GATED GRAPH RECURRENT NEURAL NETWORKS

Mariana Luna and Ricardo del Rio. Mentors: Luana Ruiz and Alejandro Ribeiro

Dept. of Electrical & Systems Engineering, University of Pennsylvania

Recurrent Neural Networks

- Applications involving learning from real-time/online data are ubiquitous
  - ⇒ Natural language processing and object recognition
- Objective is to map sequential data $\{\mathbf{x}_t\}_{t=1}^T$ to a target representation $\mathbf{y}_t$
- Data points $\mathbf{x}_t$ related through some unknown dependency $f(\mathbf{x})$

$$\mathbf{x}_{t+1} = f(\mathbf{x}_t, \mathbf{x}_{t-1}, \ldots)$$

- Dependence relationship can be modeled as a Hidden Markov Model
  - ⇒ State variable $\mathbf{h}_t$ stores relevant past time information
  - ⇒ $\mathbf{x}_{t+1}$ only depends on $\mathbf{x}_t$ and $\mathbf{h}_t$ ⇒ $\mathbf{x}_{t+1} = g(\mathbf{x}_t, \mathbf{h}_t)$
- Recurrent Neural Networks (RNNs) approximate the hidden state as

$$\mathbf{h}_t = \sigma(\mathbf{A}\mathbf{x}_t + \mathbf{B}\mathbf{h}_{t-1})$$

- $\mathbf{A}$ and $\mathbf{B}$ are linear transforms, σ is a nonlinear function (tanh)
- Representation estimate $\hat{\mathbf{y}}_t$ calculated from $\mathbf{h}_t$ ⇒ $\hat{\mathbf{y}}_t = \sigma(\mathbf{C}\mathbf{h}_t)$

Graph Processes

- RNNs are limited to processing regular data ⇒ time signals, images
- Unsuited to problems presenting alternative data structures (irregular)
  - ⇒ Weather station networks for irrigation of agricultural crops
  - ⇒ Traffic networks established by people commuting from one county to another

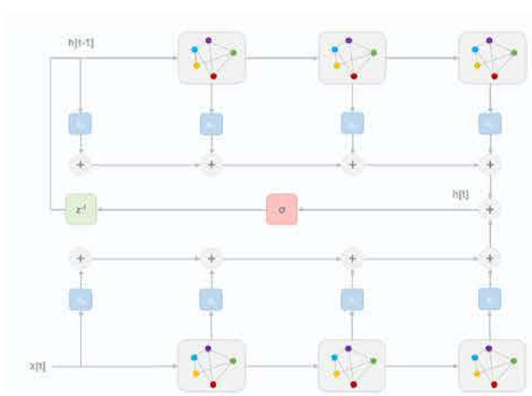
- Graphs encode arbitrary pairwise relationships between data elements
  - ⇒ Data is represented as a signal on the nodes ⇒ graph signal
  - ⇒ Underlying structure has to be incorporated into processing

Graph Recurrent Neural Networks

- Information processing architecture to learn from graph processes
  - ⇒ Exploits both sequential and graph structural information
- Network structure ⇒ Graph matrix $\mathbf{S}$
  - ⇒ $[\mathbf{S}]_{ij}$ = relationship between i, j
  - ⇒ $\mathbf{S}$ is the same for all t
- $\mathbf{x}_t$ graph signal at time t
  - ⇒ $[\mathbf{x}_t]_i$ = signal value at node i
- $\mathbf{A}$ and $\mathbf{B}$ parametrized by the graph

$$\mathbf{h}_t = \sigma(\mathbf{A}(\mathbf{S})\mathbf{x}_t + \mathbf{B}(\mathbf{S})\mathbf{h}_{t-1})$$

- Graph Recurrent Neural Network



Objective

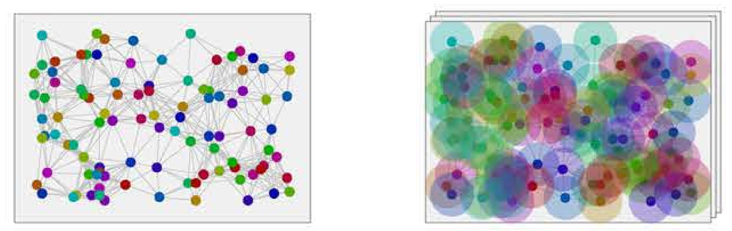
Parametrize linear operations $\mathbf{A}(\mathbf{S})$ and $\mathbf{B}(\mathbf{S})$ such that:
- ⇒ They exploit the structure of the graph
- ⇒ The number of parameters is independent of time and graph size
- ⇒ The operations are permutation invariant

Graph Convolutions

- Our GRNNs generate features through local graph convolutions
  - ⇒ Convolution is a linear shift invariant filter ⇒ K filter taps

$$\mathbf{y} = h_0\mathbf{S}^0\mathbf{x} + h_1\mathbf{S}^1\mathbf{x} + h_2\mathbf{S}^2\mathbf{x} + \ldots + h_{K-1}\mathbf{S}^{K-1}\mathbf{x} = \sum_{k=0}^{K-1} h_k\mathbf{S}^k\mathbf{x} := \mathbf{H}(\mathbf{S})\mathbf{x}$$

  - ⇒ Linear operation that exploits graph topology at the local level
  - ⇒ Permutation invariant ⇒ $\mathbf{H}(\mathbf{P}\mathbf{S}\mathbf{P}^T)\mathbf{P}\mathbf{x} = \mathbf{P}\mathbf{H}(\mathbf{S})\mathbf{P}^T\mathbf{P}\mathbf{x} = \mathbf{P}\mathbf{H}(\mathbf{S})\mathbf{x}$



Graph Convolutional Recurrent Neural Networks

- We write $\mathbf{A}(\mathbf{S})$ and $\mathbf{B}(\mathbf{S})$ as graph convolutions to exploit locality

$$\mathbf{A}(\mathbf{S}) = \sum_{k=0}^{K-1} a_k\mathbf{S}^k \qquad \mathbf{B}(\mathbf{S}) = \sum_{k=0}^{K-1} b_k\mathbf{S}^k$$

- $\mathbf{h}_t$ is a graph signal to avoid any dependence on the number of nodes
- $\mathbf{A}(\mathbf{S})$ and $\mathbf{B}(\mathbf{S})$ could be replaced by GNNs for increased capacity
- The output estimate is calculated as $\hat{\mathbf{y}}_t = \rho(\mathbf{C}(\mathbf{S})\mathbf{h}_t)$
  - ⇒ $\mathbf{C}(\mathbf{S})$ is another linear graph filter or GNN
  - ⇒ ρ is a nonlinear activation function (eg. softmax)

Time Gating

- What if a strongly/weakly correlated sequence becomes too long?
  - ⇒ Longer time dependencies get exponentially larger/smaller weights
- Gradients might vanish or explode ⇒ dependencies harder to encode
- Solution: time gating with input and forget gates $\alpha_t, \beta_t \in [0, 1]$
  - ⇒ Create paths through time where gradients are well-behaved
  - ⇒ Input gate α_t controls importance of input $\mathbf{x}_t$ at time t
  - ⇒ β_t decides how much to "forget" from previous state $\mathbf{h}_{t-1}$

$$\mathbf{h}_t = \sigma\Big(\alpha_t\mathbf{A}(\mathbf{S})\mathbf{x}_t + \beta_t\mathbf{B}(\mathbf{S})\mathbf{h}_{t-1}\Big)$$

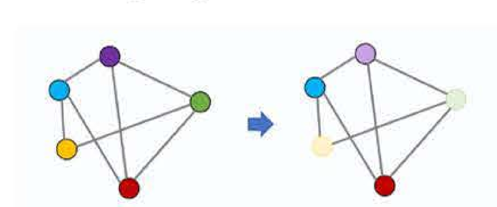
  - ⇒ Time-gated Graph Recurrent Neural Network (t-GRNN)

Node Gating

- Graphs allow for other forms of gating ⇒ spatial gates
- Node gating ⇒ α_t and β_t are vectors in $[0, 1]^N$

$$\mathbf{h}_t = \sigma\Big(\text{diag}(\alpha_t)\mathbf{A}(\mathbf{S})\mathbf{x}_t + \text{diag}(\beta_t)\mathbf{B}(\mathbf{S})\mathbf{h}_{t-1}\Big)$$

  - ⇒ One input and one forget gate for each node of the graph



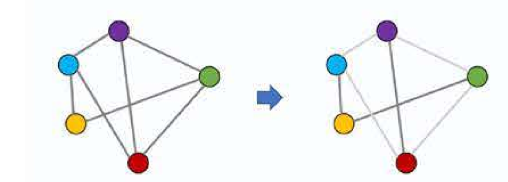
  - ⇒ Node-gated Graph Recurrent Neural Network (n-GRNN)

Edge Gating

- Edge gating ⇒ α_t and β_t are matrices in $[0, 1]^{N\times N}$

$$\mathbf{h}_t = \sigma\Big(\alpha_t \odot \mathbf{A}(\mathbf{S})\mathbf{x}_t + \beta_t \odot \mathbf{B}(\mathbf{S})\mathbf{h}_{t-1}\Big)$$

  - ⇒ One input and one forget gate for each edge of the graph



  - ⇒ Edge-gated Graph Recurrent Neural Network (e-GRNN)

Gate Computations

- Gates are calculated as the output of GRNNs themselves
- Time gating: GRNN + fully connected layer + sigmoid
  - ⇒ Fully connected layer maps state's features to a scalar
  - ⇒ Sigmoid ensures $\alpha_t, \beta_t \in [0, 1]$
- Node gating: GRNN + GNN + sigmoid
  - ⇒ GNN maps state's features to single-feature graph signal
  - ⇒ Sigmoid ensures $\alpha_t, \beta_t \in [0, 1]^N$
- Edge gating: GRNN + Graph Attention Network (GAT)
  - ⇒ Edge gates are attention coefficients a_{ij} of GAT

$$a_{ij} = \text{softmax}_j(e_{ij}) \qquad e_{ij} = a([\mathbf{W}[\mathbf{h}_t]_i, \mathbf{W}[\mathbf{h}_t]_j])$$

  - ⇒ a maps the features of $\mathbf{h}_t$ at nodes i and j to $e_{ij} \in \mathbb{R}$
  - ⇒ Softmax over j ensures $\alpha_t, \beta_t \in [0, 1]^{N\times N}$

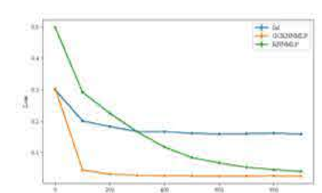
Performance Benchmark: 5-step Prediction

- 5-step noisy graph diffusion, where $\mathbf{w}$ is a Gaussian noise

$$\mathbf{x}_t = \mathbf{S}\mathbf{x}_{t-1} + \mathbf{w}_t$$

- Problem: predict $\mathbf{x}_5, \mathbf{x}_6, \mathbf{x}_7, \ldots$ from $\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots$
- GRNN compared with GNN and RNN with same number of parameters

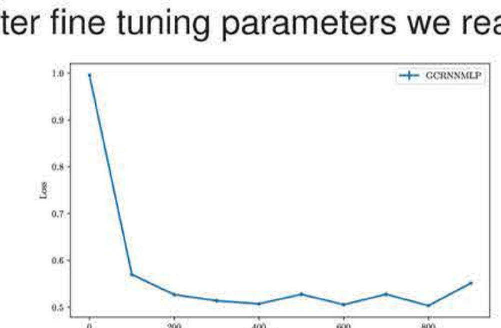
Archit.	Rel. RMSE
GRNN VS. GNN	11.39%
GRNN VS. RNN	19.95%

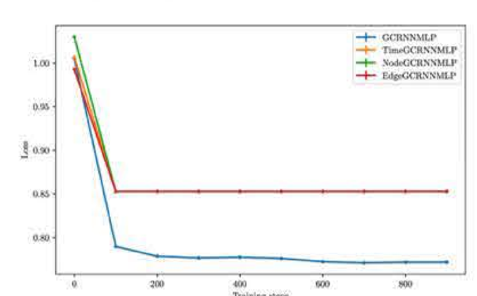


- GRNN outperforms GNN by > 10 p.p.
- GRNN and RNN achieve similar results ⇒ GRNN is a subcase of RNN
  - ⇒ But structure allows GRNN to learn faster than RNN

Influenza Case Estimation

- New York's 62 County's Commuting Network
- Dataset: 2010 Flu cases per county per four weeks
- Predict amount of cases per county in the following four weeks
- After fine tuning parameters we reached a 59%error

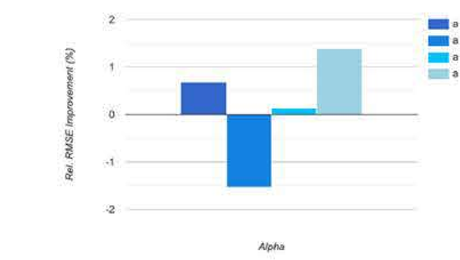




- The dataset did not have long time or distance dependencies, making a normal (non gated) GRNN more suited for the job

Performance Benchmark: Time Gating in AR(1) Process

- Noisy AR process with parameter a ⇒ $\mathbf{x}_t = a\mathbf{x}_{t-1} + \mathbf{w}_t$
- $0 < a \le 1$, $\mathbf{w}_t$ Gaussian noise, prediction 10 steps ahead
- GRNN compared with t-GRNN for multiple values of a



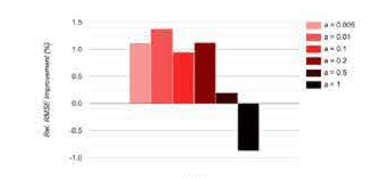
- Highly correlated process (large a) ⇒ $\mathbf{x}_t \approx \mathbf{x}_{t-1} \approx \ldots \approx \mathbf{x}_{t-10}$
  - ⇒ Memory within the process ⇒ $\mathbf{x}_t$ is more informative than $\mathbf{h}_t$
  - ⇒ Gates $\alpha_t \uparrow 1$ and $\beta_t \downarrow 0$ help improve performance

Performance Benchmark: Spatial Gating

- Node gating: graph diffusion process with $\mathbf{S}^a$

$$\mathbf{x}_t = \mathbf{S}^a\mathbf{x}_{t-1} + \mathbf{w}_t$$

- $0 < a \le 1$, $\mathbf{w}_t$ Gaussian, 10 steps ahead
- GRNN compared with n-GRNN



- Large a ⇒ $\mathbf{x}_t$ is more informative ⇒ effect of $\beta_t \downarrow 0$
- Small a ⇒ $\mathbf{h}_t$ is more informative ⇒ effect of $\alpha_t \downarrow 0$
- Performance gains are more substantial the mid-range
  - ⇒ The effect of both the input and forget gates can be perceived

Conclusions

- Introduced Graph Recurrent Neural Networks
  - ⇒ Tailored to problems involving graph processes
  - ⇒ Exploit graph structure through graph convolutions
  - ⇒ Exploit sequential data through state recurrence
  - ⇒ Gating allows encoding long-term dependencies
- Observed performace improvements
  - ⇒ Synthetic dataset ⇒ K-step prediction

References

- L. Ruiz et al., "Gated Graph Convolutional Recurrent Neural Networks," arXiv:1903.01888v3 [cs.LG], June 2019.
- F. Gama et al., "Convolutional neural network architectures for signals supported on graphs," *Trans. Signal Process.*, 67 (4), 1034-1049, Feb. 2019.
- I. Goodfellow, Y. Bengio, and A. Courville, Deep Learning, The Adaptive Computation and Machine Learning Series. The MIT Press, Cambridge, MA, 2016.
- Earthquake Commission, GNS Science, and Land Information New Zealand, "GeoNet," https://www.geonet.org.nz/, 20 Feb. 2019.