Learning from Relationships between Target Variables

Jaemin Yoo

Computer Science & Engineering

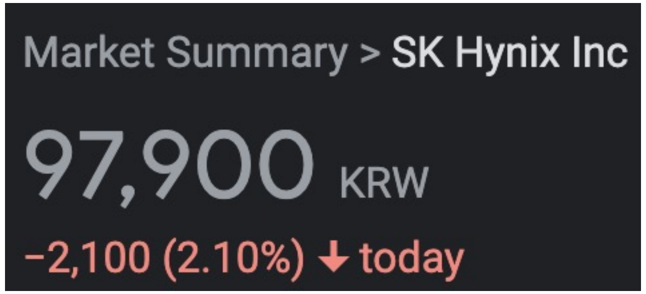
Seoul National University

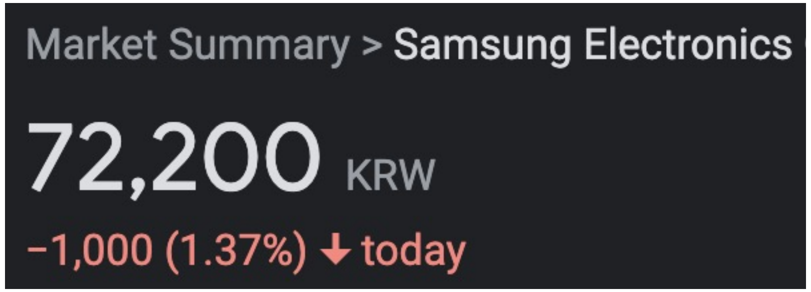
Outline

- **Introduction**
- Node classification
- Time series forecasting
- Conclusion

Relationships

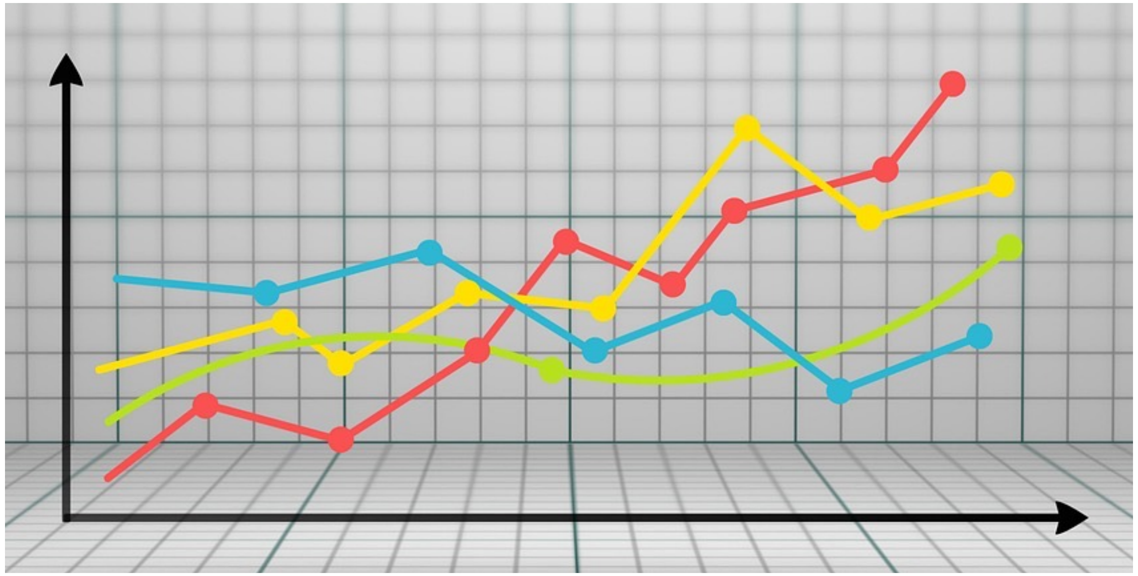
- Real-world variables are related to each other
- Such **relationships** are the key to enhance our understanding
- For example, consider the price movements of several stocks:





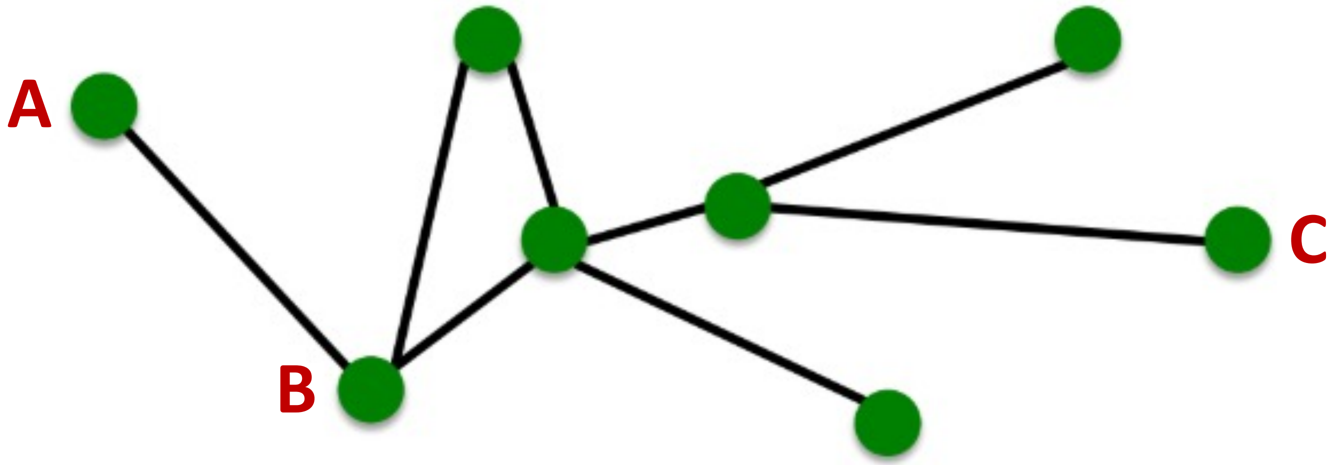
Multivariate Time Series

- Stock prices are representative data of **multivariate time series**
  - Target variables move together with strong correlations
  - Multivariate time series include weather, server usage, sales, …



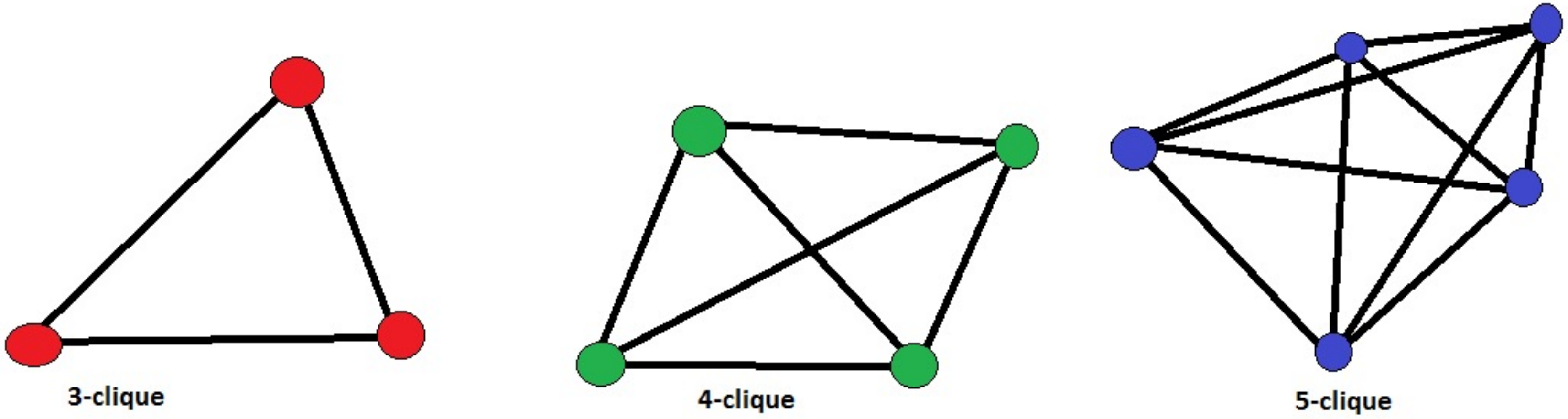
Graphs

- **Graphs** effectively represent the relationships between variables
- Consider a graph whose edges represent similarities
  - Indicates that variables A and B are similar, while A and C are different



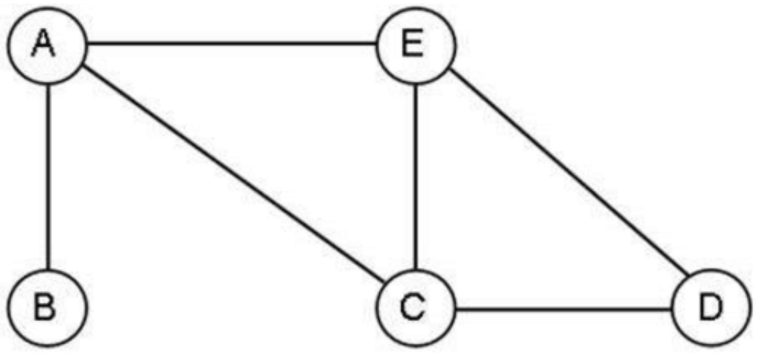
Sparsity of Graphs (1)

- *What if we don't know the exact relationships?*
- N variables make $\mathrm{O}(N^2)$ connections, making *clique* graphs
  - Clique graphs are **accurate** but provide **no information**



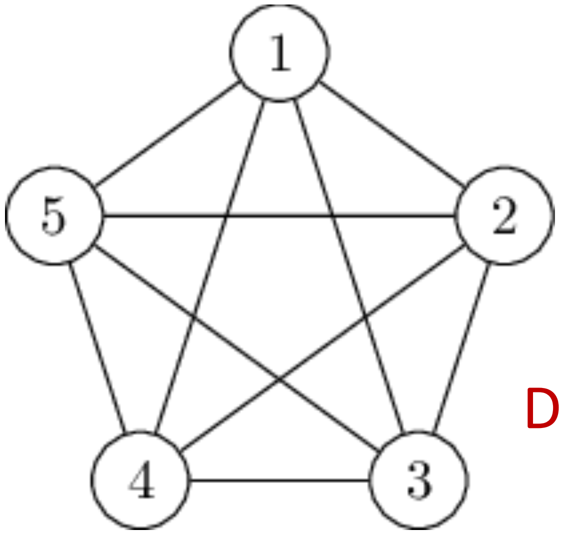
Sparsity of Graphs (2)

- It is the **sparsity** of graphs that gives us information
- A graph is more informative if it contains **fewer** edges
  - Sparsity leads to the (conditional) independence between variables



Density: 6/10 = 60%

VS.



Density: 100%

Research Goals

- My research is *learning from relationships between variables*
  - I believe that ALL target variables are connected
  - The difference is whether we know the **sparse** structure or not
- **Case 1.** If the graph is given, I use it to solve challenging tasks
- **Case 2.** If the graph is not given, I aim to learn it from data

Specific Problems

- I've studied the following problems related to my research goals:

- **Node classification**
  - **Given** a graph
  - **Goal** is to classify each node
  - **With** challenging constraints:
    1. *Cold-start learning*
    2. *Graph subsampling*
    3. *PU learning*

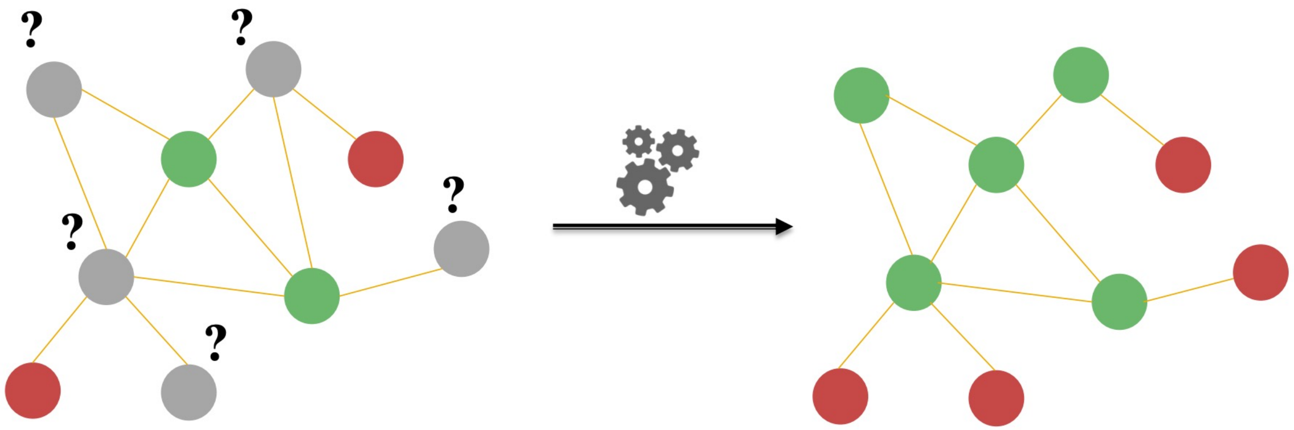
- **Time series forecasting**
  - **Given** a multivariate time series
  - **Goal** is to forecast future values
  - **With** learning the correlations:
    4. *Static attention map*
    5. *Multi-head Transformer*

Outline

- Introduction
- **Node classification**
- Time series forecasting
- Conclusion

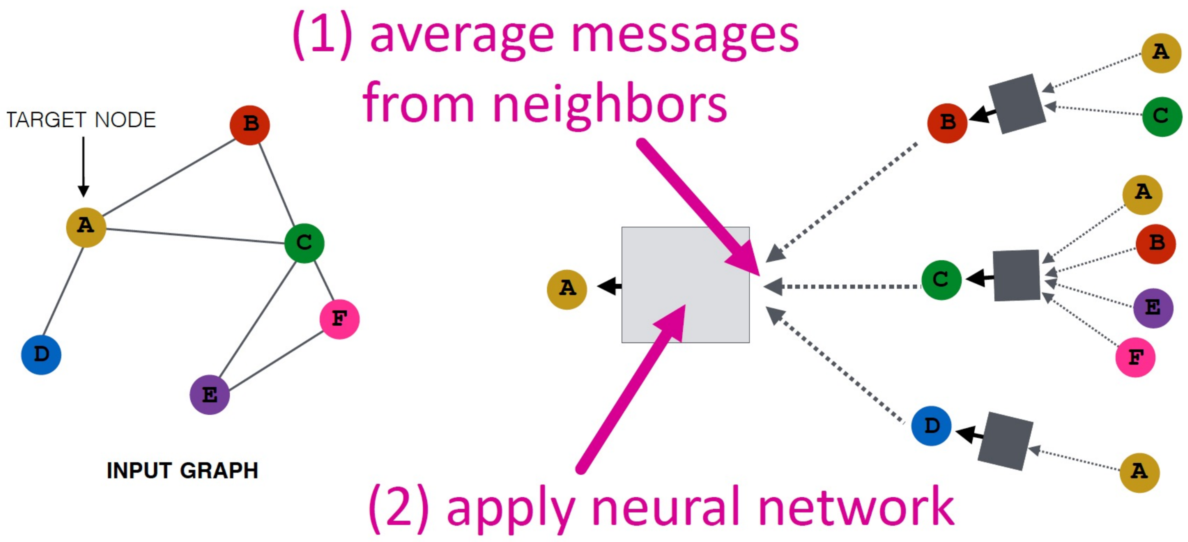
Node Classification

- **Node classification** is a popular problem in the graph domain
  - **Given** a graph $G = (V, E)$ and the feature vectors of all nodes
  - **Train** a classifier with the labels of training nodes
  - **Predict** the unknown labels of test nodes in the given graph G



Graph Neural Networks

- **Graph neural networks (GNN)** are specialized for graph data
  - Focus on combining node features and a graph structure
  - Make a computational graph following the graph structure

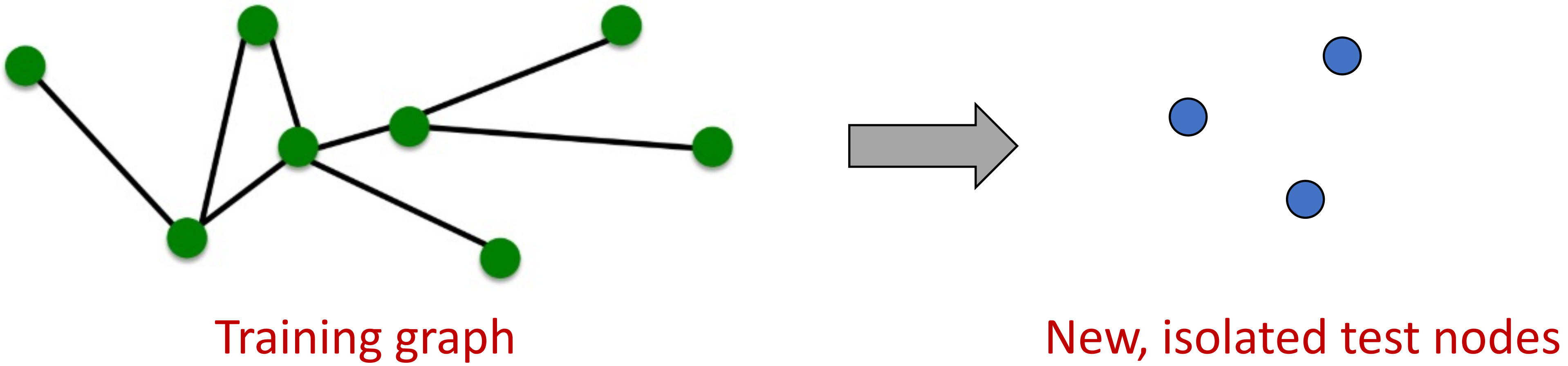


My Research on Graphs

- I’ve studied the generalization of GNNs into challenging tasks
- Specifically, I aim to answer the following questions:
  - **Q.** *How can we accurately classify new, isolated nodes?*
  - **Q.** *How can we find a small subgraph suitable for classification?*
  - **Q.** *How can we learn a GNN classifier without negative labels?*

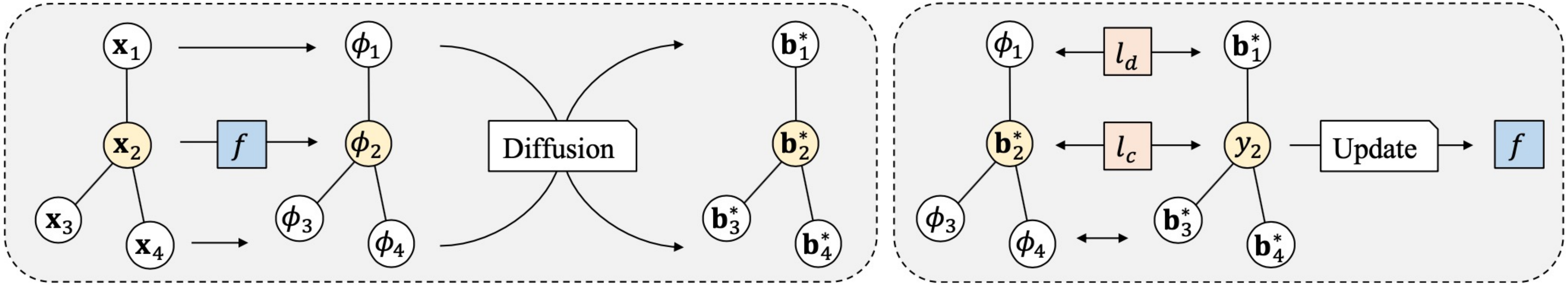
BPN [IJCAI-19]

- **Q.** *How can we accurately classify new, isolated nodes?*
- Existing GNNs perform badly if the test nodes are given in isolation
  - We call this problem **hard (cold-start) inductive learning**
  - Because GNNs mix node features and a graph without separation



BPN [IJCAI-19]

- The idea is the **separation** between *prediction* and *diffusion*
  - Predict the labels of nodes with a nodewise classifier f
  - Run graphical inference to *diffuse* the predictions
  - Update f using the diffused predictions as pseudo answers

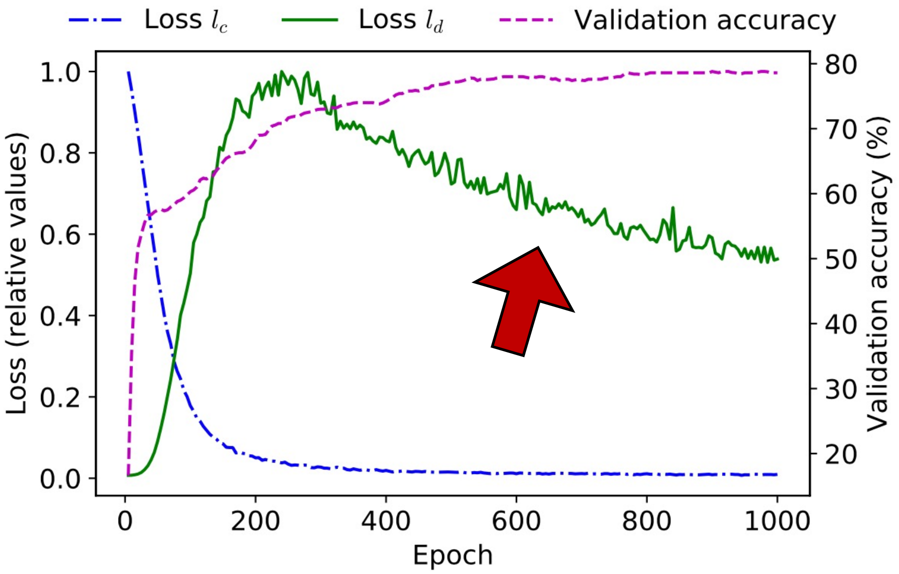


(a) Forward propagation of BPN.

(b) Backward propagation of BPN.

BPN [IJCAI-19]

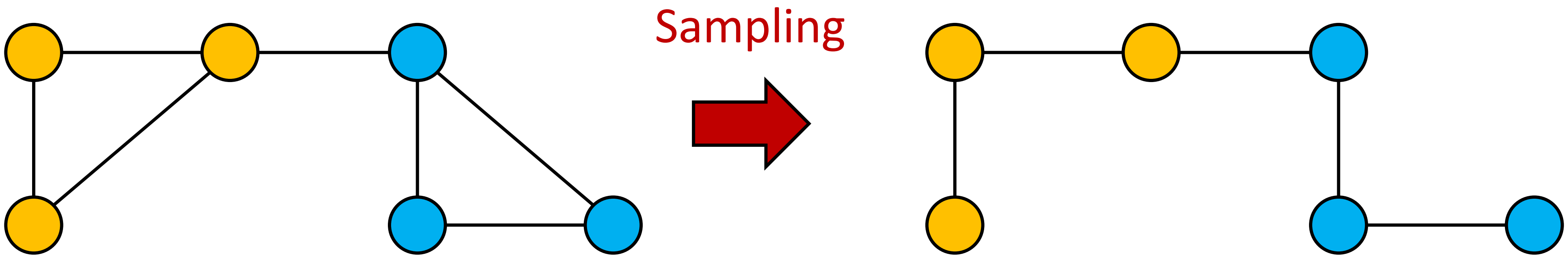
- The resulting classifier f requires no neighborhood at test time
  - Still, the training graph is fully leveraged in the training process
- Observe that *induction loss* increases at first and then decreases



Method	Pubmed	Cora	Citeseer	Amazon
Planetoid	74.6 ± 0.5	66.2 ± 0.9	66.8 ± 1.0	70.1 ± 1.9
GCN-I	74.1 ± 0.2	67.8 ± 0.6	63.6 ± 0.5	76.5 ± 0.3
SEANO	75.7 ± 0.4	64.5 ± 1.2	66.3 ± 0.8	78.6 ± 0.6
GAT	76.5 ± 0.4	70.1 ± 1.0	66.7 ± 1.0	77.5 ± 0.4
BPN (ours)	$\mathbf{78.3 \pm 0.3}$	$\mathbf{72.2 \pm 0.5}$	$\mathbf{70.1 \pm 0.9}$	$\mathbf{81.5 \pm 1.3}$

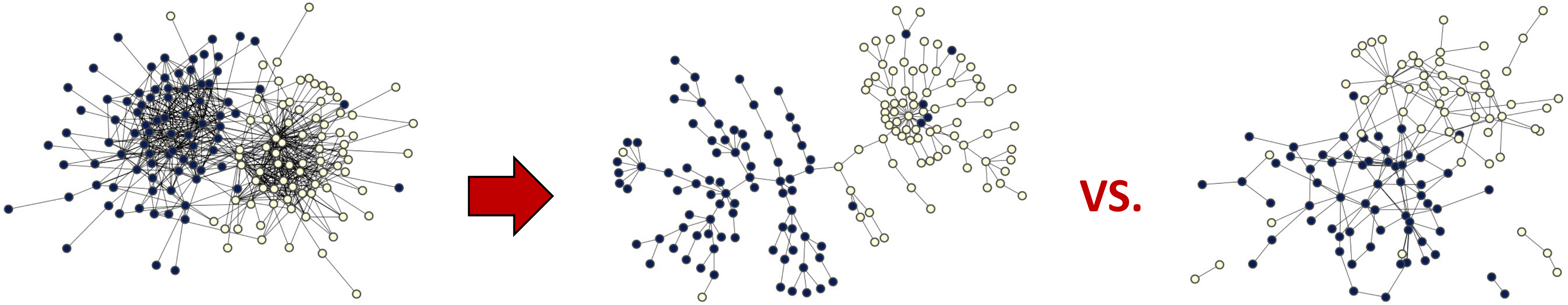
BTW [WSDM-20]

- **Q.** *How can we find a small subgraph suitable for classification?*
- **Motivation:** Not all edges are required for node classification
  - *Graph sampling* can improve both the accuracy and speed of inference



BTW [WSDM-20]

- The main idea is to sample a subgraph with **bounded treewidth**
- Low treewidth makes a graph with a simple, efficient structure
  - Still, it preserves the connectivity of the original graph



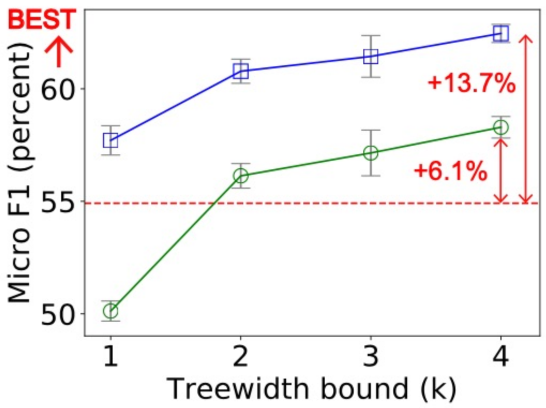
(a) Original graph. (TW $\approx$ 24)

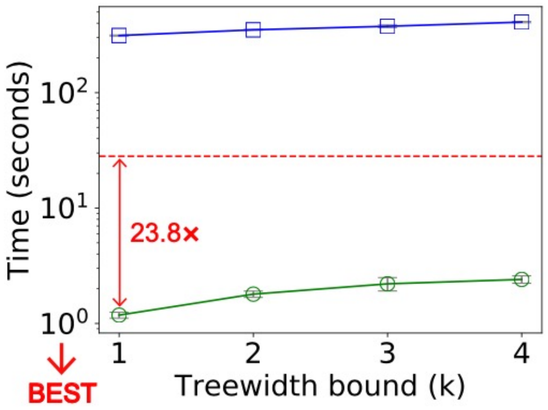
(d) Sampled subgraph by BTW. (TW = 1)

(b) Sampled subgraph by RE. (TW $\approx$ 8)

BTW [WSDM-20]

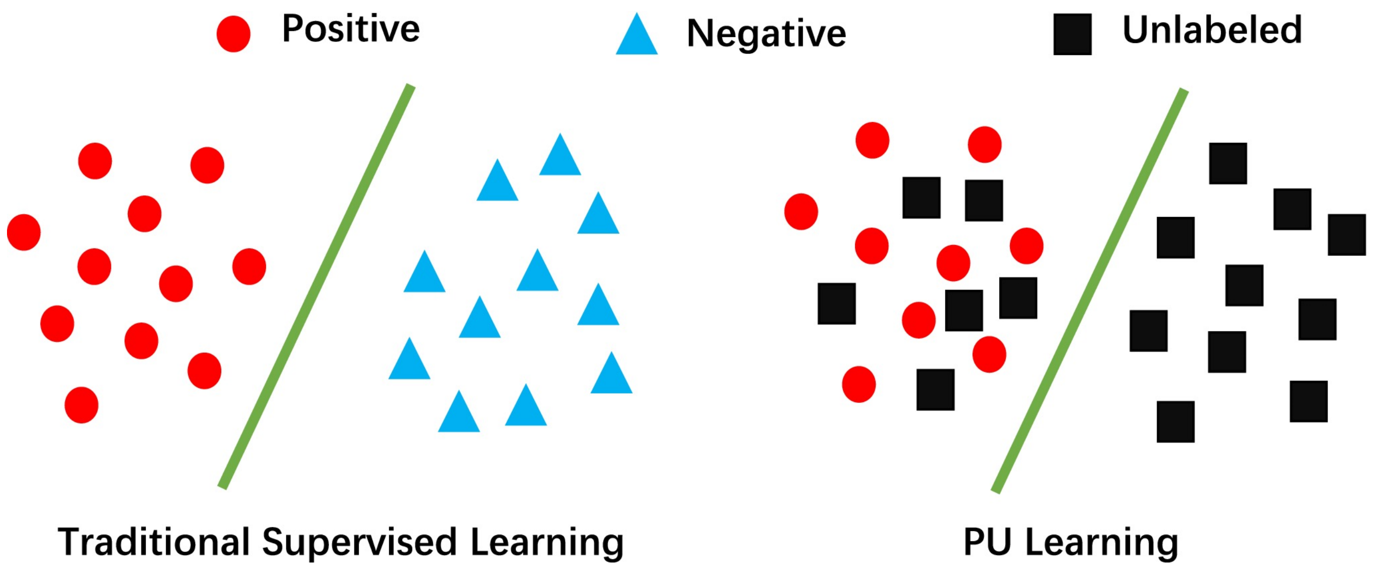
- We evaluate the subgraphs with two inference algorithms
  - **Baseline** is loopy belief propagation (BP) run on the original graph
- **The JT algorithm** (blue lines)
  - Runs exact inference
  - Improves accuracy of inference
- **Loopy BP** (green lines)
  - Much faster due to our sampling
  - Still, the accuracy is preserved well





GRAB [ICDM-21]

- **Q.** *How can we learn a GNN classifier without negative labels?*
- **Positive-unlabeled (PU) learning** is a challenging but essential task
  - We are given *positive* and *unlabeled* nodes, without negative labels



GRAB [ICDM-21]

- The main idea is to treat all unlabeled nodes as latent variables
  - We consider the given graph as a **Markov network**
  - Then, we estimate $p(\mathbf{z})$ and include it in the objective function:

$$\mathcal{L}(\theta; \mathbf{X}, \mathbf{y}, \mathcal{P}, \mathcal{U}) = \frac{1}{|\mathcal{P}|} \sum_{i \in \mathcal{P}} (-\log \hat{y}_i(+1))$$

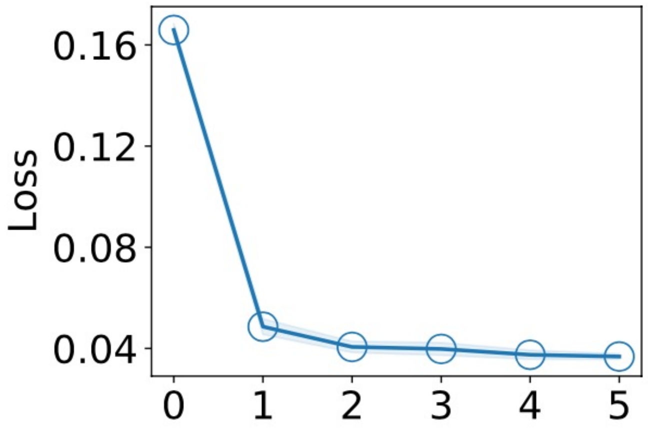
$$+ \mathbb{E}_{\mathbf{z} \sim p(\mathbf{z}|\mathbf{X}, \mathbf{y})} \left[\frac{1}{|\mathcal{U}|} \sum_{j \in \mathcal{U}} (-\log \hat{y}_j(z_j)) \right],$$

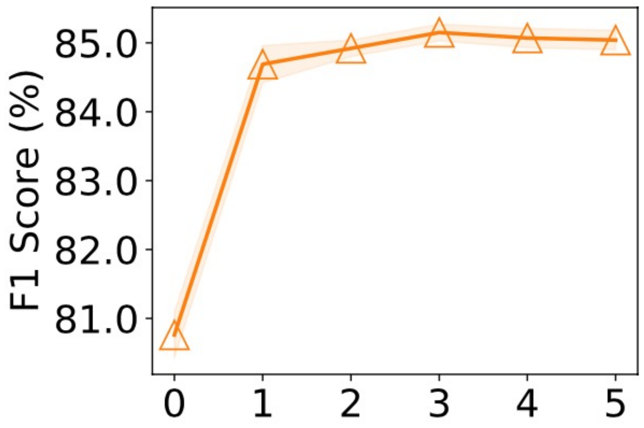
Predictions of a GNN classifier

Expectation over $p(\mathbf{z})$

GRAB [ICDM-21]

- We take an EM-like approach to minimize the objective function
  - **Expectation:** Estimate the marginals of Z from the prediction of f
  - **Maximization:** Update a GNN classifier f with the computed marginals
- GRAB improves the accuracy of f through iterative optimization



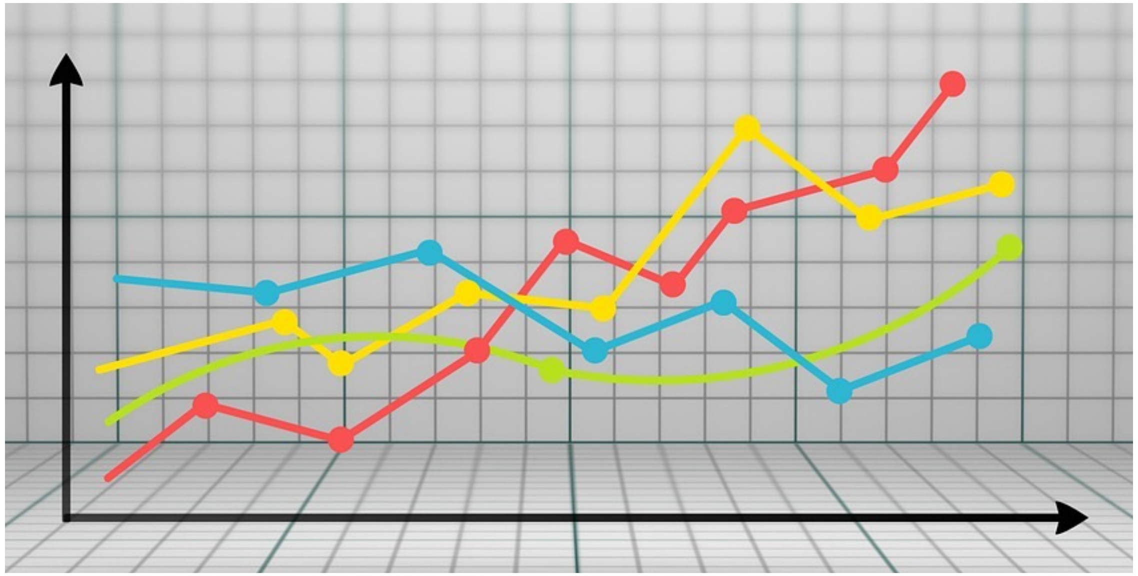


Outline

- Introduction
- Node classification
- **Time series forecasting**
- Conclusion

Time Series Forecasting

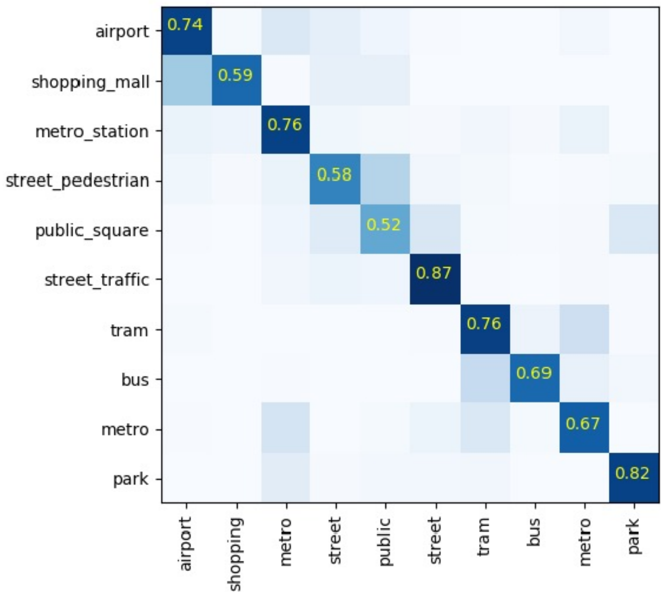
- Time series forecasting is a core problem related to many tasks
- We focus on **multivariate time series** data
  - There are a set of variables that are correlated with each other

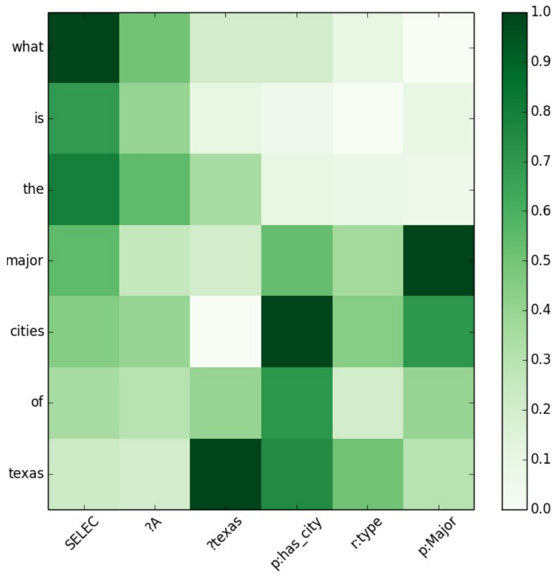


Future values?

Attention Mechanism

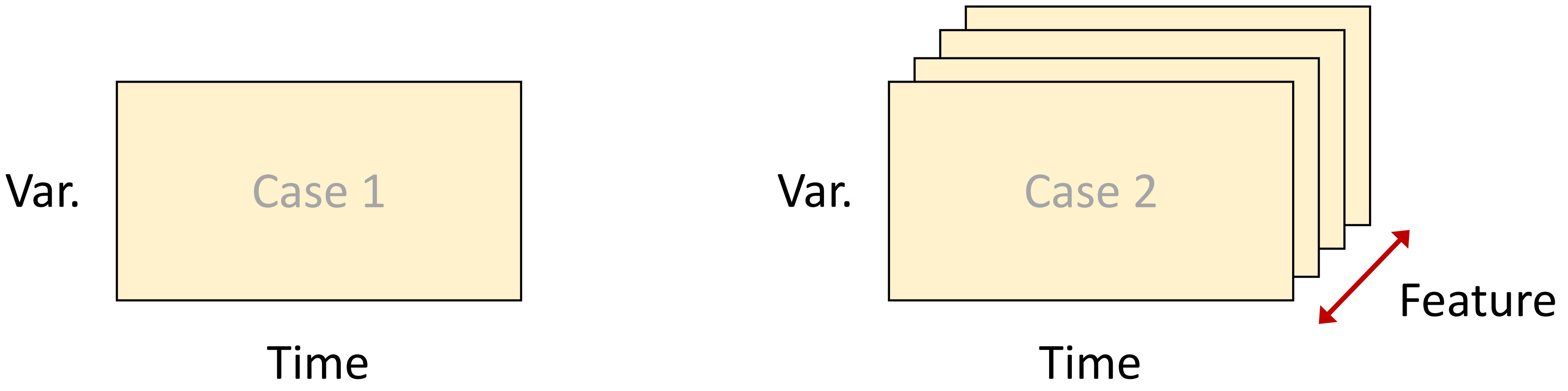
- **Attention** measures asymmetric correlations between variables
  - Makes an $N \times N$ correlation matrix having different weights
  - Different values of weights give structural information like sparsity





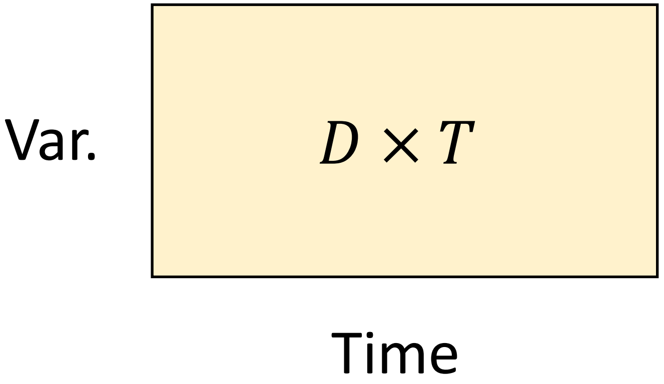
My Research on Time Series

- I’ve studied multivariate time series forecasting in two domains
  - 1. General data where each variable has one observation at a time
    - An input is a 2D matrix
  - 2. Stock price data where each variable has a feature vector at a time
    - An input is a 3D tensor



Attention AR [SDM-21]

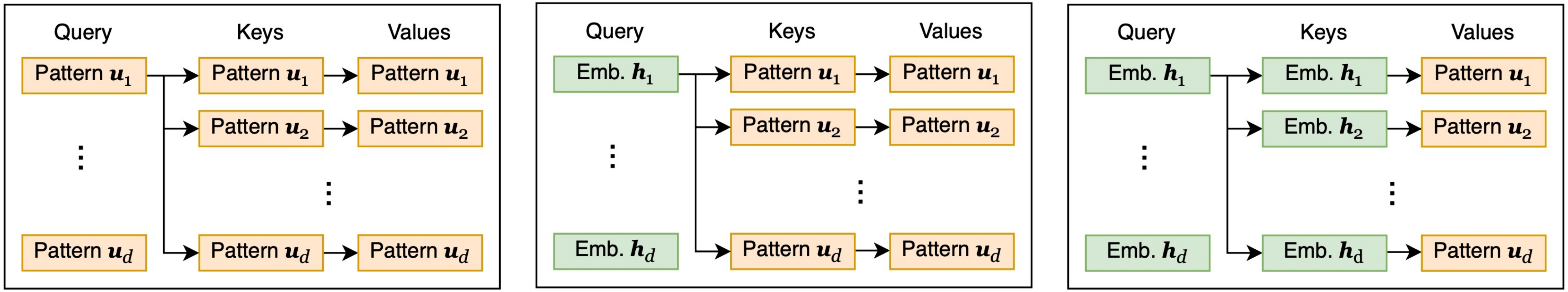
- **Q.** *How can we efficiently learn the correlation matrix?*
- **Challenge:** Multivariate models easily overfit to training data
  - **Why?** They use all D variables at the same time
  - This makes $D\times$ larger inputs and $D\times$ fewer training examples



- **Multivariate models**
  - 1 instance of size $D \times T$
- **Univariate models**
  - D instances of length T

Attention AR [SDM-21]

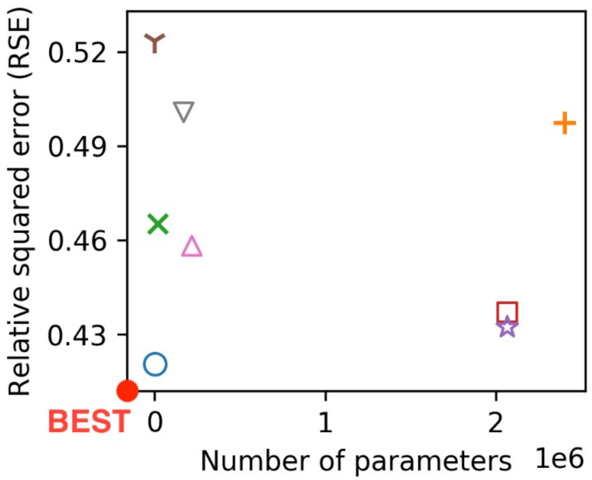
- Our idea is to minimize the model size using simple attention
- We compare three attention functions having different properties
  - **Time-invariant attention** works the best in all our datasets

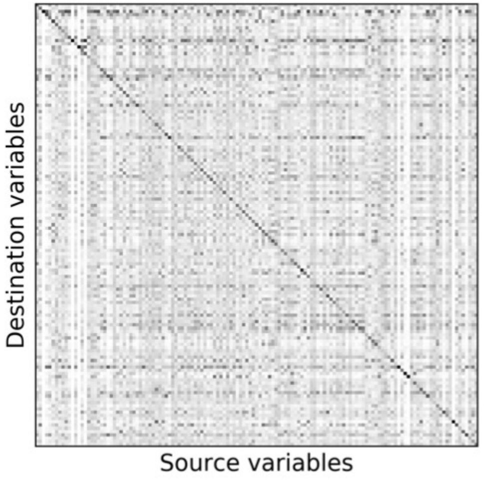


(a) Basic attention (Section 3.3.1). (b) Hybrid attention (Section 3.3.2). (c) Time-invariant attention (S. 3.3.3).

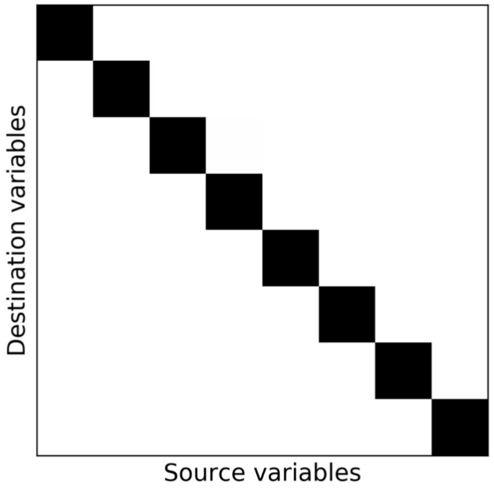
Attention AR [SDM-21]

- Our model makes the smallest error with the fewest parameters
- The learned correlation matrix gives an insight about the dataset





Correlations make high accuracy



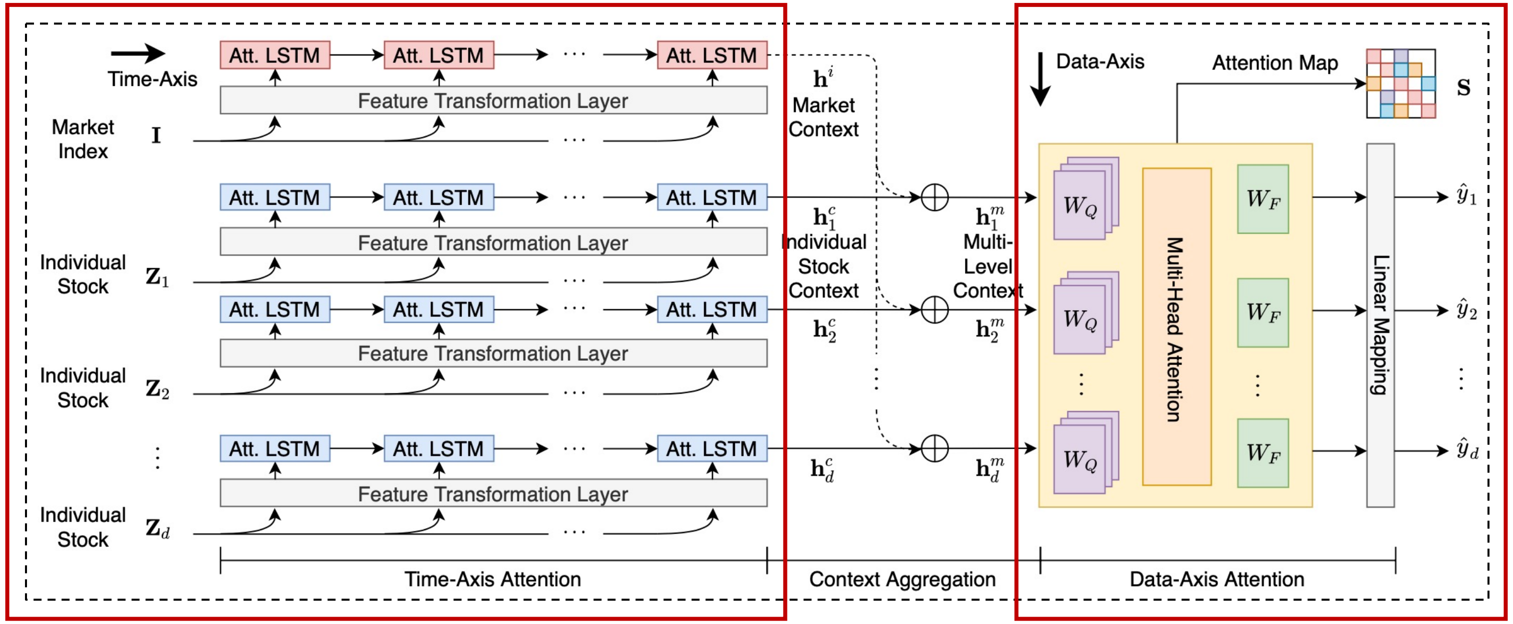
No correlations in some cases

DTML [KDD-21]

- **Q.** *How can we forecast the price movements of stocks?*
- We specialized the previous work into the financial domain
  - **Diff. 1.** Each variable has multiple features at each time step
  - **Diff. 2.** Data have no clear temporal patterns (such as repetition)
- We changed the problem into classification
  - **Binary classification:** To predict the movement of price into up/down

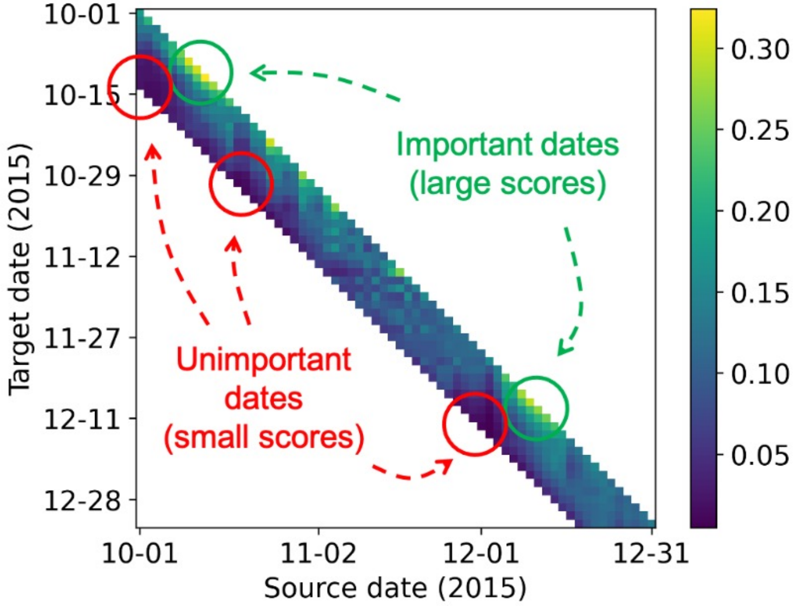
DTML [KDD-21]

- Our idea is to combine **temporal** and **spatial** attention modules
  - **Temporal:** attention LSTM that focuses on recent observations
  - **Spatial:** multi-head Transformer that models rich stock correlations

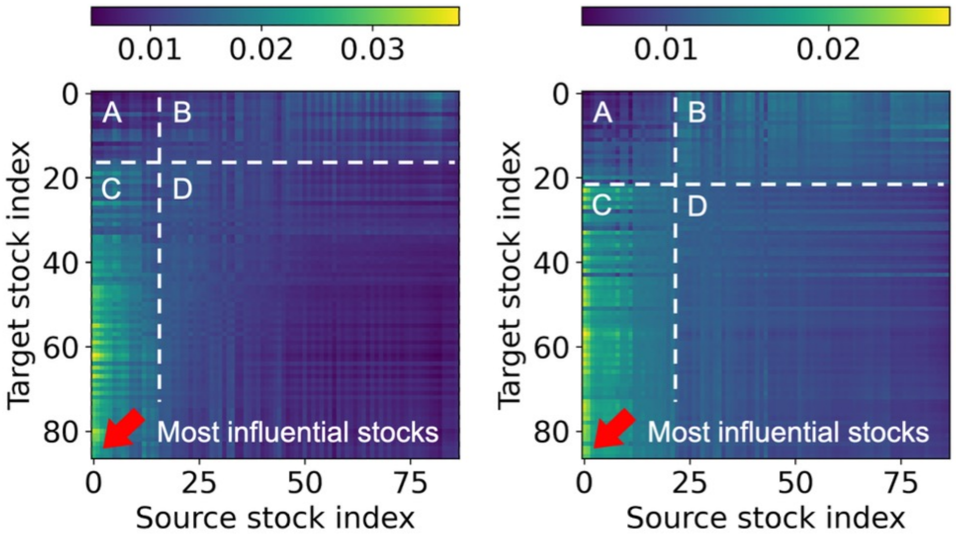


DTML [KDD-21]

- Our attention modules work well *between dates* and *between stocks*
- Temporal attention map:



- Spatial attention map:



(a) 2015-10-01 (b) 2015-10-30

Outline

- Introduction
- Node classification
- Time series forecasting
- <u>**Conclusion**</u>

Conclusion

- I’ve introduced my previous works on two research fields
  - **Node classification**: inductive learning, graph sampling, PU learning
  - **Time series forecasting:** general multivariate data, stock prices
- My goal is to utilize relationships between variables in **all** tasks
  - I believe that ALL variables are connected to each other
  - I love working on new, challenging tasks that are underexplored

Future Work

- I'm currently doing active research in the following topics:
    - **Topic 1.** Graph-based generative learning
        - ***Q.*** *How can we estimate missing node features in a graph?*
    - **Topic 2.** Graph augmentation
        - **Q.** *How can we augment graphs for accurate graph classification?*
    - **Topic 3.** Temporal graph structure learning
        - **Q.** *How can we learn a sparse temporal graph from time series?*

References

- **Node classification**
  - [1] Jaemin Yoo et al., “Belief Propagation Network for Hard Inductive Semi-Supervised Learning”, **IJCAI 2019**
  - [2] Jaemin Yoo et al., “Sampling Subgraphs with Guaranteed Treewidth for Accurate and Efficient Graphical Inference”, **WSDM 2020**
  - [3] Jaemin Yoo et al., “Accurate Graph-Based PU Learning without Class Prior”, **ICDM 2021**
- **Time series forecasting**
  - [4] Jaemin Yoo and U Kang, “Attention-Based Autoregression for Accurate and Efficient Multivariate Time Series Forecasting”, **SDM 2021**
  - [5] Jaemin Yoo et al., “Accurate Multivariate Stock Movement Prediction via Data-Axis Transformer with Multi-Level Contexts”, **KDD 2021**

Thank you!

Email: jaeminyoo@snu.ac.kr

Homepage: https://jaeminyoo.github.io