AI607: GRAPH MINING AND SOCIAL NETWORK ANALYSIS (FALL 2021)

Homework 4: Loopy Belief Propagation

Release: Nov 12, 2021,
Due: Nov 26, 2021, 11:59pm

In this assignment, we will infer node labels using the Loopy Belief Propagation algorithm. We will compare the results with the ground-truth labels.

1 Loopy Belief Propagation

Loopy Belief Propagation (LBP) is an inference algorithm that approximately calculates the marginal distribution of unobserved variables in a probabilistic graphical model. Among many applications, we will implement LBP in a pairwise Markov Random Field (MRF). A pairwise MRF is an **undirected graphical model**. In the pairwise MRF, nodes represent discrete random variables and edges represent the relationships between two variables.

We need two cencepts, node potentials ϕ and edge potentials ψ to understand the equations of LBP. A node potential $\phi_i(x_i)$ of a node i is roughly the prior knowledge about the probability of the node i being in the state x_i. An edge potential $\psi_{ij}(x_i, x_j)$ is roughly the prior knowledge about the conditional probability that the node j is in the state x_j given that the node i is in the state x_i.

At each iteration, all nodes send messages to their neighboring nodes. The algorithm assumes that a message m_{ij} successfully captures the influence of the node i on its neighboring node j's marginal probability. The equation for the update of the messages $m_{ij}(x_j)$ is as follows.

$$m_{ij}^{(t+1)}(x_j) \leftarrow \sum_{x_i \in L} \phi_i(x_i)\psi_{ij}(x_i, x_j) \frac{\prod_{l \in N(i)} m_{li}^{(t)}(x_i)}{m_{ji}^{(t)}(x_i)}, \tag{1}$$

where L is the set of labels and $N(i)$ is the neighbors of the node i.

Then, you should normalize each $m_{ij}^{(t+1)}$ so that $\sum_{s=1}^{S} m_{ij}^{(t+1)}(s)$ can be a constant. The belief b_i of the node i is computed based on the converged messages as follows:

$$b_i^{(t+1)}(x_i) = c_i \phi_i(x_i) \prod_{l \in N(i)} m_{li}^{(t+1)}(x_i) \tag{2}$$

where c_i is a normalization constant for making the beliefs of the node i being in different states x_i sum up to 1.

2 Implementation [30 points]

In this section, you will implement LBP. You are **NOT** allowed to use any other external libraries (Numpy, Scipy, Snap.py, NetworkX, etc.). In this assignment, we assume that all node indices are between 0 to $N-1$, where N is the number of nodes in the input graph.

Dataset	**Nodes**	**Edges**
PubMed	19,717	44,327
PolBlogs	1,224	16,718

Table 1: Summary of datasets

State	0	1
0	**0.501**	0.499
1	0.499	**0.501**

Table 2: ψ for PolBlogs

State	0	1	2
0	**0.334**	0.333	0.333
1	0.333	**0.334**	0.333
2	0.333	0.333	**0.334**

Table 3: ψ for PubMed

2.1 Loading a graph and labels [15 points]

We will test your code on two datasets, PubMed and PolBlogs. PolBlogs is a graph of hyperlinks where each node represents a web blog on US politics with two possible states. Each state indicates political leanings. Specifically, nodes with label 0 are left or liberal, and those with label 1 are right or conservative. PubMed is a citation graph where each node is a scientific publication from the PubMed database. The label of each node is the type of diabetes that the publication is pertaining to. Specifically, the nodes with labels 0, 1, and 2 are pertaining to “Diabetes Mellitus Experimental”, “Diabetes Mellitus Type 1”, and “Diabetes Mellitus Type 2”, respectively.

Edges and labels are saved in files ended with ‘`XXX_edges.txt`’ and ‘`XXX_labels.txt`’, respctively.[1] Each edge is written in a single line of an edge file in the form of ‘<SOURCE NODE ID> <DESTINATION NODE ID>’. The label of each node is written in a single line of an label file by in the form of ‘<NODE ID> <STATE ID>’. There are test nodes whose labels are not provided in `XXX_labels.txt`. The format of the line corresponding to a test node is ‘<NODE ID>’. You should fill in the __init__ function of the Graph class in graph.py.

```
class Graph:
    def __init__(self, edges_path: str, states_path: str) -> None:
        ############## TODO: Fill out the class variables #################
        #  self._nodes : Set of nodes                                      #
        #  self._neighbors : key: node, value: set of neighbors of node    #
        #  self._node_labels: key: node, value: label of node              #
        # WARNING: Do not declare another class variables                  #
        ####################################################################

        self._nodes: Set[int] = set()
        self._neighbors: Dict[int, Set[int]] = dict()
        self._node_labels: Dict[int, int] = dict()
```

[1]XXX: name of graph (e.g., polblogs, pubmed).

```
        ######################## Implementation end ######################
```

2.2 Loopy belief propagation [15 points]

The edge potential values are given in Tables 2 and 3. The node potentials should be set to 0.9 for the correct state, and remaining 0.1 should be divided equally to the other states. The node potentials of unlabeled nodes (i.e., test nodes) are $\frac{1}{\#states}$ for each state. You should fill in the loopy_belief_propagation function in `loopy_belief_propagation.py`. The algorithm should stop after 200 iterations, and it should stop earlier if the convergence criterion is met. The convergence criterion is satisfied if any only if the L1-distance (also known as, Manhattan distance) between the previous and current belief vectors for every node is less than 1e-4.

```
def loopy_belief_propagation(
    graph: Graph,
    max_iters: int,
    tol: float,
) -> Dict[int, Dict[int, float]]:
    belief: Dict[int, Dict[int, float]] = dict()
    ####### TODO: Implement the loopy belief propagation algorithm #########
    #  belief : key: node, value: dictionary in which the key is state and #
    #  the value is belief of state                                       #
    ########################################################################

    for itr in range(maxiters):

        #### Check the convergence ###
        # Stop the iteration if L1norm[belief(t) - belief(t-1)] < tol
        delta: float = 0.0
        print(f"[Iter {itr}]\tDelta = {delta}")

        if delta < tol:
            break
    ######################## Implementation end ########################

    return belief
```

3 Evaluation [70 points]

We will evaluate your implementation by comparing the states of test nodes and the beliefs provided by the implementation regarding the states of the test nodes. You should submit the result file (`XXX_belief.txt`). Each line of the result file should be in the following format:

```
<NODE ID> <STATE ID>:<STATE BELIEF> <STATE ID>:<STATE BELIEF> ...
```

Each line should contain the NODE ID and the belief value (STATE BELIEF) for each state (STATE ID). STATE ID and STATE BELIEF should be separated by a colon; and NODE ID and each <STATE ID>:<STATE BELIEF> pair should be separated by a space.

4 Notes

- Your implementation should run on TA's desktop within **10 minutes** for a single dataset.
- You may encounter some subtleties when it comes to implementation. Please come up with your own design and/or contact Taehyung Kwon (taehyung.kwon@kaist.ac.kr) and Hyeonsoo Jo (hsjo@kaist.ac.kr) for discussion. Any ideas can be taken into consideration when grading if they are written in the *readme* file.
- In `graph.py`, `main.py` and `loopy_belief_propagation.py`, do not modify the code outside of the "TODO" regions. For example, do not import additional python libraries outside "TODO" regions.

5 How to submit your assignment

1. Create hw4-[your student id].tar.gz, which should contain the following files:
    - **graph.py, loopy_belief_propagation.py, main.py**: these files should contain your implmentation.
    - **result.tar.gz**: this file should contain all the `XXX_belief.txt` files, which contain the beliefs of all nodes.
    - **readme.txt**: this file should contain the names of any individuals from whom you received help, and the nature of the help that you received. That includes help from friends, classmates, lab TAs, course staff members, etc. In this file, you are also welcome to write any comments that can help us grade your assignment better, your evaluation of this assignment, and your ideas.
2. All **3 files** should be included in a **single folder**.
3. Make sure that no other files are included in the tar.gz file.
4. Submit the tar.gz file at KLMS (`http://klms.kaist.ac.kr`).