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - \hat{x}_t^{(i)})$
Particle Filters in One Slide

Approximate posterior as $\hat{p}(x_t|z_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(x_t - \hat{x}_t^{(i)})$
Distributed Particle Filters

Multiple sensors \(\Rightarrow\) Multiple observations
\(\Rightarrow\) Communication

Under conditional independence assumption, PF update becomes

\[
    w_t^{(i)} \propto w_{t-1}^{(i)} \exp \left( \bar{\gamma}_t^{(i)} \right)
\]

where

\[
    \bar{\gamma}_t^{(i)} = \sum_{v \in V} \ln p(z_{t,v} | \hat{x}_t^{(i)})
\]

\(\Rightarrow\) Linear function of local likelihoods, use gossip/consensus
Distributed Particle Filters

Multiple sensors $\implies$ Multiple observations $\implies$ Communication

Under conditional independence assumption, PF update becomes

$$w_t^{(i)} \propto w_{t-1}^{(i)} \exp \left( \gamma_t^{(i)} \right)$$

where

$$\gamma_t^{(i)} = \sum_{v \in V} \ln p(z_{t,v} | \hat{x}_t^{(i)})$$

$\implies$ Linear function of local likelihoods, use gossip/consensus

Problem: Comm. overhead $\propto$ num. particles $N$

*How to reduce overhead without sacrificing accuracy?*
Transform Coding

**Observation:** All sensors sample/predict same particle locations, only weights need to be synchronized.

Nearby particles have similar weights
Transform Coding

**Observation:** All sensors sample/predict same particle locations, only weights need to be synchronized.

Nearby particles have similar weights

Compress weighted particle cloud by adapting representation to particle distribution

\[
\ln p(z_{t,v} | \hat{x}^{(i)}_t) \in \mathbb{R}^N \\
y_{t,v} \in \mathbb{R}^k \\
(\gamma \ll N) \\
\bar{y}_t \in \mathbb{R}^k \\
\tilde{\gamma}_t \in \mathbb{R}^N
\]
Transform Coding

Observation: All sensors sample/predict same particle locations, only weights need to be synchronized.

Nearby particles have similar weights

Compress weighted particle cloud by adapting representation to particle distribution

\[
\begin{align*}
\ln p(z_{t,v} | \hat{x}_t^{(i)}) &\in \mathbb{R}^N \\
y_{t,v} &\in \mathbb{R}^k (k \ll N) \\
\bar{y}_t &\in \mathbb{R}^k \\
\tilde{\gamma}_t &\in \mathbb{R}^N
\end{align*}
\]

Encoding: Piece-wise constant approximation

Decoding: Graph-Laplacian interpolation
1. Cluster particles (e.g., $k$-means), assignment matrix $C \in \{0, 1\}^{k \times N}$

2. Aggregate graph signal (log-likelihoods) within clusters

$$y_{t,v}^c = \sum_{i=1}^{N} C_{c,i} \ln p(z_{t,v} | \hat{x}_t^{(i)})$$

3. Gossip to obtain $\overline{y}_t \in \mathbb{R}^k$

$$\overline{y}_t^c = \sum_{v \in \mathcal{V}} y_{t,v}^c, \quad c = 1, \ldots, k$$
Decoding by Minimizing Variation

1. Form a graph over the particles (e.g., nearest neighbor graph) and calculate the Laplacian $L$

2. Obtain $\{\tilde{\gamma}_t^{(i)}\}_{i=1}^N$ as solution to

$$\begin{align*}
\text{minimize} & \quad \frac{1}{2} \gamma^T L \gamma \\
\text{subject to} & \quad C \gamma = \bar{y}_t
\end{align*}$$
Decoding by Minimizing Variation

1. Form a graph over the particles (e.g., nearest neighbor graph) and calculate the Laplacian \( L \)

2. Obtain \( \{\tilde{\gamma}_t^{(i)}\}_{i=1}^N \) as solution to

   \[
   \begin{align*}
   \text{minimize} & \quad \frac{1}{2} \gamma^T L \gamma \\
   \text{subject to} & \quad C \gamma = \bar{y}_t
   \end{align*}
   \]

Remarks:

- Graph connected \( \Rightarrow \) QP has unique solution
- Found by solving a linear system
Example

\[ N = 2000 \text{ particles, } k = 50 \text{ clusters/coefficients} \]

True weights

Piece-wise const. approx. \((\ell_1 \text{ err } 0.3743)\)

Min. var. reconstruction \((\ell_1 \text{ err } 0.1211)\)
Example

\[ N = 2000 \text{ particles, } k = 50 \text{ clusters/coefficients} \]

True weights

Piece-wise const. approx. \((\ell_1 \text{ err 0.3743})\)

Min. var. reconstruction \((\ell_1 \text{ err 0.1211})\)

(Using 50 GFT coeffs: \(\ell_1 \text{ err 0.3341}\))
Tracking Experiment

- Two objects, leader/follower
- 16 sensors, perturbed grid
- Noisy bearings, $\sigma = 5^\circ$
- Compare with
  - Centralized
  - Distributed, no compression
  - Set-Membership approach
    [Farahmand et al. 2011]

Fix $N = 2000$ particles, vary clusters, gossip
Fix communication overhead at 200

Gossip iterations

Total ARMSE of the two targets (km)

- BPF–distributed (200 particles)
- BPFC–kmeans (2000 particles, 200 clusters)
- SetM (200 particles)
Is This Phenomenon More General?

Run experiments:

- Graphs with $n = 2000$ (Random geo. and Watts-Strogatz)
- Generate signals with GFT coefficients

$$\hat{x}(j) \sim \mathcal{N}(0.9^j, 0.2^2), \quad j = 1, \ldots, n$$

(Laplacian eigenbasis, $L = U\Lambda U^T$)
- Add noise, SNR $\approx 13$ dB

$$z = U\hat{x} + \epsilon$$

Compare methods:

1. Piece-wise constant approx. over $k$ clusters (normalized cut), with minimum variation reconstruction
2. Project $z$ onto $k$ lowest Laplacian eigenvectors
Preliminary Results

For Rand. geo. and Small world, the MSE decreases rapidly with increasing k for both PWC + Min Var and Lap Eig methods. The MSE values are given in $10^{-4}$.
Conclusion

Summary:
• Communication-efficient fusion of particle weights
• **Encode**: Piece-wise constant approximation via clustering
• **Decode**: Minimum variation reconstruction
• Interesting performance (accuracy and computationally)

Open questions:
• Theoretical justification?
• Connections to other areas (sampling, signal representations)?

michael.rabbat@mcgill.ca