Predicting Power Outages

- Power outages have huge economic cost
  - $22B and $135B annually (Campbell [12])
- Most caused by weather conditions (Period [15])
- Weather data can be expressed as a graph signal
  - Graph based on distance between weather stations

Convolvutional Networks for Graphs

- Existing CNNs → Remarkable performance in processing regular data
  - Convolution, pooling need a regular, multi-resolution domain
- Lots of data presents alternative irregular structural information
  - Especially, many problems in wireless systems (network = graph)

Graph Neural Networks (GNNs) generalize CNNs

- Convolution → Local shift-invariant graph filters
- Pooling → Local nonlinearity followed by downsampling

Convolutional Networks on Graphs

- Network structure → Graph matrix S (Adjacency, Laplacian L)
  - \( |S| = \text{Relationship between i and j underlying graph structure} \)
- Define a signal \( x \) on the top of the graph
  - \( x_0 \) = Signal value at node \( i \)
- Graph Signal Processing → Exploit structure encoded in \( S \) to process \( x \)
  - Generate features through (local) convolution and (local) pooling
  - Convolution is a linear shift invariant filter
    - \( y = \sum_{x \in G} S_{xi} x \)
  - Graph Signal Processing

Selection GNNs

- Bypass the need to generate new graphs
  - Achieved by downsampling and zero padding.
- Selection defined by matrix \( C_i \subset \{ 0, 1 \}^{K \times K} \)
  - Feature \( x_i \) is not supported on the same graph (smaller dimension)
  - Problem can be solved by remembering location of sampled nodes
- Place signal \( x_i \) on the original input graph
  - Zero-pad input features
  - This leads to multi-node aggregation, a hybrid between selection and aggregation

Selection CNNs

- Define a signal \( x \) on the top of the graph
  - \( x_0 \) = Signal value at node \( i \)
- Graph Signal Processing
- Convolution is a linear shift invariant filter
  - \( y = \sum_{x \in G} S_{xi} x \)

Pooling

- Pooling summarizes information in graph neighborhoods
- But beyond first layer there’s no graph connecting signal components
- Define pooling operations on \( S \) as well
  - Clustering (Dawson‘16)
  - Selection (Gama‘18)
  - Aggregation (Gama‘18)

Aggregation GNN

- Input \( \{ x_i \} \rightarrow \{ x_i \} = \{ x_i \}
- Convolution
- Pooling
- Output

Regular Convolution in Aggregation GNNs

- Input \( x_i \) is a signal over known \( N \)-node graph
- Select node \( p \in V \) → Perform N local exchanges
- Consecutive elements encode nearby neighbors
  - \( z^T(p, n) \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \)

- This resulting signal has a regular structure
- We can use a regular convolution
  - \( \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \)

- Effectively relates neighboring information encoded by the graph

- Therefore regular pooling and downsampling can be used as well
- \( N \) exchanges can be expensive → select a subset of nodes \( P \subset V \)

Multi-Node Aggregation GNNs

- Place signal \( x \) on the original input graph
  - Zero-pad input features
  - Therefore the convolution operation becomes
    - \( \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \rightarrow \{ x_i \} \)

- “Graph filter” using sampled k-shift matrices
  - \( S^k_j = D^{-j} S^k D^j \)

- Pooling. Get \( k \)-hop neighborhood using \( S^k_j \) for some \( k \leq r \)

Graph Neural Network Architectures

- Clustering
  - Multi-scale hierarchical clustering algorithm applied at each layer
  - Corresponding \( S \) has a lower dimension
  - More details can be found in [Dawson‘16]
- Selection
  - Linear shift invariant graph filters to build convolutional features
  - Pooling as subsampling → Remember sampled locations on graph
  - Use zero padding for convolutional features at hidden layers

Aggregation

- Successfully apply graph shift. Store observed values at one node
- Creates signal with time structure that incorporates graph topology
- We can now perform convolution and pooling on the time domain
- Multi-node version with outer layers using zero padding

Aggregation

- Define a signal \( x \) on the top of the graph
  - \( x_0 \) = Signal value at node \( i \)
- Graph Signal Processing
  - Convolution is a linear shift invariant filter
  - \( y = \sum_{x \in G} S_{xi} x \)

Pool

- Pooling summarizes information in graph neighborhoods
- But beyond first layer there’s no graph connecting signal components
- Define pooling operations on \( S \) as well
  - Clustering (Dawson‘16)
  - Selection (Gama‘18)
  - Aggregation (Gama‘18)

Input

- Convolution
- Pooling
- Output

Graph Neural Networks (GNNs) generalize CNNs

- Convolution → Linear shift-invariant graph filters
- Pooling → Local nonlinearity followed by downsampling

Convolvutional Networks for Graphs

- Existing CNNs → Remarkable performance in processing regular data
  - Convolution, pooling need a regular, multi-resolution domain
- Lots of data presents alternative irregular structural information
  - Especially, many problems in wireless systems (network = graph)

Dataset

- We use weather data to predict power outage events
- Considering Jan ‘11 - Dec ‘15
- NYC Weather data → Hourly logs from 123 weather stations → 26,304 datapoints
- NY electrical disturbance events → 25 such events occur
- Preprocessing needed to condense datasets
  - Some weather data was missing from the original dataset
  - Implement a greedy algorithm to select \( N = 25 \) stations
  - and a subset of datapoints with no missing datapoints
  - In total there are 5,777 datapoints (hours) → during 218 of which there was a major disturbance event
  - 10% of the dataset is randomly chosen as a test set → the rest is used for training and validation

Defining a Graph Signal

- Each weather station takes a variety of hourly measurements
  - We use pressure, temperature, wind speed, precipitation rate per hour, humidity, hourly rate per hour and precipitation rate
  - At each hour we consider a datapoint \( (x_k, y) \)
  - \( y = 1 \) if major electrical disturbance at that hour, otherwise \( y = 0 \)
  - \( x_k \) is the weather feature \( g \) at station \( i \)

- Define the graph shift operator \( S \) based on distance between stations

- We apply a Gaussian kernel to \( d(i, j) \) → distance between stations \( i, j \)

- Additionally a threshold, \( r_{max} \), is applied to \( S \)

Selected References


Numerical Experiments

- We compare the performance of the architectures against several baseline methods.
  - Neural network
    - We concatenate all features into one long vector of length \( N \times 7 \)
    - Perform a hyperparameter search
      - \( 1 \) FC layers, \( F \) hidden units at layer \( i \) and dropout \( d \)
    - Due to prevalence of negative labels in the dataset \( N \) converges to trivial solution
      - \( \text{The network would output 0 for all inputs} \)
    - We correct this by weighting positive labels more heavily: \( 10 \)
  - Affine space model (PCA)
    - We found using just pressure data results in highest performance
    - Estimate interclass mean \( \mu \) and covariance matrix \( \Sigma \)
    - We minimize the projection of the input on the class eigenvector matrix

Results

- We predict power outages in NY from weather data
- This is used to compare graph neural networks against baseline methods
  - Neural network and an affine space model
- No pooling yielded the best results with the highest F1 score
- However, all GNN architectures outperform the baseline methods

Conclusions and Future Work

- Graph neural networks outperform NN at this task
  - Simultaneously, they have far fewer parameters
- All architectures managed to perform better than the trivial solution
- The best performing architecture brings a 70% improvement in prediction error
- Successfully demonstrated the use of GNNs
- Other potential architectures such as Graph RNNs
  - Applications in machine translation