Graph-Guided Networks for Complex Time Series

Marinka Zitnik

Department of Biomedical Informatics
Broad Institute of Harvard and MIT
Harvard Data Science

Joint work with X. Zhang, M. Zeman, and T. Tsiligkaridis

zitniklab.hms.harvard.edu

HARVARD MEDICAL SCHOOL

HDSI | Harvard Data Science Initiative

BROAD INSTITUTE

Complex time series

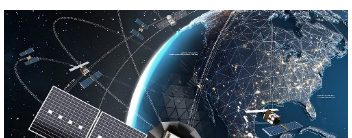
Multivariate time series are prevalent, including in healthcare, biology & climate science

Often irregularly sampled with varying time intervals between successive readouts and different sensors observed at different time points

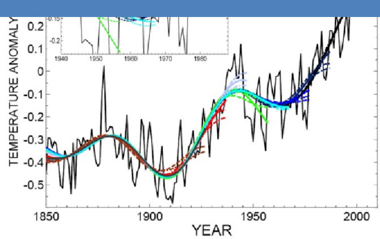
Climate

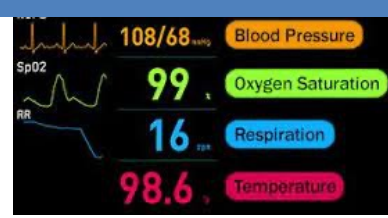
Healthcare

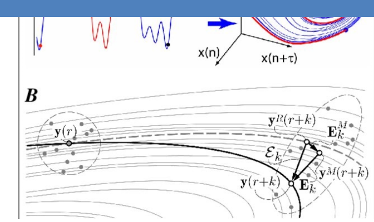
Space systems



It is critical to develop time series learning methods that are adaptive and flexible

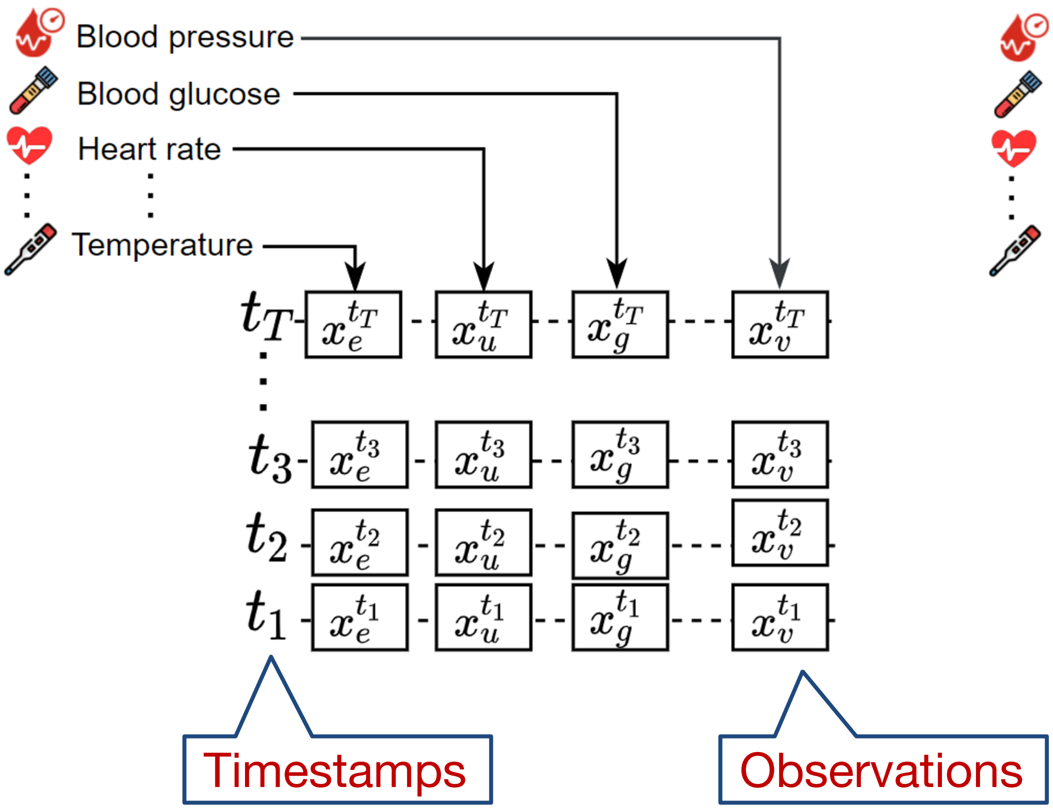




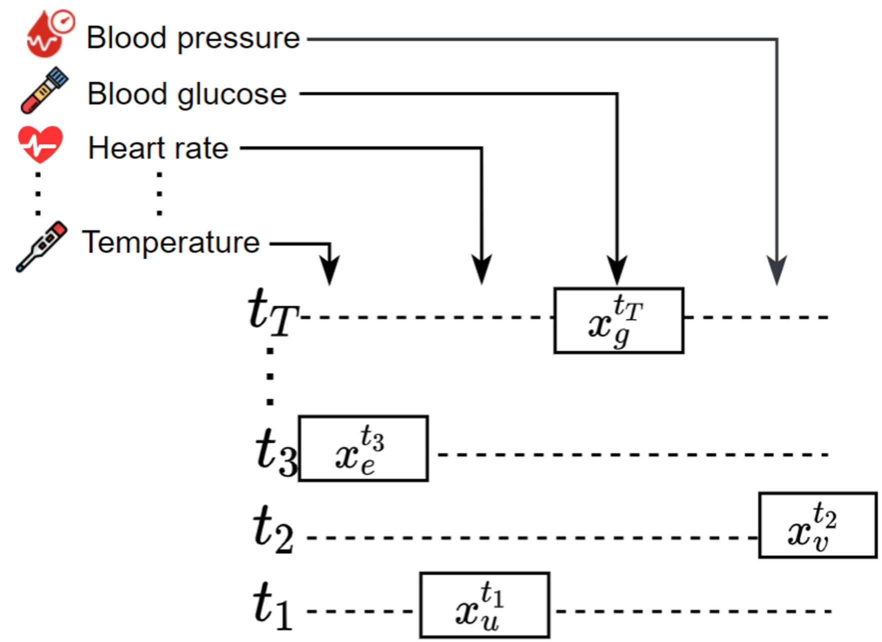


Irregular vs. regular time series

Regular time series



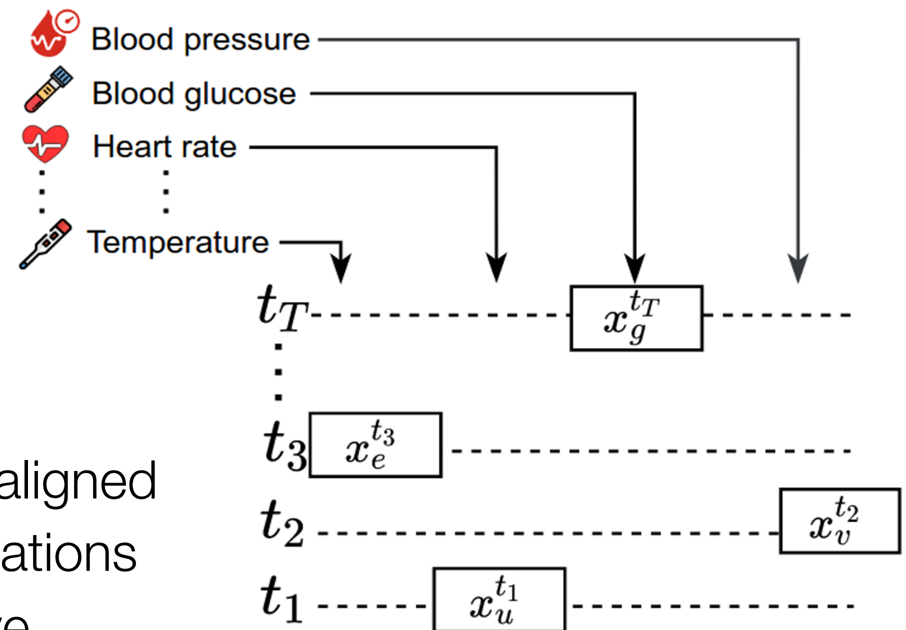
Irregular time series



Why are irregular time series challenging?

Prevailing ML methods:

- Assume aligned measurements
- Assume fixed-sized input data
- Impute or fill-in missing values



Irregular time series:

- Observations across sensors are not aligned
- Varying times among adjacent observations
- Arbitrary length: different samples have varying number of observations
- Different subsets of sensors recorded at different time points

Plan for today

- Motivation for Raindrop

- Hierarchical learning of irregular time series
  - Constructing sensor dependency graphs
  - Generating embeddings of observations
  - Generating sensor embeddings
  - Generating sample embeddings
- New datasets and experiments

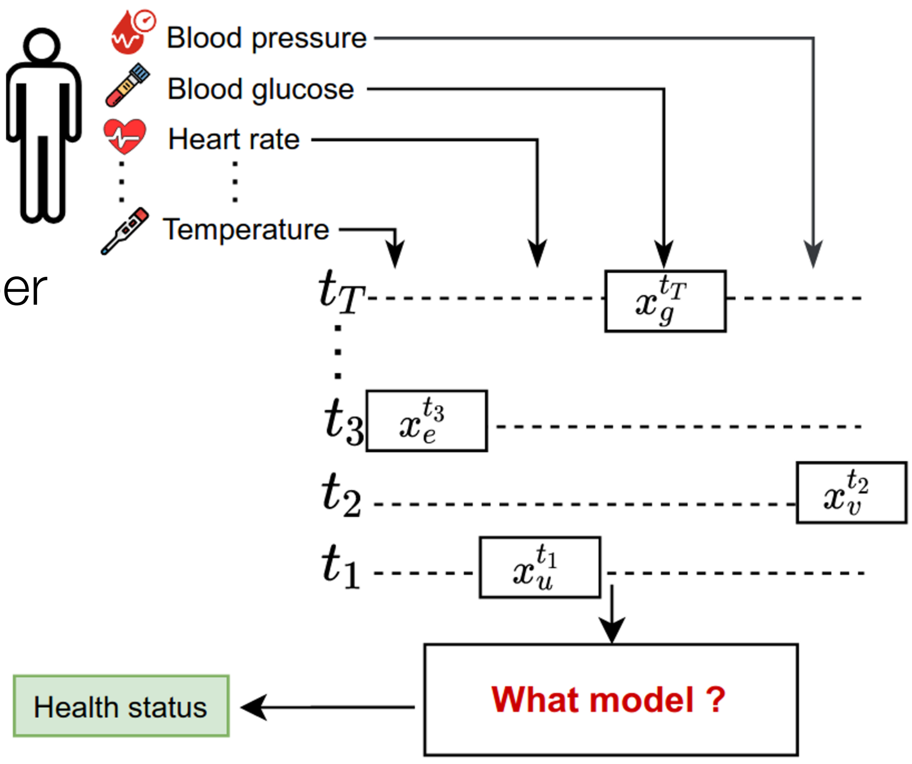
Today: Flexible approach for learning on irregular time series

Inputs:

- Dataset of samples, e.g., sample =
- Each sample can have many sensors
- Each sensor can have arbitrary number of irregularly sampled observations

Outputs:

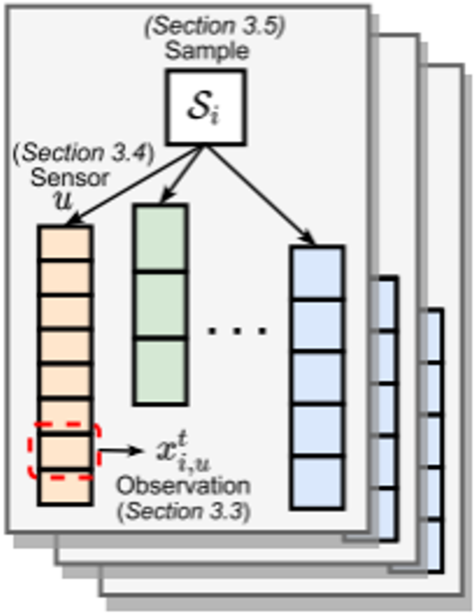
- Sample embeddings
- Sensor embeddings
- Observation embeddings
- Predictors of sample-specific labels



Problem definition

Input

- Dataset D of irregularly time series samples
- Every sample S_i can have multiple sensors
- Every sensor can have arbitrary number of irregularly sampled observations/readouts



- Raindrop learns a function $f: S_i \rightarrow \mathbf{z}_i$ that maps S_i to a fixed-length representation z_i suitable for downstream tasks of interest, such as classification
- Using learned z_i, one can predict label $\hat{y}_i \in \{1, \dots, C\}$ for S_i

Tackle irregularity by leveraging inter-sensor dependencies

- Inter-sensor dependencies bring rich information to time series modeling
- We integrate recent advances in GNNs to fully take advantage of relational structure among sensors
  - We learn latent graph structures from multivariate time series and model time-varying inter-sensor dependencies through neural message passing
  - First to use GNNs to model sample-varying and time-varying relational structure in irregular time series

Next: What motivates the use of inter-sensor dependencies?

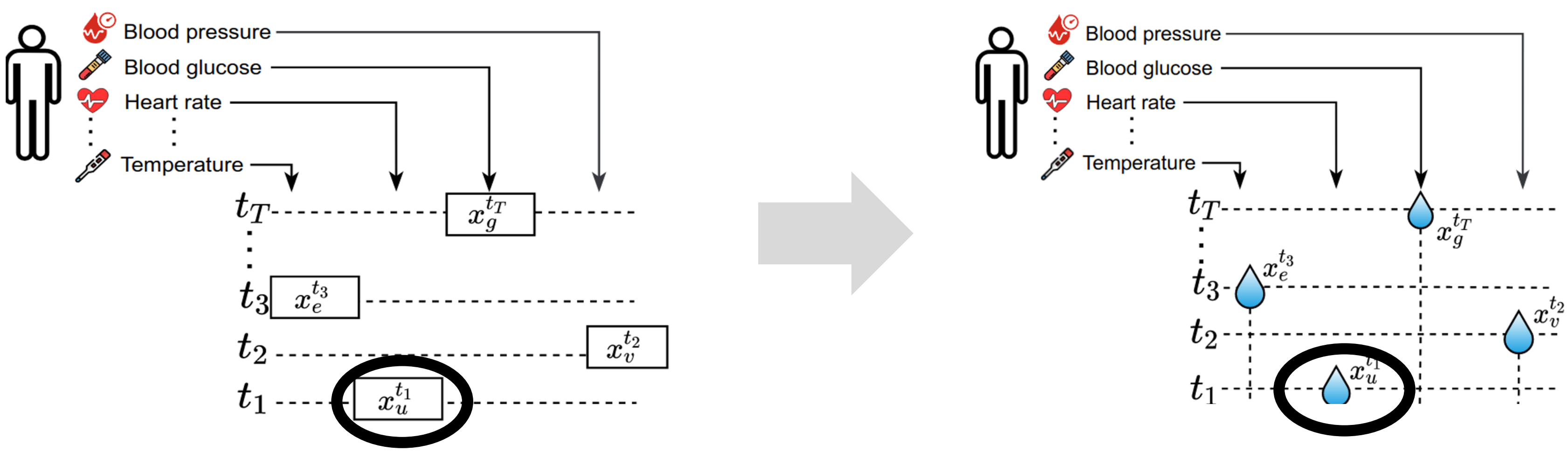
Motivation

Raindrop is inspired by how raindrops hit a surface at varying times & create ripple effects that propagate through the surface

- Observations (i.e., raindrops) hit the sensor graph (i.e., surface) asynchronously and at irregular time intervals
- Every observation is processed by passing messages to neighboring sensors (i.e., creating ripples), taking into account the learned sensor dependencies

Raindrop: Irregular observations as “raindrops” hitting a “surface”

- **Observations** (i.e., raindrops) hit the **sensor graph** (i.e., surface) asynchronously and at irregular times
- Observations are processed by **passing messages to neighboring sensors** (i.e., creating ripples), taking into account the learned sensor dependencies

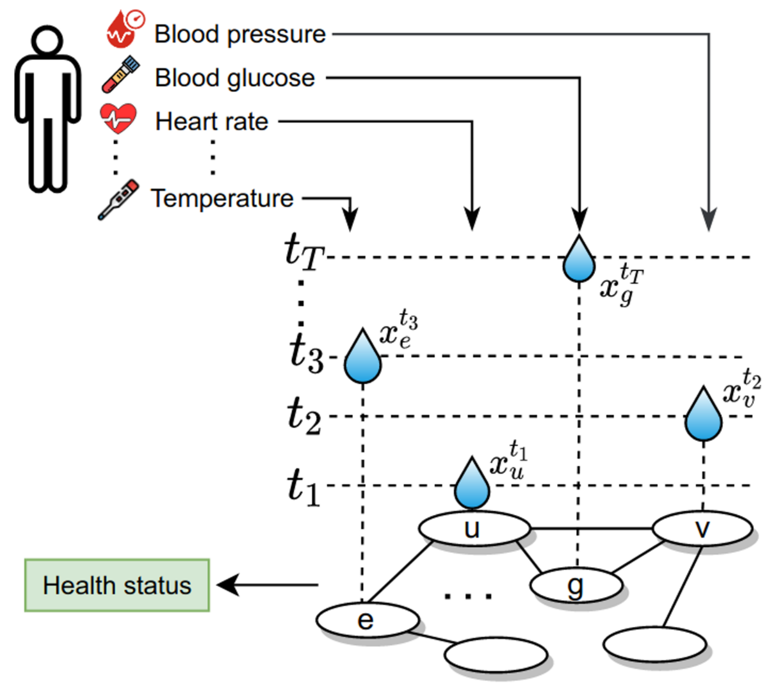


Raindrop: Irregular observations as “raindrops” hitting a “surface”

Raindrops
- Observations

Surface
- Sensor dependency graph

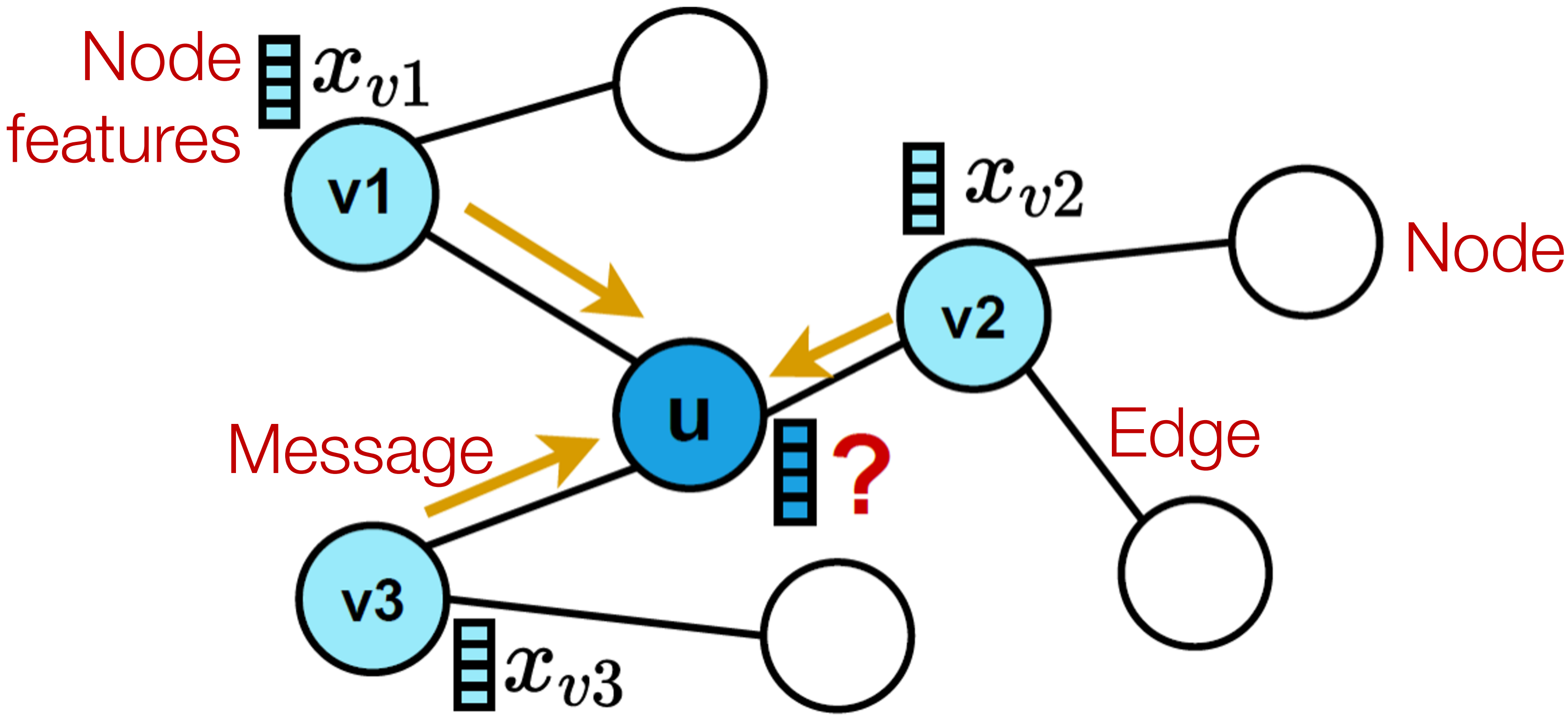
Ripples:
- Neural message exchanged between neighboring sensors within each sample



Sensor dependency graph

Next: How to learn inter-sensor dependencies that can vary across samples and time?

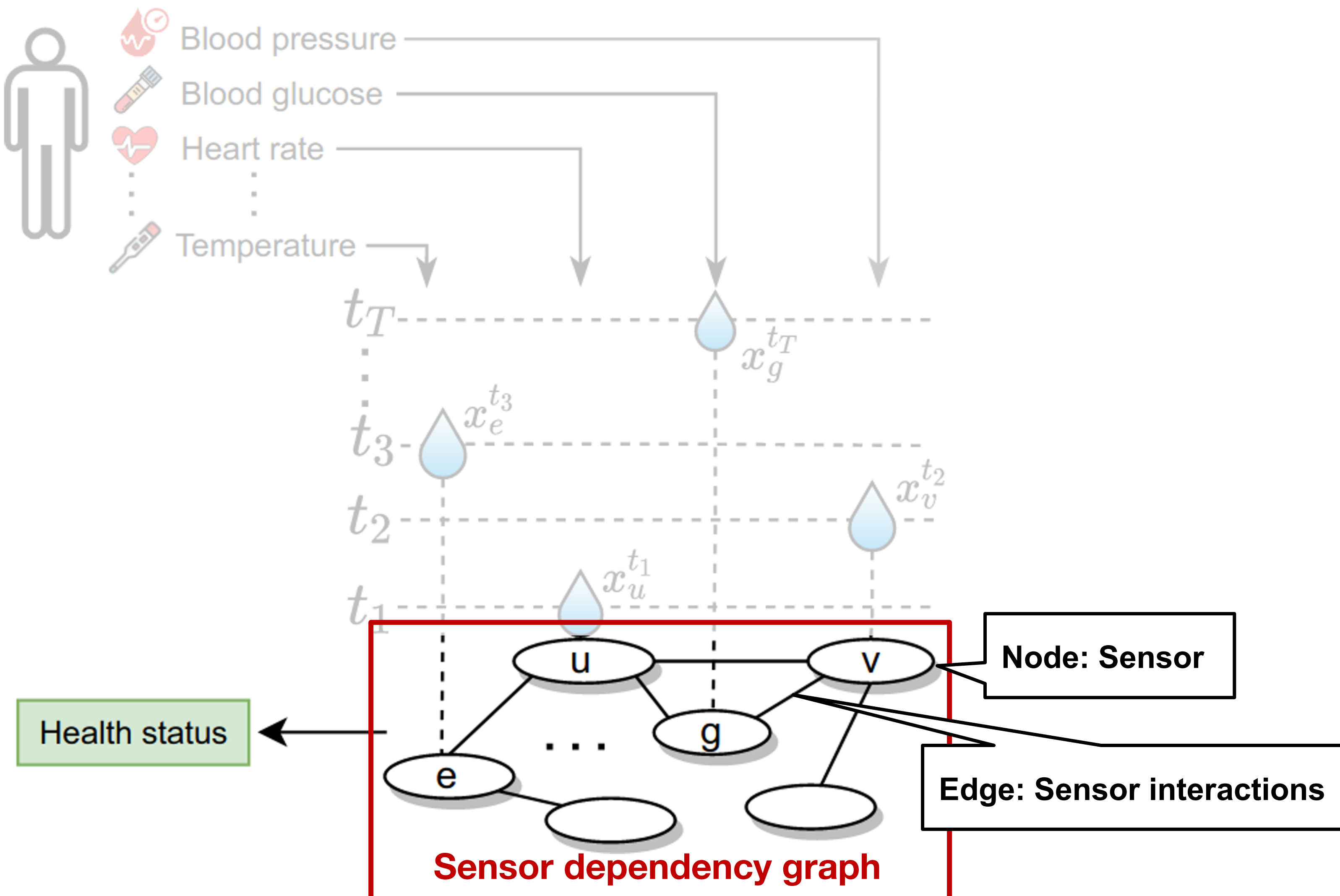
Sensor dependency graphs



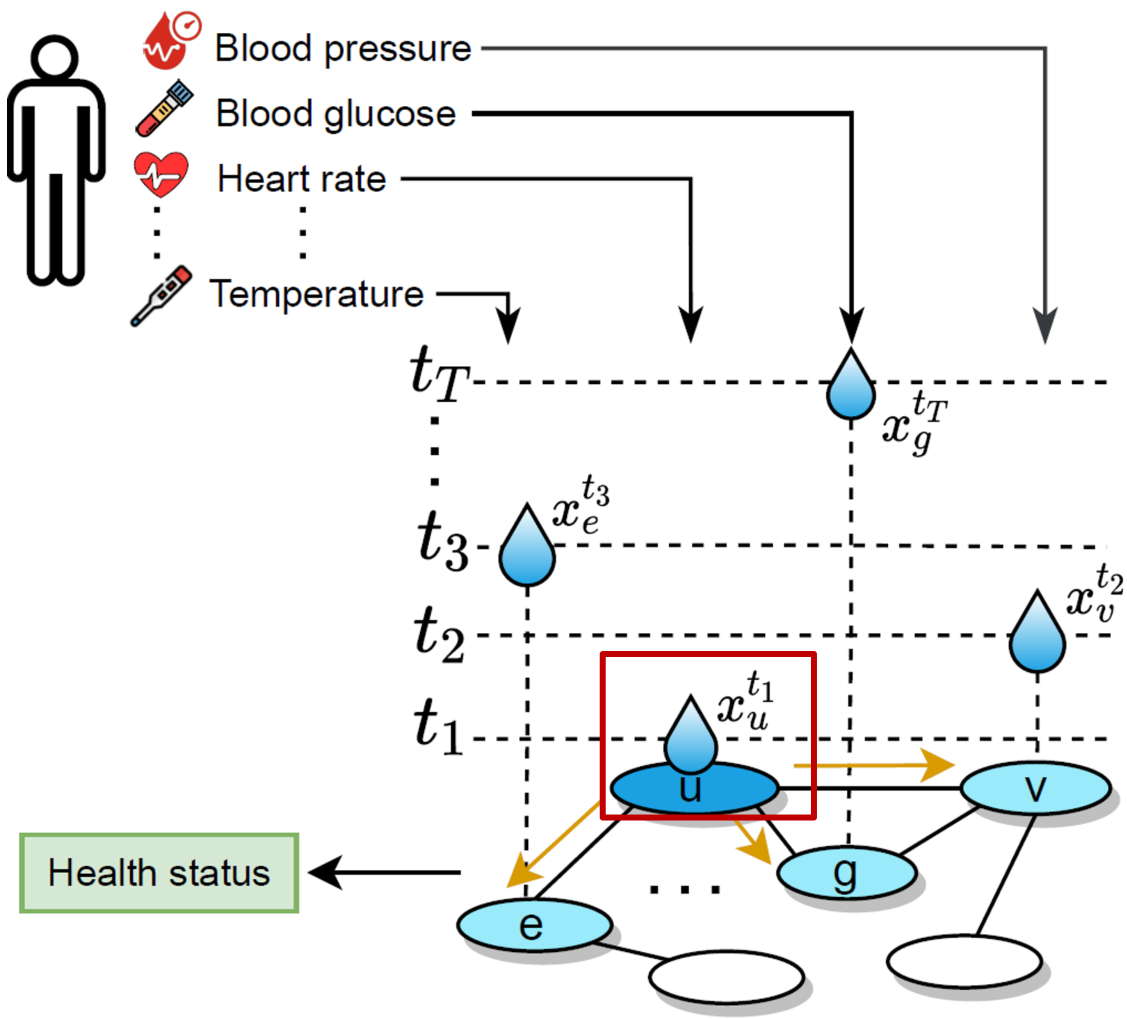
$$x_u = f(x_{v1}, x_{v2}, x_{v3})$$

Generate embedding of node *u* by capturing node dependencies through message passing

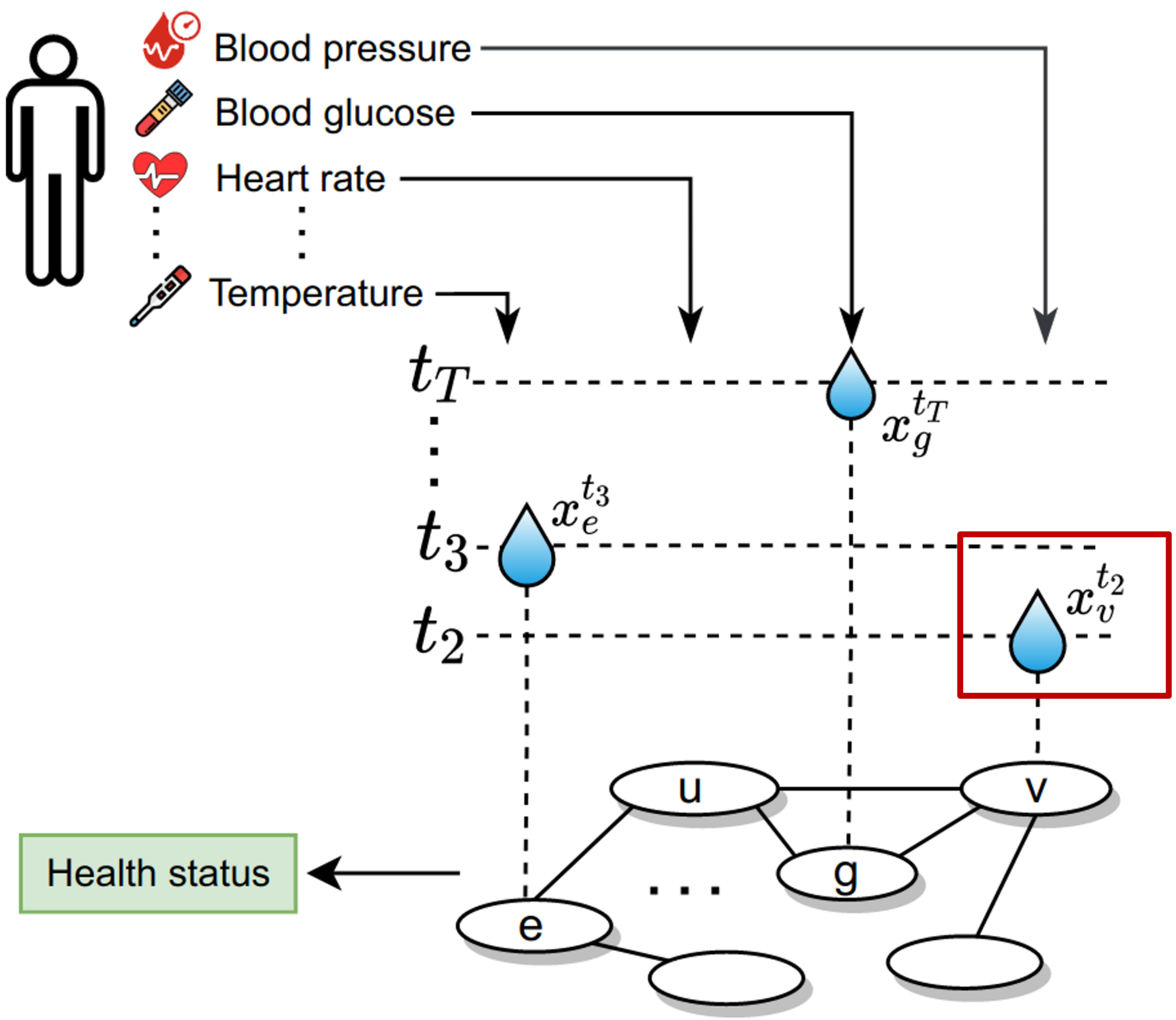
Sensor dependency graphs



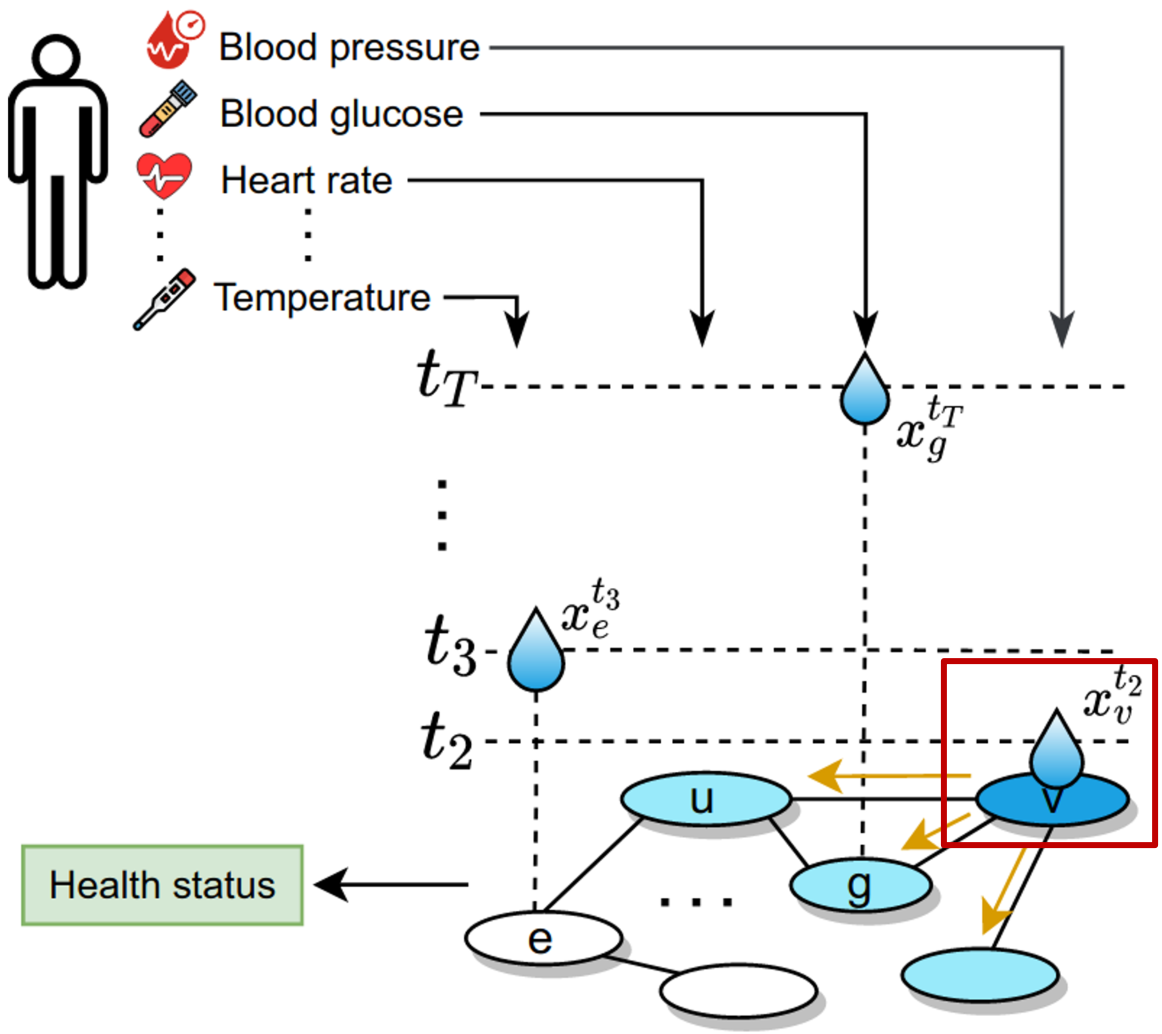
Passing messages between neighboring sensors in every sample



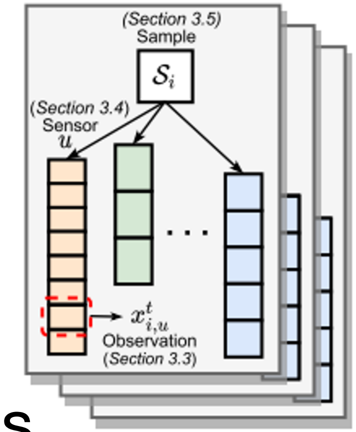
Passing messages between neighboring sensors in every sample



Passing messages between neighboring sensors in every sample



Overview of Raindrop model



Hierarchical learning of irregular time series:

- Step 1: Construct sensor dependency graphs
- Step 2: Generate embeddings of observations
- Step 3: Generate sensor embeddings
- Step 4: Generate sample embeddings

Step 1: Construct sensor dependency graphs

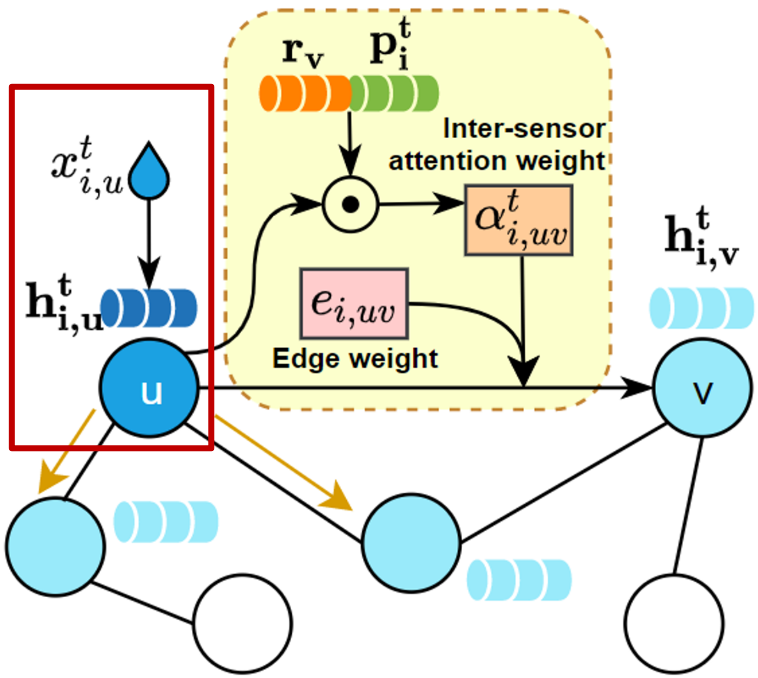
- Build a **directed weighted graph for each sample**

- Initialize as fully-connected graphs:
  - Can integrate additional domain knowledge

- During training, update neighbors & edge weights:
  - Graphs are **time-sensitive**
  - Graph are **sample-sensitive**
  - Similar graph for similar samples

Step 2: Embed observations

Directly learn observation embedding for active sensor



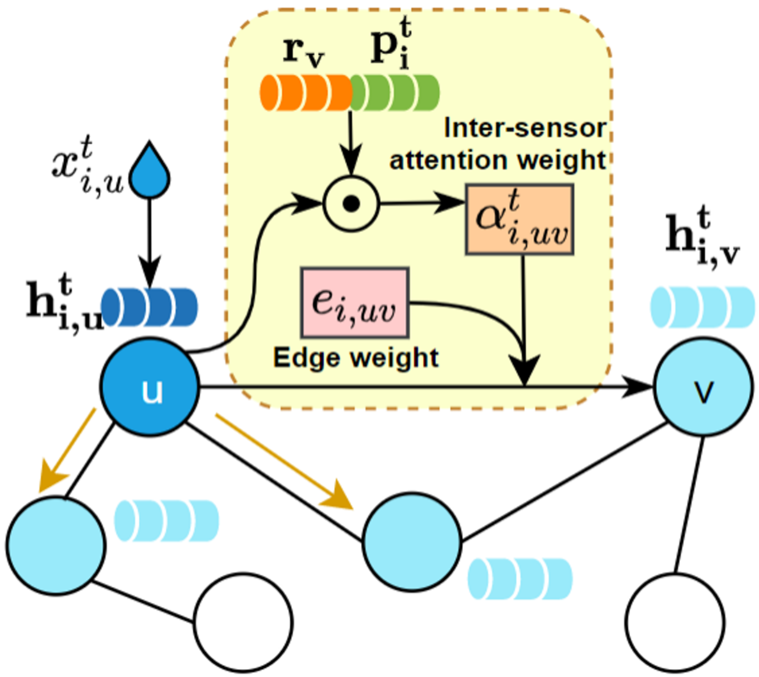
- **Active sensor:** Node u has been observed at time t

$$\boldsymbol{h}_{i,u}^t = \sigma(x_{i,u}^t \boldsymbol{R}_u)$$

- Sensor-specific weight vectors $\boldsymbol{R}_u$

Step 2: Embed observations

Generate observation embedding for neighbors of the active sensor through message passing



- Edge weight $e_{i,uv}$
- Inter-sensor attention weight $\alpha^t_{i,uv}$

$$\alpha^t_{i,uv} = \sigma(\boxed{\boldsymbol{h}^t_{i,u}}\boldsymbol{D}[\boldsymbol{r}_v||\boldsymbol{p}^t_i]^T)$$

- Embedding observation $x^t_{i,u}$

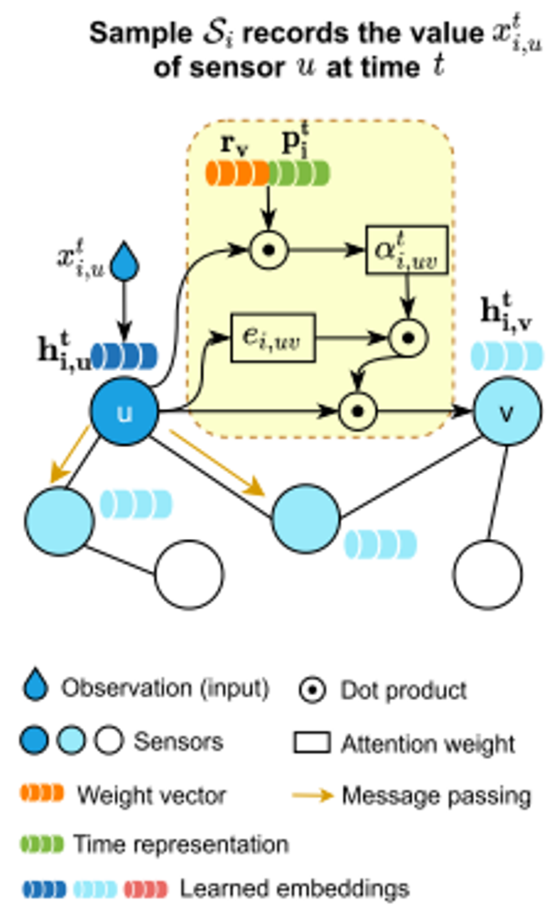
$$\boldsymbol{h}^t_{i,v} = \sigma(\boldsymbol{h}^t_{i,u}\boldsymbol{w}_u\boldsymbol{w}_v^T\boxed{\alpha^t_{i,uv}}e_{i,uv})$$

Step 2: Update sensor dependency graphs

- Average $\alpha^t_{i,uv}$ across timestamps t, update edge weights as:

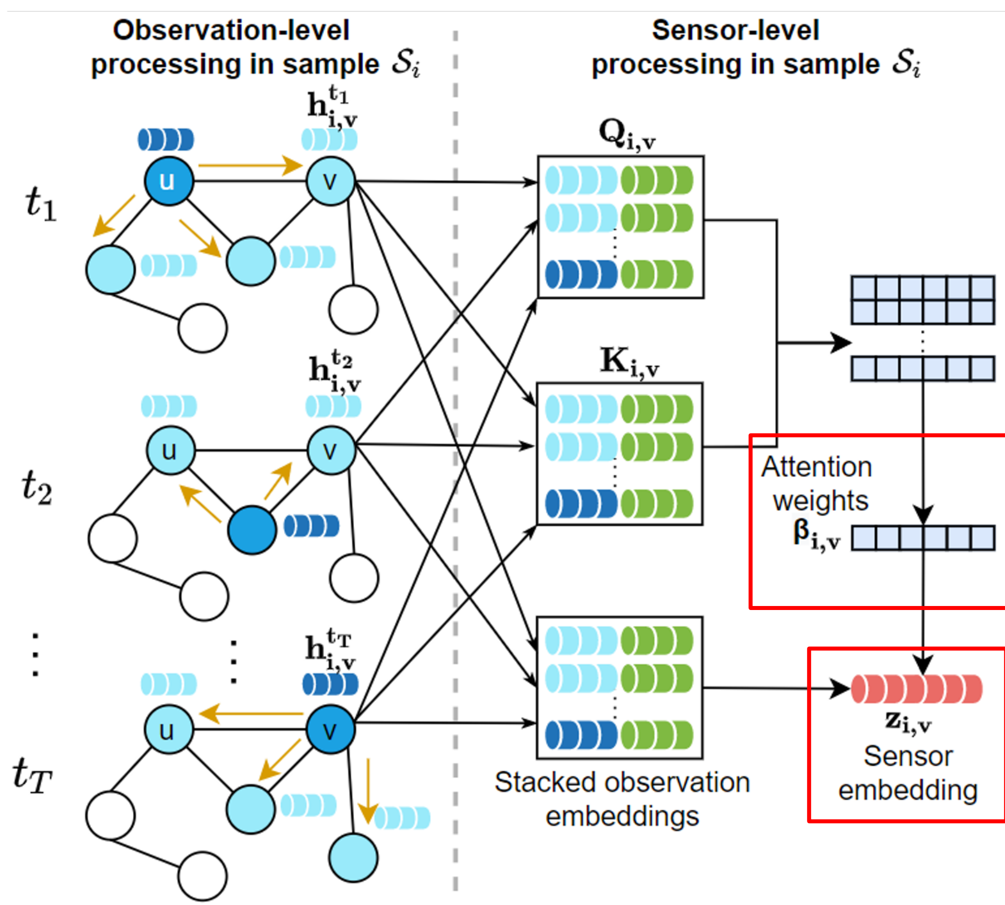
$$e^{(l)}_{i,uv} = \frac{e^{(l-1)}_{i,uv}}{|\mathcal{T}_{i,u}|} \sum_{t \in \mathcal{T}_{i,u}} \alpha^{(l),t}_{i,uv},$$

- Prune edges in sensor graphs
  - Remove bottom K% edges with smallest edge weights



Step 3: Embed sensors

For sensor v, aggregate observation embeddings across all timestamps into a single sensor embedding



Temporal self-attention

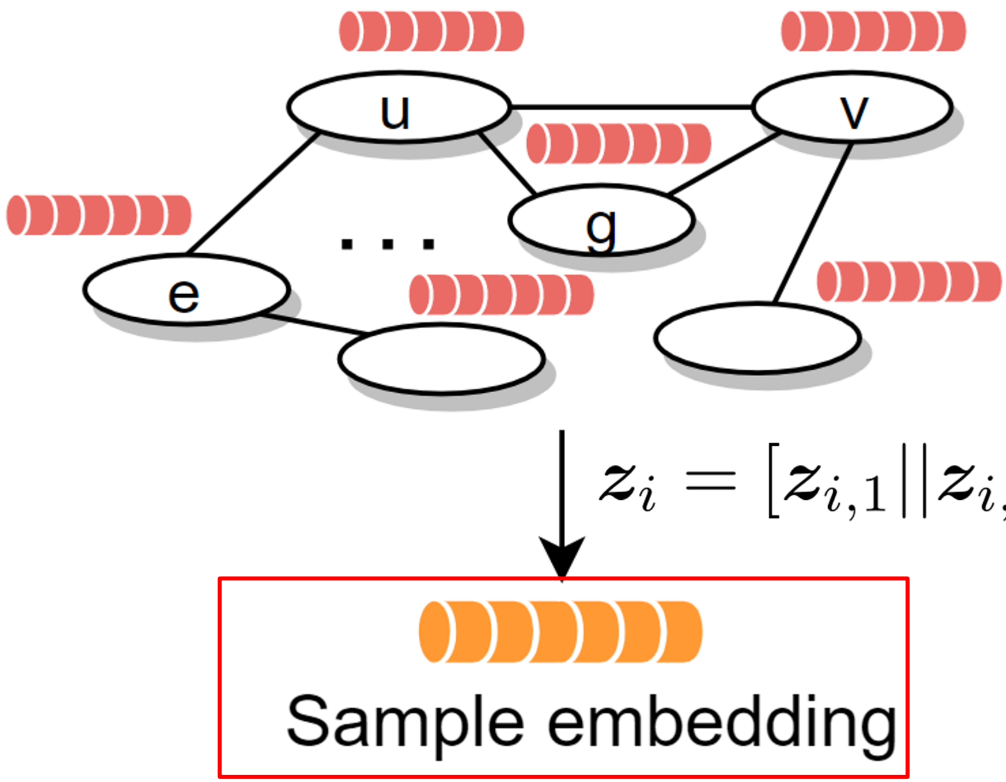
- Generate a single fixed-dimensional sensor embedding by temporal attention
- Apply to every sensor

$$\beta_{i,v} = \text{softmax}\left(\frac{\boldsymbol{Q}_{i,v}\boldsymbol{K}_{i,v}^T}{\sqrt{d_k}}\boldsymbol{s}\right)$$

Sensor embedding

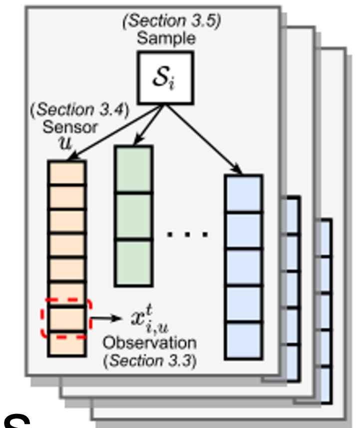
Step 4: Embed samples

Gather all sensor embeddings into a sample embedding using a readout function; Learned sample embedding support downstream tasks



$$z_i = [z_{i,1} || z_{i,2} || \cdots || z_{i,M}]$$

Recap: Raindrop model



Hierarchical learning of irregular time series:

- Step 1: Construct sensor dependency graphs
- Step 2: Generate embeddings of observations
- Step 3: Generate sensor embeddings
- Step 4: Generate sample embeddings

Applications

- Experiments:
  - Datasets
  - Baseline methods
  - Evaluation metrics

- Results:
  - Setting 1: Classic time series classification
  - Setting 2: Leave-random-sensors-out
  - Setting 3: Group-wise time series classification

Experimental setup (1/2)

- P19: PhysioNet Sepsis Early Prediction
  - 40,336 patients, 34 sensors
  - Classification: Sepsis occurring or not

- P12: PhysioNet Mortality Prediction
  - 11,988 patients, 36 sensors
  - Classification: Length of stay in the ICU (>3 days or not)

- PAM: PAMAP2 Physical Activity Monitoring
  - 5,333 samples, 17 sensors
  - 8-class classification: 8 activities of daily lives

Experimental setup (2/2)

- **Baselines:**
  - Transformer: replacing missing values with zeros
  - Transformer-mean: Transformer + Imputation
  - GRU-D: RNN-based model
  - SeFT: Set functions-based model
  - mTAND: Multi-time attention

- **Imbalanced datasets:**
  - P19, P12
  - P19: 96% negative samples
  - P12: 93% positive samples
  - AUROC, AUPRC

- **Balance datasets:**
  - PAM
  - Accuracy, Precision, Recall, F1 score

Setting 1/3: Time series classification

- Predict the label for a given time series sample
- Randomly split into training set (80%), validation set (10%), and testing set (10%)

Table 1: Method benchmarking on irregularly sampled time series classification task (Setting 1).

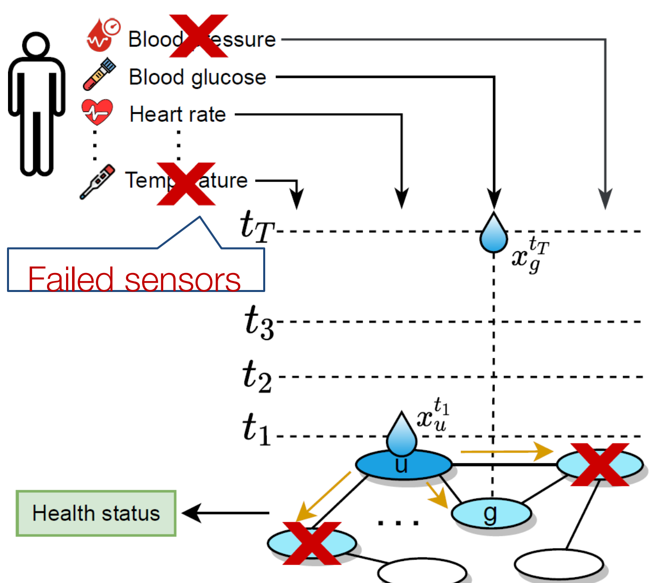
Models	P19		P12		PAM			
	AUROC	AUPRC	AUROC	AUPRC	Accuracy	Precision	Recall	F1 score
Transformer	83.2 ± 1.3	47.6 ± 3.8	65.1 ± 5.6	95.7 ± 1.6	83.5 ± 1.5	84.8 ± 1.5	86.0 ± 1.2	85.0 ± 1.3
Trans-mean	84.1 ± 1.7	47.4 ± 1.4	66.8 ± 4.2	95.9 ± 1.1	83.7 ± 2.3	84.9 ± 2.6	86.4 ± 2.1	85.1 ± 2.4
GRU-D	83.9 ±1.7	46.9 ± 2.1	67.2 ± 3.6	95.9 ± 2.1	83.3 ± 1.6	84.6 ± 1.2	85.2 ± 1.6	84.8 ± 1.2
SeFT	78.7 ± 2.4	31.1 ± 2.8	66.8 ± 0.8	96.2 ± 0.2	67.1 ± 2.2	70.0 ± 2.4	68.2 ± 1.5	68.5 ± 1.8
mTAND	80.4 ± 1.3	32.4 ± 1.8	65.3 ± 1.7	96.5 ± 1.2	74.6 ± 4.3	74.3 ± 4.0	79.5 ± 2.8	76.8 ± 3.4
RAINDROP	**87.0 ± 2.3**	**51.8 ± 5.5**	**72.1 ± 1.3**	**97.0 ± 0.4**	**88.5 ± 1.5**	**89.9 ± 1.5**	**89.9 ± 0.6**	**89.8 ± 1.0**

Setting 2/3: Leave-sensors-out

Dataset P19:

- 38,803 patients, 34 sensors
- Label: Sepsis or not
- Missing sensors: 10-50%

Results: Larger missing rate, larger margin over existing methods



Missing rate	Model	Accuracy	Precision	Recall	F1 score
10%	Transformer	60.3 ± 2.4	57.8 ± 9.3	59.8 ± 5.4	57.2 ± 8.0
	Trans-mean	60.4 ± 11.2	61.8 ± 14.9	60.2 ± 13.8	58.0 ± 15.2
	GRU-D	65.4 ± 1.7	72.6 ± 2.6	64.3 ± 5.3	63.6 ± 0.4
	SeFT	58.9 ± 2.3	62.5 ± 1.8	59.6 ± 2.6	59.6 ± 2.6
	mTAND	58.8 ± 2.7	59.5 ± 5.3	64.4 ± 2.9	61.8 ± 4.1
	Raindrop	**77.2 ± 2.1**	**82.3 ± 1.1**	**78.4 ± 1.9**	**75.2 ± 3.1**
20%	Transformer	63.1 ± 7.6	71.1 ± 7.1	62.2 ± 8.2	63.2 ± 8.7
	Trans-mean	61.2 ± 3.0	**74.2 ± 1.8**	63.5 ± 4.4	64.1 ± 4.1
	GRU-D	64.6 ± 1.8	73.3 ± 3.6	63.5 ± 4.6	64.8 ± 3.6
	SeFT	35.7 ± 0.5	42.1 ± 4.8	38.1 ± 1.3	35.0 ± 2.2
	mTAND	33.2 ± 5.0	36.9 ± 3.7	37.7 ± 3.7	37.3 ± 3.4
	Raindrop	**66.5 ± 4.0**	72.0 ± 3.9	**67.9 ± 5.8**	**65.1 ± 7.0**
30%	Transformer	31.6 ± 10.0	26.4 ± 9.7	24.0 ± 10.0	19.0 ± 12.8
	Trans-mean	42.5 ± 8.6	45.3 ± 9.6	37.0 ± 7.9	33.9 ± 8.2
	GRU-D	45.1 ± 2.9	51.7 ± 6.2	42.1 ± 6.6	47.2 ± 3.9
	SeFT	32.7 ± 2.3	27.9 ± 2.4	34.5 ± 3.0	28.0 ± 1.4
	mTAND	27.5 ± 4.5	31.2 ± 7.3	30.6 ± 4.0	30.8 ± 5.6
	Raindrop	**52.4 ± 2.8**	**60.9 ± 3.8**	**51.3 ± 7.1**	**48.4 ± 1.8**
40%	Transformer	23.0 ± 3.5	7.4 ± 6.0	14.5 ± 2.6	6.9 ± 2.6
	Trans-mean	25.7 ± 2.5	9.1 ± 2.3	18.5 ± 1.4	9.9 ± 1.1
	GRU-D	46.4 ± 2.5	**64.5 ± 6.8**	42.6 ± 7.4	44.3 ± 7.9
	SeFT	26.3 ± 0.9	29.9 ± 4.5	27.3 ± 1.6	22.3 ± 1.9
	mTAND	19.4 ± 4.5	15.1 ± 4.4	20.2 ± 3.8	17.0 ± 3.4
	Raindrop	**52.5 ± 3.7**	53.4 ± 5.6	**48.6 ± 1.9**	**44.7 ± 3.4**
50%	Transformer	21.4 ± 1.8	2.7 ± 0.2	12.5 ± 0.4	4.4 ± 0.3
	Trans-mean	21.3 ± 1.6	2.8 ± 0.4	12.5 ± 0.7	4.6 ± 0.2
	GRU-D	37.3 ± 2.7	29.6 ± 5.9	32.8 ± 4.6	26.6 ± 5.9
	SeFT	24.7 ± 1.7	15.9 ± 2.7	25.3 ± 2.6	18.2 ± 2.4
	mTAND	16.9 ± 3.1	12.6 ± 5.5	17.0 ± 1.6	13.9 ± 4.0
	Raindrop	**46.6 ± 2.6**	**44.5 ± 2.6**	**42.4 ± 3.9**	**38.0 ± 4.0**

Setting 3/3: Group-wise time series classification

- Split samples into two groups based on attributes
  - **Split by age:** Patients older than 65 years vs. younger patients
  - **Split by gender:** Male patients vs. Female patients
- Use one group as training set and randomly split the other group into validation (50%) and test set (50%)
  - Train on Young group → test on Old group
  - Train on Old group → test on Young group
  - Train on Male group → test on Female group
  - Train on Female group → test on Male group

Model	Generalizing to a new patient group							
	Train: Young → Test: Old		Train: Old → Test: Young		Train: Male → Test: Female		Train: Female → Test: Male	
	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC	AUROC	AUPRC
Transformer	76.2 ± 0.7	30.5 ± 4.8	76.5 ± 1.1	33.7 ± 5.7	77.8 ± 1.1	26.0 ± 6.2	75.2 ± 1.0	30.3 ± 5.5
Trans-mean	80.6 ± 1.4	39.8 ± 4.2	78.4 ± 1.1	35.8 ± 2.9	80.2 ± 1.7	32.1 ± 1.9	76.4 ± 0.8	32.5 ± 3.3
GRU-D	76.5 ± 1.7	29.5 ± 2.3	79.6 ± 1.7	35.2 ± 4.6	78.5 ± 1.6	31.9 ± 4.8	76.3 ± 2.5	31.1 ± 2.6
SeFT	77.5 ± 0.7	26.6 ± 1.2	78.9 ± 1.0	32.7 ± 2.7	78.6 ± 0.6	31.1 ± 1.2	76.9 ± 0.5	26.4 ± 1.1
mTAND	79.0 ± 0.8	28.8 ± 2.3	79.4 ± 0.6	29.8 ± 1.2	78.0 ± 0.9	26.5 ± 1.7	78.9 ± 1.2	29.2 ± 2.0
RAINDROP	**83.2 ± 1.6**	**43.6 ± 4.7**	**82.0 ± 4.4**	**44.3 ± 3.6**	**85.0 ± 1.4**	**45.2 ± 2.9**	**81.2 ± 3.8**	**40.7 ± 2.9**

Graph-Guided Networks For Ir

zitniklab.hms.harvard.edu/projects/Raindrop/

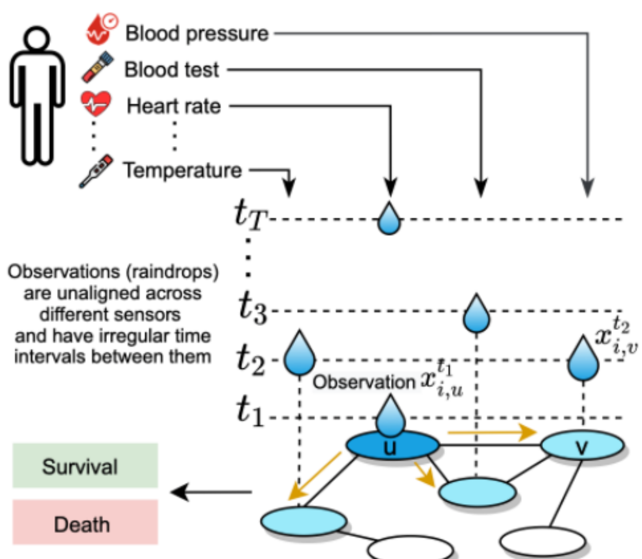
Incognito Update

Motivation for Raindrop

Raindrop takes samples as input, each *sample* containing multiple *sensors* and each sensor consisting of irregularly recorded *observations* (e.g., in clinical data, an individual patient's state of health, recorded at irregular time intervals with different subsets of sensors observed at different times). Each *observation* is a real-value scalar (sensor readout).

Raindrop is inspired by how raindrops hit a surface at varying time intervals and create ripple effects that propagate throughout the surface (as shown in the following figure). Mathematically, in Raindrop, observations (i.e., raindrops) hit the sensor graph (i.e., the surface) asynchronously and at irregular time intervals; each observation is processed by passing messages to neighboring sensors (i.e., creating ripples), taking into account the learned sensor dependencies.

The key idea of Raindrop is that the observed sensors can indicate how the unobserved sensors currently behave, which can further improve the representation learning of irregular multivariate time series. Taking advantage of the inter-sensor dependencies and temporal attention, Raindrop leans a fixed-dimensional embedding for irregularly sampled time series.



Take away messages:

- Irregular time series: Raindrop addresses the complexity of time series, e.g., misaligned observations, varying time gaps & varying numbers of observations per sensor
- Inter-sensor structure: Raindrop adopts neural message passing to model inter-sensor dependencies in irregular time series
- Great generalization: Raindrop has excellent performance in challenging settings, including setups where a subset of sensors have malfunctioned (i.e., no readouts at all)

Thank you!

Joint work with X. Zhang, M. Zeman, and T. Tsiligkaridis

https://github.com/mims-harvard/Raindrop