Graph Signal Sampling via Reinforcement Learning

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 $G = (V,E)$
  - data graph specified by the adjacency matrix $W \in \mathbb{R}^{N \times N}$
  - 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: $|M| \ll |V|$
- Goal:
  - For a fixed sampling set size (sampling budget) $|M|$ we want to choose the sampling set such that the signal samples $\{x[i]\}_{i \in M}$ carry maximal information about the entire graph signal.

Sampling as Multi-Armed Bandit Problem
- Given action space with $H$ actions $A = \{1, \ldots, H\}$.
- Graph is decomposed into $a$-step neighbourhoids 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 neighbourhoids
  - each $a$-step neighbourhoid is associated with the $a$-th arm of the hypothetical MAB machine
- At a given time step $t$, the sampling agent chooses an action $a \in A$ 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., $M := M \cup \{i_{t+1}\}$

How the optimal policy is learned?
- Policy is represented as a probability distribution over arms of the MAB and parametrized by the trainable vector $w = (w_1, \ldots, w_H)^T$:
  $$\pi^{(w)}(a) = \sum_{b \in A} w_a$$
- Reward is negative Mean Squared Error (MSE) of graph signal recovery:
  $$MSE := (1/N) \sum_{j \in V} [x[j] - \hat{x}^{M}[j]]^2$$
  $$R := -MSE$$
- 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 $k = 1, \ldots, M - 1, a \in A$
- 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 $\pi^{(w)}$ over 500 training instances:
- Dependency signal recovery error vs sampling budget: