Graph Signal Processing Workshop

Carnegie Mellon University June 2, 2017

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.

Social network

Brain structure

Telecommunication

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!

ConvNets on graphs: spatial approach

Niepert, Ahmed, and Kutzkov 2016

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

$\mathcal{G} = (\mathcal{V}, \mathcal{E}, W)$: 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
- $L = I_n - D^{-1/2} W D^{-1/2}$: 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" $\Lambda = \begin{bmatrix} \lambda_1 & & 0 \\ & \ddots & \\ 0 & & \lambda_n \end{bmatrix} \in \mathbb{R}^{n \times n}$

Graph Fourier Transform

1. Graph signal $x : \mathcal{V} \rightarrow \mathbb{R}$ seen as $x \in \mathbb{R}^n$
2. Transform: $\hat{x} = \mathcal{F}_{\mathcal{G}}\{x\} = U^T x \in \mathbb{R}^n$
3. Inverse: $x = U\hat{x} = UU^T x = x$

Definitions: convolution on graphs

Shuman, Narang, Frossard, Ortega, and Vandergheynst 2013

Convolution theorem:

$$\begin{aligned} x *_{\mathcal{G}} g &= U\left(U^T g \odot U^T x\right) \\ &= U\left(\hat{g} \odot U^T x\right) \end{aligned}$$

Conveniently written as:

$$\begin{aligned} x *_{\mathcal{G}} g &= U \begin{bmatrix} \hat{g}(\lambda_1) & & 0 \\ & \ddots & \\ 0 & & \hat{g}(\lambda_n) \end{bmatrix} U^T x \\ &= U \hat{g}(\Lambda) U^T x \\ &= \hat{g}(L) x \end{aligned}$$

Spectral filtering of graph signals

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

Non-parametric filter:

$$\hat{g}_\theta(\Lambda) = \mathrm{diag}(\theta),\ \theta \in \mathbb{R}^n$$

- 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

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

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

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, Vandergheynst, and Gribonval 2011

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

- Chebyshev polynomials: $T_k(x) = 2xT_{k-1}(x) - T_{k-2}(x)$ with $T_0 = 1$ and $T_1 = x$
- Filtering: $y = \hat{g}_\theta(L)x = \sum_{k=0}^{K-1} \theta_k T_k(\tilde{L})x$
- 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

Applications

- Semi-supervized 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
- Computer Graphics [Monti, Boscaini, et al. 2016; Yi, Su, Guo, and Guibas 2016; Wang, Gan, Zhang, and Shui 2017; Simonovsky and Komodakis 2017]
- 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

- $h_t = \tanh(W_x *_{\mathcal{G}} x_t + W_h *_{\mathcal{G}} h_{t-1})$
- $y_t = W *_{\mathcal{G}} h_t$
- 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

Real data: Wikipedia

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

- Paper: Defferrard, Bresson and Vandergheynst, Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering, NIPS, 2016.
- Paper: Seo, Defferrard, Bresson and Vandergheynst, Structured Sequence Modeling with Graph Convolutional Recurrent Networks, arXiv, 2016.
- Code: `https://github.com/mdeff/cnn_graph`

Thanks Questions?

PS: next year, we're organizing the GSP workshop at EPFL!

Graph Signal Processing Workshop 2017

May 31 - June 2, 2017 | Carnegie Mellon University

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

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

NEUROIMAGING AND NETWORK SCIENCE

Fornito et al. 2015
Nature Neuro

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

The application of Graph Theory for neuroimaging is now widespread in the neuroimaging community.

THE NEXT STEP ?

- Laplacian eigenvectors of a graph based on brain structure (gray and white matter) can predict spontaneous brain activity (Atasoy, Donelly & Pearson, 2017, Nature communications)

THE NEXT STEP ?

- Laplacian eigenvectors can be interpreted functionally (Margulies et al. 2016. PNAS. here termed “connectivity gradients”)

Fig. 4. NeuroSynth metaanalysis of regions of interest along the principal gradient using 24 topic terms. Terms are ordered by the weighted mean of their location along the gradient. Sensory processing terms are located at the top followed by domain-general cognitive functions and then, higher-order abstract cognitive and memory-related processes. Similar results using the BrainMap database are available in *SI Materials and Methods*. autobiographical mem., autobiographical memory; multisensory proc., multisensory processing.

-> Therfore GSP appears as an ideal framework to analyse, predict and interpret brain activity.

GSP APPLIED TO NEUROIMAGING

- 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
“Evaluating Graph Signal Processing for Neuroimaging through Classification and Dimensionality Reduction”, https://arxiv.org/abs/1703.01842

Haxby et al. 2001, Science

Brain parcellation to create ROIs (clusters)
Geometry : baricenter of ROIs

BASC Atlas,
Bellec et al.
Neuroimage 2010

IMT Atlantiq
Bretagne-Pays de
École Mines-Télé

fMRI simulation : neuRosim

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

Contrast Condition 1 – Condition 2

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

GSP using 6 types of graph

Geometric graphs:

- Fully connected geometric graph: $w_{i,j}$=distance(i,j)
- Geometric:

  if distance(i,j)< alpha: $w_{i,j}$=distance(i,j)

  else $w_{i,j}$=0

Functional graphs ***(based on rest data)***:

- Covariance matrix ($w_{i,j}$=cov(i,j))
- Kalofolias method (estimates graph on which observed signals are smooth)

Mixed functional and geometric graphs:

- Semilocal :

  if distance(i,j)< alpha: $w_{i,j}$=cov(i,j)

  else $w_{i,j}$=0
- Graph associating both distance between brain areas and connectivity measures (« fundis »):

$$w_{i,j} = \exp\left(-\frac{\left(1 - \| correlation(i,j) \|\right)^2}{2\theta}\right) . \exp\left(-\frac{distance(i,j)^2}{2\sigma}\right)$$

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 I

ACCURACY OF THE CLASSIFICATION FOR ALL GRAPH-BASED SAMPLING METHODS (IN %) FOR THE TWO DIFFICULTY GROUPS. STARRED NUMBERS INDICATE BEST SCORES IN THEIR CATEGORY.

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

-> next results are shown only for the semilocal graph

RESULTS SIMULATION – SEMILOCAL GRAPH

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

Keeping 50 dimensions

RESULTS SIMULATION – SEMILOCAL GRAPH

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

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

RESULTS SIMULATION AND HAXBY SEMILOCAL GRAPH

TABLE II

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

Method	Simulation		Haxby	
	Easy	Difficult	Face-House	Cat-Face
PCA	88.8%	65.5%	82.7%	63.6%
ICA	90.2%	65.3%	84.4%	67.0%
ANOVA	**92.1%**	67.3%	85.5%	65.5%
Graph sampling	90.9%	**72.5%**	**88.2%**	**69.0%**

RESULTS ON HAXBY DATASET

Dim: 50

Contrast House-Face

SVM weight after Graph Sampling (Semilocal)

IMT Atlantique
Bretagne-Pays de la Loire
École Mines-Télécom

GSP FOR BRAIN DECODING - CONCLUSIONS

- 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 ?

Graph Signal Processing Workshop 2017

May 31 - June 2, 2017 | Carnegie Mellon University

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 unpredicatable variability of "workers" in highly parallel graph analyses.
- Main message: error-correcting codes $>$ replication.

  ... shown theoretically, and experimentally.

The Straggler Effect in Distributed and Parallel Computing

Some processors unpredictably slower than others.
A core problem in parallelization: how to deal with this unpredictable variability?

Replication: a strategy to address straggling

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

Coding for Speeding up Distributed Computing

Classical coded computation

Coding for Speeding up Distributed Computing

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]

Normalized Adj matrix Restart Prob.

$$x=(1-d)Ax+dr$$

PageRank output (stationary distribution) PageRank input (preference)

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

$$\mathbf{x}^{(l+1)} = (1-d)\mathbf{A}\mathbf{x}^{(l)} + d\mathbf{r}.$$

Coded graph analytics through an example problem

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

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

$$\mathbf{x}^{(l+1)} = (1-d)\mathbf{A}\mathbf{x}^{(l)} + d\mathbf{r}.$$

- **Problem formulation**: Solve k personalized PageRank problems using $P > k$ processors.

Power-iterations converge to PageRank solution

$$\mathbf{x}^{(l+1)} = (1-d)\mathbf{A}\mathbf{x}^{(l)} + d\mathbf{r}.$$

Converges to $\mathbf{x}^*$ satisfying $\mathbf{x}^* = (1-d)\mathbf{A}\mathbf{x}^* + d\mathbf{r}$.

Subtracting, $\mathbf{e}^{(l+1)} = (1-d)\mathbf{A}\mathbf{e}^{(l)}$, where $\mathbf{e}^{(l)} = \mathbf{x}^{(l)} - \mathbf{x}^*$.

Classical coded computation applied to personalized PageRank

- Initialize (Encoding)

$$[\mathbf{s}_1, \ldots, \mathbf{s}_P] = [\mathbf{r}_1, \ldots, \mathbf{r}_k] \cdot \mathbf{G}_{k \times P}.$$

- Parallel Computing:
  l_i power iterations at the i-th worker with input $\mathbf{s}_i$

$$\mathbf{Y}_{N \times P}^{(T_{\text{dl}})} = [\mathbf{y}_1^{(l_1)}, \ldots, \mathbf{y}_P^{(l_P)}].$$

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

$$\widehat{\mathbf{X}}^\top = \tilde{\mathbf{G}}^{-1} (\mathbf{Y}^{(T_{\text{dl}})})^\top.$$

Classical coded computation on for personalized PageRank: errors blow up!

Decoding: $\widehat{\mathbf{X}}^\top = \tilde{\mathbf{G}}^{-1}(\mathbf{Y}^{(T_{\text{dl}})})^\top$.

- 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)

$$[\mathbf{s}_1, \ldots, \mathbf{s}_P] = [\mathbf{r}_1, \ldots, \mathbf{r}_k] \cdot \mathbf{G}.$$

- Parallel Computing:
  l_i power iterations at the i-th worker with input $\mathbf{s}_i$

$$\mathbf{Y}_{N\times P}^{(T_{\text{dl}})} = [\mathbf{y}_1^{(l_1)}, \ldots, \mathbf{y}_P^{(l_P)}].$$

- Post Processing (Decoding)

$$\widehat{\mathbf{X}}^\top = (\mathbf{G}\mathbf{\Lambda}^{-1}\mathbf{G}^\top)^{-1}\mathbf{G}\mathbf{\Lambda}^{-1}(\mathbf{Y}^{(T_{\text{dl}})})^\top.$$

Similar to the "weighted least-square" solution.

The diagonal weight matrix $\mathbf{\Lambda}$

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

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

$$\mathbf{C}(l_i) = \mathbf{B}^{l_i}\mathbf{C}_E(\mathbf{B}^\top)^{l_i}.$$

- $l_i =$ number of iterations at the i-th processor
- $\mathbf{B} =$ "contraction" matrix in the iterative algorithm
- $\mathbf{C}_E =$ error covariance at the 0-th iteration
- $\text{trace}(\mathbf{C}(l_i)) =$ **MSE at the i-th processor**

Coded computing provably outperforms repetition

For iid speeds of computation:

Theorem 1: For fixed deadline T_{dl}, as $P \to \infty$,
Coded MSE $<$ **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 **coded** $>$ **replication** $=$ **uncoded**, i.e., **the coded PageRank error drops fastest**.

$$ErrExp := \lim_{T_{\mathsf{dl}} \to \infty} -\frac{1}{T_{\mathsf{dl}}} \log MSE(T_{dl})$$

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 $O(PkN)$
- Parallel computing $O(N^a l)$ $a \approx 1.68$ 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!

Graph Signal Processing Workshop 2017

May 31 - June 2, 2017 | Carnegie Mellon University

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)

Quickest Anomaly Detection

opportunities

opportunity cost

risks

recovery cost

features

computational cost

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 in a line leads to
change in correlation structure

surveillance systems

presence of an object
changes the correlation structure

Correlation Detection

anomaly detection in networks can be viewed as *detecting* and *localizing* **correlation structures**

background noise

$\mathcal{N}(\mathbf{0}, \boldsymbol{I}_n)$

versus

object present

$\mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma})$

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 $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\}$ are inter-dependent

$$\mathsf{H}_0 : \{\mathcal{X}_1, \ldots, \mathcal{X}_n\} \sim \mathbb{P} \quad \text{vs.} \quad \mathsf{H}_1 : \{\mathcal{X}_1, \ldots, \mathcal{X}_n\} \not\sim \mathbb{P}$$

Quickest Correlation Detection

Question:
minimum number of samples to determine if $\{\mathcal{X}_1, \ldots, \mathcal{X}_n\} \sim \mathbb{P}$?

$$\begin{bmatrix} X_1 \\ X_2 \\ \vdots \\ \vdots \\ X_n \end{bmatrix} \xrightarrow{\text{sequential sampling}} \begin{bmatrix} Y_1 \\ \vdots \\ Y_\tau \end{bmatrix}$$

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

$$\inf \quad \mathbb{E}\{\#\text{ of samples}\}$$

$$\text{s.t.} \quad \mathsf{P}_0^1 \leq \alpha$$

$$\mathsf{P}_1^0 \leq \beta$$

Q1: what is the optimal stopping time τ?

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 $\mathcal{X}^1 \quad \rightarrow \quad Y_1$
- $t = 2$: depending on the measurement at $t = 2$: $\quad \rightarrow \quad Y_2$

- $t = \tau$: 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

optimize delay (quality) s.t. controlled quality (delay)

- Bayesian integrated cost:

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

- $\mathcal{F}_t$: 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 Rules

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

optimal stopping time	terminate sampling and declare a decision if cost of detection $\leq$ $\min\{$cost of observation , cost of exploration$\}$
optimal switching rule	discard the current sequence and switch to a new one cost of exploration $\leq$ cost of observation

Intuition From A Special Case

linear search over line graphs

- correlation structure:

$$\mathbb{P}(\mathcal{X}^i \sim F_1 \mid \mathcal{X}^{i-1} \sim F_0) = \epsilon_0$$
$$\mathbb{P}(\mathcal{X}^i \sim F_1 \mid \mathcal{X}^{i-1} \sim F_1) = \epsilon_1$$

positive dependency, i.e., $\epsilon_1 \geq \epsilon_0$	negative dependency, i.e., $\epsilon_1 < \epsilon_0$
optimal rules	*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*

Positive Dependency

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

minimize delay s.t. controlled reliability

$$\inf \quad \mathbb{E}\{\#\text{ of samples}\}$$

$$\text{s.t.} \quad \mathsf{P}_0^1 \leq \alpha$$

$$\mathsf{P}_1^0 \leq \beta$$

Q1: what is the optimal stopping time τ?

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

Existing Art: Chernoff Rule

at time t assign two measures to node i:

if H_0 is true: $$D_0^i(t) = D_{\mathrm{KL}}\Big(f_0(\cdot; i|\text{ past})\ ,\ f_1(\cdot; i)\Big)$$

if H_1 is true: $$D_1^i(t) = D_{\mathrm{KL}}\Big(f_1(\cdot; i|\text{ past})\ ,\ f_0(\cdot; i)\Big)$$

H_0: independent sources

H_1: correlated sources

Sampling Strategy: Chernoff Rule

at time t assign two measures to node i:

if H_0 is true: $$D_0^i(t) = D_{\mathrm{KL}}\Big(f_0(\cdot; i|\ \mathrm{past})\ ,\ f_1(\cdot; i)\Big)$$

if H_1 is true: $$D_1^i(t) = D_{\mathrm{KL}}\Big(f_1(\cdot; i|\ \mathrm{past})\ ,\ f_0(\cdot; i)\Big)$$

posterior at $t-1$ that H_1 is true

$$\psi(t) = \begin{cases} \arg\max\limits_{i \in \varphi_t}\ D_0^i(t) & \text{if } \pi_{\mathrm{t}-1} < \frac{1}{2} \\ \arg\max\limits_{i \in \varphi_t}\ D_1^i(t) & \text{if } \pi_{\mathrm{t}-1} \geq \frac{1}{2} \end{cases} .$$

index of source to be sampled at t

Sampling Strategy: Chernoff Rule

Example: text a Gaussian MRF versus independence.

covariance matrix of the GMRF $\mathbf{\Sigma}_{ij} = \rho_{ij}$

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

increasing in ρ_{ij}

- # of neighbors
- correlation level

at time t sample node $\arg\max_i D_j^i(t)$

H_j is more likely

Sampling Strategy: Chernoff Rule

Example: text a Gaussian MRF versus independence.

covariance matrix of the GMRF $\mathbf{\Sigma}_{ij} = \rho_{ij}$

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

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 $\mathbf{\Sigma}_{ij} = \rho_{ij}$

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

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

Reason:
impacts of future decisions not incorporated

Optimal Path Selection Rule

$$D_1^i(t) = D_{\mathrm{KL}}\Big(f_1(\cdot; i|\ \text{past})\ ,\ f_0(\cdot; i) \Big)$$

change to

$$D_1^i(t, S) = \frac{1}{|S|}\, D_{\mathrm{KL}}\Big(f_1(\cdot; S|\ \text{past})\ ,\ f_0(\cdot; S) \Big)$$

all possible unobserved sequences of nodes that contain i

$$\psi(t) = \begin{cases} \arg\max\limits_{i,S}\ D_0^i(t, S) & \text{if } \pi_{\mathrm{t}-1} < \frac{1}{2} \\ \arg\max\limits_{i,S}\ D_1^i(t, S) & \text{if } \pi_{\mathrm{t}-1} \geq \frac{1}{2} \end{cases} .$$

Optimal Path Selection Rule

$$D_1^i(t) = D_{\mathrm{KL}}\Big(f_1(\cdot; i|\ \text{past})\ ,\ f_0(\cdot; i) \Big)$$

change to

$$D_1^i(t, S) = \frac{1}{|S|}\, D_{\mathrm{KL}}\Big(f_1(\cdot; S|\ \text{past})\ ,\ f_0(\cdot; S) \Big)$$

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

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

all possible sequences of unobserved nodes that contain i

all possible sequences of unobserved of neighbors of i

Implication:

significant reduction in search complexity

Optimality

Theorem (optimality)

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: $$\tau \; = \; \inf \; \Big\{ t \; : \; \Lambda_t \notin (\gamma_L, \gamma_U) \Big\} \; ,$$

γ_L and γ_U are selected to control error rates below α and β

Anomaly Detection in Power Grid

Localizing Line Outages

line outage: failure of transmission lines due to system faults or natural failures

Goals:

1. Collect measurements to determine whether an outage exists
2. If an outage is deemed to exist, localize it

θ_i: expected (assumed) measurements from bus i post event

$\hat{\theta}_i$: *anomalous* measurements from bus i post event

Correlation Model

x_{ij} $\begin{cases} \text{reactance of the line connecting buses } i \text{ and } j \longrightarrow r_{ij} \triangleq \dfrac{\beta_{ij}}{r_{ij}} \\ \text{captures the correlation between } \theta_i \text{ and } \theta_j \end{cases}$

data

$$M_i(t) = \boxed{\frac{1}{2}\sum_{j\in\psi_{t-1}} \log\frac{1}{1-r_{ij}^2} + r_{ij}^2(\Delta\theta_j^2 - 1)} + \boxed{\frac{1}{2|\mathcal{S}_t^i|}\sum_{j\in\mathcal{S}_t^i} \log\frac{1}{1-r_{ij}^2}}.$$

correlation structure

measure assigned to bus i at time t

IEEE 14-bus model

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

Number of measurements	30	50	70	90
$t_{\mathrm{DA}}(sec)$	0.2774	0.8069	1.4681	2.0817
$t_{\mathrm{ES}}(sec)$	6.4415	14.4312	20.8759	23.3983
$t_{\mathrm{ES}}/t_{\mathrm{DA}}$	23.2	17.9	14.2	11.2

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

all possible sequences of unobserved nodes that contain i

all possible sequences of unobserved of neighbors of i

Scalability

- IEEE standard 2383-bus model

Graph Signal Processing Workshop 2017

May 31 - June 2, 2017 | Carnegie Mellon University

Decentralized Fusion of Low-Pass Weighted Particle Clouds

Michael Rabbat

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

Motivation

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

- State dynamic model:
$$\boldsymbol{x}_t = f(\boldsymbol{x}_{t-1}) + \boldsymbol{\xi}_t$$
- Observation model at sensor v:
$$\boldsymbol{z}_{t,v} = h_{t,v}(\boldsymbol{x}_t) + \boldsymbol{\zeta}_{t,v}$$

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

Sensors communicate to synchronize, improve accuracy.

Particle Filters in One Slide

Approximate posterior as $\widehat{p}(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(\boldsymbol{x}_t - \widehat{\boldsymbol{x}}_t^{(i)})$

Particle Filters in One Slide

Approximate posterior as $\widehat{p}(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(\boldsymbol{x}_t - \widehat{\boldsymbol{x}}_t^{(i)})$

Particle Filters in One Slide

Approximate posterior as $\widehat{p}(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(\boldsymbol{x}_t - \widehat{\boldsymbol{x}}_t^{(i)})$

Particle Filters in One Slide

Approximate posterior as $\widehat{p}(\boldsymbol{x}_t|\boldsymbol{z}_{1:t}) = \sum_{i=1}^{N} w_t^{(i)} \delta(\boldsymbol{x}_t - \widehat{\boldsymbol{x}}_t^{(i)})$

Distributed Particle Filters

Multiple sensors $\Longrightarrow$ Multiple observations
$\Longrightarrow$ Communication

Under conditional independence assumption, PF update becomes

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

where

$$\overline{\gamma}_t^{(i)} = \sum_{v \in \mathcal{V}} \ln p(\boldsymbol{z}_{t,v} | \widehat{\boldsymbol{x}}_t^{(i)})$$

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

Distributed Particle Filters

Multiple sensors $\Longrightarrow$ Multiple observations
$\Longrightarrow$ Communication

Under conditional independence assumption, PF update becomes

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

where

$$\overline{\gamma}_t^{(i)} = \sum_{v \in \mathcal{V}} \ln p(\boldsymbol{z}_{t,v} | \widehat{\boldsymbol{x}}_t^{(i)})$$

$\Longrightarrow$ 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

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

Encoding: Piece-wise constant approximation

Decoding: Graph-Laplacian interpolation

Encoding

1. Cluster particles (e.g., k-means), assignment matrix $\boldsymbol{C} \in \{0,1\}^{k \times N}$
2. Aggregate graph signal (log-likelihoods) within clusters

$$y_{t,v}^{c} = \sum_{i=1}^{N} C_{c,i} \ln p(\boldsymbol{z}_{t,v} | \widehat{\boldsymbol{x}}_t^{(i)})$$

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

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

Decoding by Minimizing Variation

1. Form a graph over the particles (e.g., nearest neighbor graph) and calculate the Laplacian $\boldsymbol{L}$
2. Obtain $\{\widetilde{\gamma}_t^{(i)}\}_{i=1}^N$ as solution to

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{\gamma}^\mathsf{T}\boldsymbol{L}\boldsymbol{\gamma} \\ \text{subject to} & \boldsymbol{C}\boldsymbol{\gamma} = \overline{\boldsymbol{y}}_t \end{array}$$

Decoding by Minimizing Variation

1. Form a graph over the particles (e.g., nearest neighbor graph) and calculate the Laplacian $\boldsymbol{L}$
2. Obtain $\{\widetilde{\gamma}_t^{(i)}\}_{i=1}^N$ as solution to

$$\begin{array}{ll} \text{minimize} & \frac{1}{2}\boldsymbol{\gamma}^\mathsf{T}\boldsymbol{L}\boldsymbol{\gamma} \\ \text{subject to} & \boldsymbol{C}\boldsymbol{\gamma} = \overline{\boldsymbol{y}}_t \end{array}$$

Remarks:

- Graph connected $\Rightarrow$ QP has unique solution
- Found by solving a linear system

Example

$N = 2000$ particles, $k = 50$ clusters/coefficients

True weights

Piece-wise const. approx.
(ℓ_1 err 0.3743)

Min. var. reconstruction
(ℓ_1 err 0.1211)

Example

$N = 2000$ particles, $k = 50$ clusters/coefficients

True weights

Piece-wise const. approx.
(ℓ_1 err 0.3743)

Min. var. reconstruction
(ℓ_1 err 0.1211)

(Using 50 GFT coeffs: ℓ_1 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]

S. Farahmand, S. Roumeliotis, and G. Giannakis, "Set-membership constrained particle filter: Distributed adaptation for sensor networks," *IEEE Trans. SP*, Sep 2011.

Fix $N = 2000$ particles, vary clusters, gossip

Fix communication overhead at 200

Is This Phenomenon More General?

Run experiments:

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

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

(Laplacian eigenbasis, $\boldsymbol{L} = \boldsymbol{U}\boldsymbol{\Lambda}\boldsymbol{U}^\mathsf{T}$)

- Add noise, SNR $\approx$ 13 dB

$$\boldsymbol{z} = \boldsymbol{U}\widehat{\boldsymbol{x}} + \boldsymbol{\epsilon}$$

Compare methods:

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

0.2
0.15
0.1
0.05
0

0.2
0.15
0.1
0.05
0
-0.05

1
0.8
0.6
0.4
0.2
0
$\widehat{x}_j$
0
500
1000
1500
2000
2500
3000
3500
λ_j

1.2
1
0.8
0.6
0.4
0.2
0
$\widehat{x}_j$
0
500
1000
1500
λ_j

Preliminary Results

Rand. geo.

Small world

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