Graph Signal Sampling via Reinforcement Learning

Aalto University School of Science

Oleksii Abramenko and Alexander Jung

Aalto University, Finland

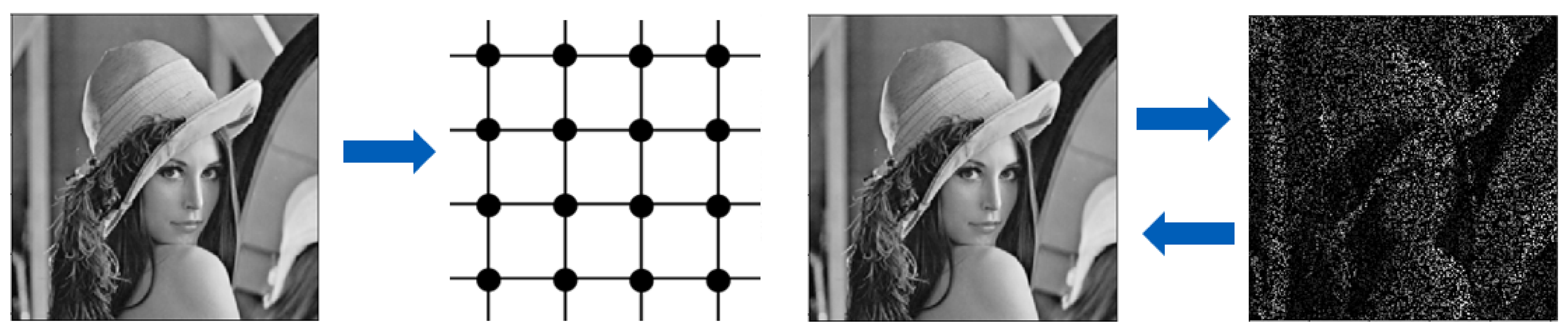
Motivation

Two core problems within graph signal processing:

- How to sample/select the most representative (few) nodes of the graph?
- How to recover original representation based on these few samples?

One motivating application can be image compression:

- We can represent image as a grid graph with pixels being graph nodes:
- We can recover image by observing only small portion of graph nodes (pixels):



Contribution

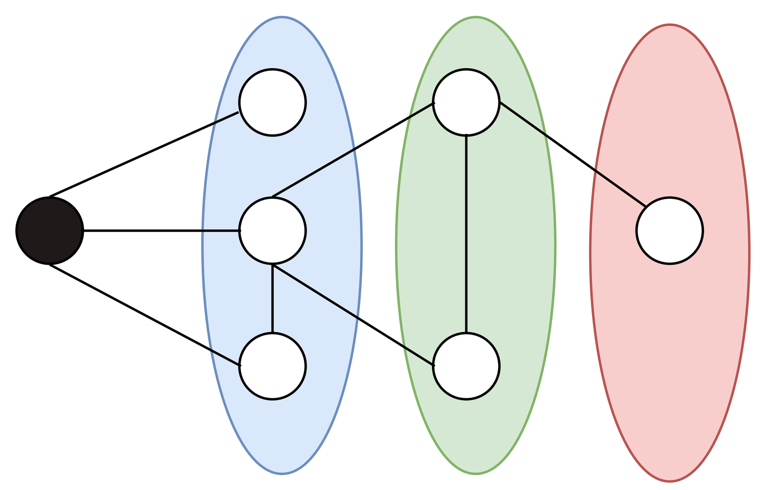
- we formulate graph sampling problem as Multi-Armed Bandit (MAB) problem
- the sample selection is performed by the "sampling agent" which crawls over the graph and decides whether or not a particular graph node should be sampled

Problem Formulation

- given:
  - data graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$
  - data graph specified by the adjacency matrix $\mathbf{W} \in \mathbb{R}_+^{N \times N}$
  - each graph node is associated with label $x[i]$
  - we are provided with the desired sampling set size which is small compared to the size of the graph: $|\mathcal{M}| \ll |\mathcal{V}|$
- goal:
  - For a fixed sampling set size (sampling budget) $|\mathcal{M}|$ we want to choose the sampling set such that the signal samples $\{x[i]\}_{i \in \mathcal{M}}$ carry maximal information about the entire graph signal.

Sampling as Multi-Armed Bandit Problem

- Given action space with H actions $\mathcal{A} = \{1, ..a, ..H\}$.
- Graph is decomposed into a-step neighbourhoods based on the shortest path from the current location of sampling agent



- filled node – current location of sampling agent
- blue, green and red regions represent 1-,2-,3-step neighbourhoods
- each a-step neighbourhood is associated with the a-th arm of the hypothetical MAB machine
- At a given time step t, the sampling agent chooses an action $a \in \mathcal{A}$ which refers to the number of hops the sampling agent performs starting at the current node i_t to reach a new node i_{t+1}, which will be added to the sampling set, i.e., $\mathcal{M} := \mathcal{M} \cup \{i_{t+1}\}$.

How the optimal policy is learned?

- Policy is represented as a probability distribution over arms of the MAB and parametrized by the trainable vector $\mathbf{w} = (w_1, \dots, w_H)^T$:

$$\pi^{(\mathbf{w})}(a) = \frac{e^{w_a}}{\sum_{b \in \mathcal{A}} e^{w_b}}$$

- Reward is negative Mean Squared Error (MSE) of graph signal recovery:

$$MSE := (1/N) \sum_{j \in \mathcal{V}} (x[j] - \hat{x}^{\mathcal{M}}[j])^2$$

$$R := -MSE$$

- Weights are learned via standard gradient ascent:

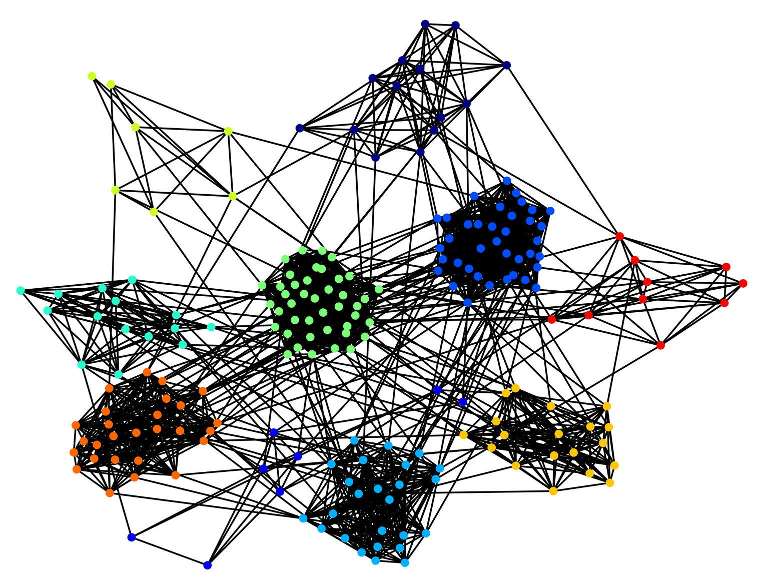
$$w_a := \begin{cases} w_a + \alpha R(1 - \pi(a)), a = a_k \\ w_a - \alpha R \pi(a), \forall a \neq a_k \end{cases}$$

for $k = 1..M-1, a \in \mathcal{A}$

- Training procedure aims at maximizing reward and, respectively, minimizing MSE of graph signal recovery

Simulation Setup

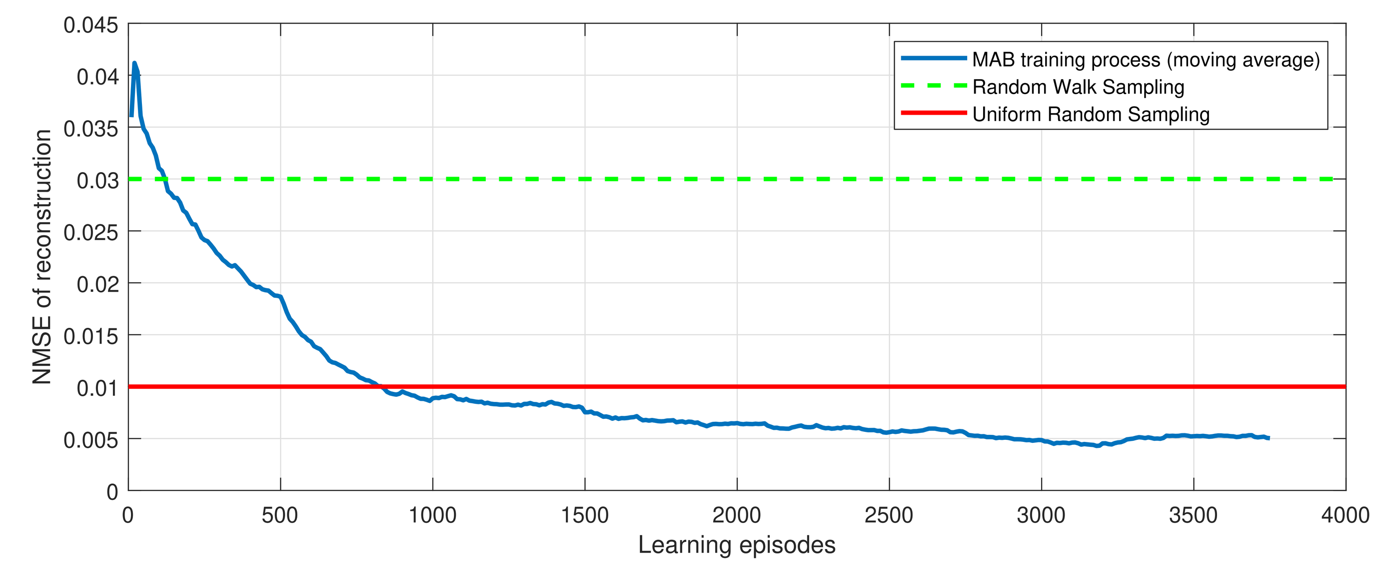
- Stochastic Block Model family with 10 clusters



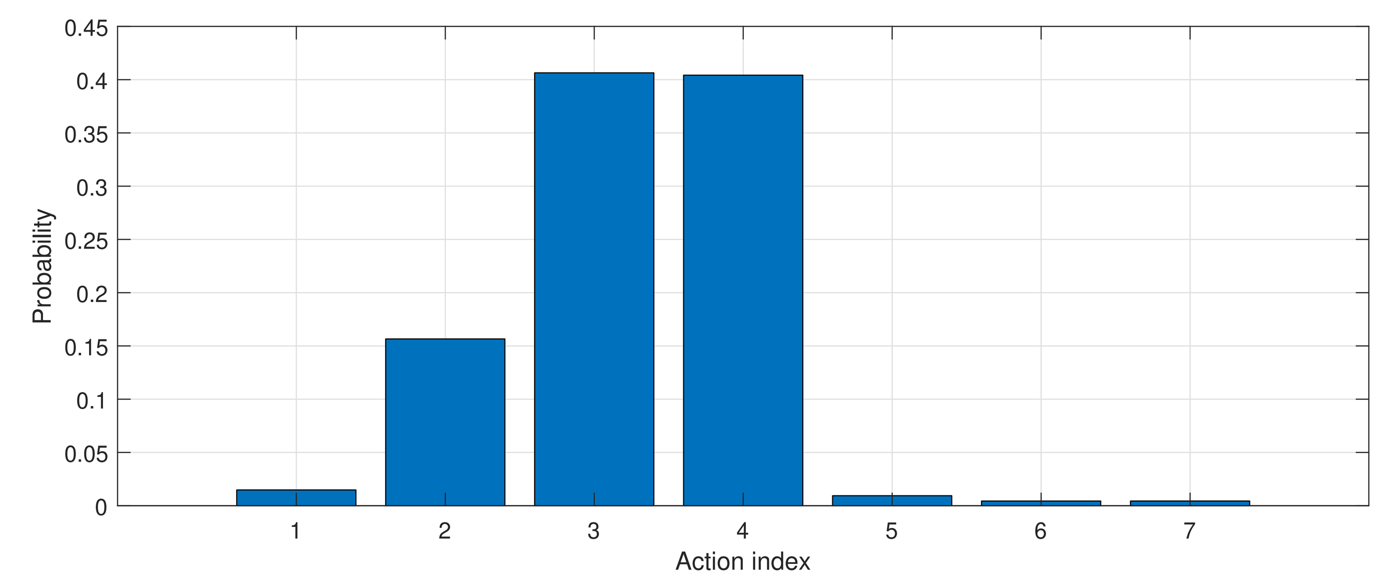
- cluster sizes have geometric distribution with probability of success 0.08
- signal values in i-th cluster are all equal to i
- intra-cluster connection probability $p = 0.7$, inter-cluster connection probability $q = 0.01$

Simulation Results

- Convergence process for one training instance:



- Average policy $\pi^{(\mathbf{w})}$ over 500 training instances:



- Dependency signal recovery error vs sampling budget: