Assignment 1

CS 512: Data Mining Principles (Spring 2021)
Instructor: Hanghang Tong

Release date: Feb. 1st, 2021
Due date: Mar. 5th, 2021

- This assignment will cover the content from slides #1 (Introduction), #2 (Frequent Pattern Mining), and #3 (Classification).
- Feel free to talk to other members in the class when doing the homework. We care more about whether you learn how to solve the problem entirely on your own than you demonstrate that you solved it. You should, however, write down your solution yourself. Please try to keep the solution brief and clear.
- Please use Piazza first if you have questions about the homework. Also feel free to send us e-mails and come to office hours.
- The homework is due at 11:59 PM on the due date. We will be using Compass (`http://compass2g.illinois.edu`) for collecting assignments. **Please do not hand in a scan of your handwritten solution, only the typed solution (e.g., Microsoft Word, Latex, etc) will be graded**. The datasets for HW1 is in **HW1_dataset.zip** on Compass. Contact the TAs if you are having technical difficulties in submitting the assignment. We do **NOT** accept late homework!
- The solution report should be submitted as a **single** pdf file using the name convention `yourNetid_HW1.pdf`. If you use additional source code (Python/Java/C++ are all acceptable) for solving problems, you are required to submit them and use the file names to identify the corresponding questions. For instance, '`yourNetid_HW1_problem1.py`' refers to the python source code for Problem 1 for HW 1. Compress all the files (pdf and source code files) into one zip file. Submit the compressed file ONLY.
- For each question, you will NOT get full credits if you only give out a final result. Necessary calculation steps are required. If the result is not an integer, round your result to 2 decimal places.

Problem 1. Short Answers. (8 points)

(a) (2 points) List two differences between Frequent Pattern Growth algorithm and Apriori algorithm (e.g., pattern generation, candidate generation, processing time).

(b) (2 points) The Apriori algorithm make use of prior knowledge of subset support properties. Prove (1) all nonempty subsets of a frequent itemset also must be frequent; and (2) the support of any nonempty subset s' of itemset s must be at least as large as the support of s.

(c) (2 points) Regarding measures of interestingness, what problem might *Lift* and χ^2 have? To address this problem, null-invariant measures are used, explain why Jaccard function is null-invariant? (hint: Venn diagram)

(d) (2 points) Overall, in sequential pattern mining, PrefixSpan has better performance compared to GSP and SPADE. Under what circumstance, PrefixSpan will also have poor performance?

Problem 2. Apriori vs Frequent Pattern Growth. (15 points)

TID	Items
1	$\{A, B, C, H, P\}$
2	$\{A, C, M, B, L\}$
3	$\{B, F, O, L\}$
4	$\{A, C, D, F, G, M, P\}$
5	$\{A, C, F, P, M\}$
6	$\{C, F, H, D\}$

Table 1: Transaction database.

A transactional database is given in Table 1.

(a) (6 points) Let $min\_sup = 3$ and apply Apriori algorithm on the dataset. Present the intermediate results of the first **three** scans and the final derived frequent itemsets.

(b) (6 points) Let $min\_sup = 3$ and sort the frequent items first in frequency descending and then alphabetical order. Construct the Frequent Pattern-tree from the dataset in the same format as in the lecture.

(c) (3 points) List the top-3 association rules in terms of confidence from 2-itemsets.

Problem 3. Frequent Pattern Mining Implementation (12 points)
Implement the core algorithm of Apriori (direct calling Apriori from external packages is NOT allowed) on the grocery market transaction data (i.e., **groceries.csv**), and finish the following tasks.

(a) (2 points) Report data statistics, for example, #product types, average #products per transaction, 5 most popular products, etc.

(b) (5 points) Manually define the threshold settings, e.g., $min\_support, min\_confidence$, record the settings and list the top-10 corresponding frequent itemsets if existed. Choose any pair of items (e.g., "yogurt" and "coffee"), construct the contingency table and evaluate the interestingness using *Lift*. Report the table and evaluation results.

(c) (5 points) Report the running time of Apriori w.r.t. different data sizes (e.g., let the number of transactions be $1,000$, $2,000$, etc.). Compare the efficiency performance of Apriori and FP-Growth (You can directly call third-party libraries to run FP-Growth).

Problem 4. Sequential and Graph Pattern Mining (10 points)

(a) A sequential database is presented in Table 2.

(1) (3 points) Given a set of prefixes, $\langle ab\rangle, \langle (ab)\rangle, \langle de\rangle$, $\langle abc\rangle, \langle acb\rangle, \langle acba\rangle$, list the corresponding projection(s) according to the definition of prefix and projection;

(2) (2 points) List the sequential pattern(s) where $support = 3$ and length ≥ 3;

SID	Sequence
1	$\langle a(bcf)(ac)b\rangle$
2	$\langle (ab)c(bc)ab(ce)\rangle$
3	$\langle de(af)acb(acf)acd\rangle$
4	$\langle (ad)de(abc)(ae)\rangle$
5	$\langle (aef)ac(abd)df(ab)\rangle$
6	$\langle (abc)eab(cd)f(abd)\rangle$

Table 2: Sequence Database.

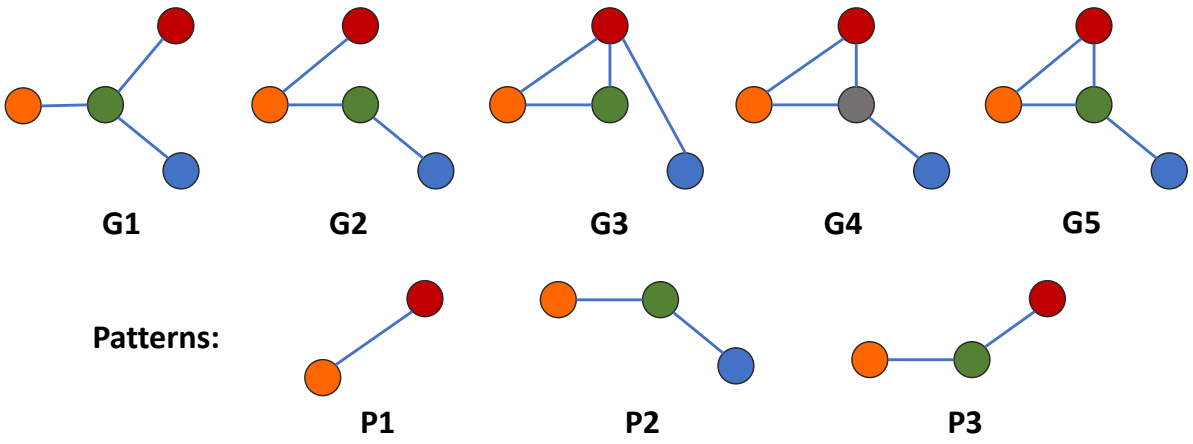

Figure 1: Input graphs and subgraph patterns.

(b) Given the input graphs (i.e., first row) and patterns (i.e., second row) in Figure 1.

(1) (3 points) Compute the support (in fraction) of the subgraph patterns and list the corresponding input graphs in which the patterns appear.

(2) (2 points) Let $min\_sup = 3$, show how Apriori-based method works to find all frequent graph patterns using either edge growing or vertex growing (present intermediate results/patterns at each iteration).

Problem 5. SVM, RBF Kernel (15 points)
Given n training examples $(\mathbf{x}_i, y_i)(i = 1, 2, \ldots, n)$ where $\mathbf{x}_i$ is the feature vector of the i-th training example and y_i is its label, we train an support vector machine (SVM) with Radial Basis Function (RBF) kernel on the training data. Note that the RBF kernel is defined as $K_{RBF}(\mathbf{x}, \mathbf{y}) = \exp\left(-\gamma||\mathbf{x} - \mathbf{y}||_2^2\right)$.

(a) (5 points) Let $\mathbf{G}$ be the $n \times n$ kernel matrix of RBF kernel, i.e., $\mathbf{G}[i, j] = K_{RBF}(\mathbf{x}_i, \mathbf{x}_j)$. Prove that all eigenvalues of $\mathbf{G}$ are non-negative.

(b) (5 points) Prove that RBF kernel is the sum of infinite number of polynomial kernels.

(c) (3 points) Suppose the distribution of training examples is shown in Figure 2 where '+' denotes positive example and '-' denotes negative example. If we set γ large enough (say 1000 or larger), what could possibly be the decision boundary of the SVM after training? Please draw it on Figure 2.

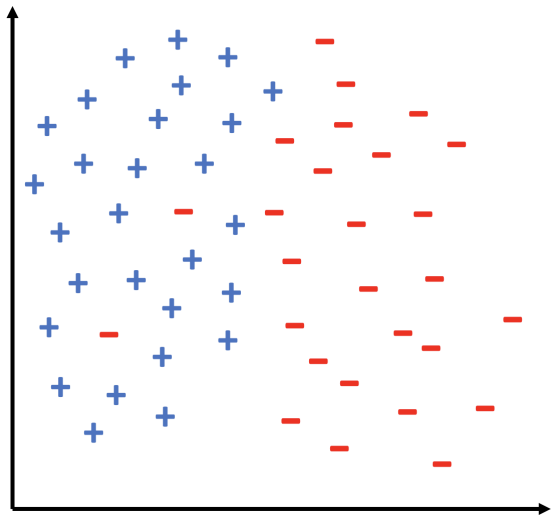

Figure 2: Distribution of Training Examples.

(d) (2 points) if we set γ to be infinitely large, what could possibly happen when training this SVM?

Problem 6. Feature Selection (15 points)
Given the **risk auditing** dataset where every row refers to a data sample, the last column denotes the labels of data samples, and the remaining columns serve as the input features, try to:

(a) (5 points) implement an **SVM** classifier using the **train.csv**, **validation.csv**, and **test.csv**. Report the classification results over the test set. (3rd-party packages are **allowed**)

(b) (10 points) improve the performance of SVM classifier by implementing **Fisher Score** feature selection method. (3rd-party packages are **NOT allowed**)

Remarks that you should strictly follow the conventional machine learning paradigm to train your model on the training set, select the best hyper-parameter on the validation set and report the corresponding results on the test set. The source code you write should be submitted with the .pdf report in the compressed .zip file.

Problem 7. Self-Training (15 points)

Follow the improvement made from **Problem 6** via feature selection, try to further improve the classification results by self-training using the unlabelled data **unlabelled.csv**. Remarks that all the columns in unlabelled.csv should serve as input features. You should strictly follow the conventional machine learning paradigm to train your model on the training set, select the best hyper-parameter on the validation set and report the corresponding results on the test set. The source code you write should be submitted with the .pdf report in the compressed .zip file. [**HINT:** change the output of SVM into probability output and define the 'confidence' towards every unlabelled sample] (for the implementation of SVM, 3rd-party packages are **allowed** but for the implementation of self-training framework, 3rd-party packages are **NOT allowed**)

Problem 8. Random Walk with Restart (10 points)

(a) (3 points) From its formula, try to explain why random walk with restart (RWR) can capture multiple weighted relationships between nodes?

(b) (7 points) Implement random walk with restart (RWR) on the **email-EU-core** unweighted undirected graph. The given file is an edge list and since it is an unweighted undirected graph, when see 'i j' in the edge list, set both $\mathbf{A}[i, j]$ and $\mathbf{A}[j, i]$ as 1. Set preference vector $\mathbf{e}$ as a uniform vector to obtain global ranking and set the damping factor c as 0.9. Report the indices of top-10 nodes and the source code you write should be submitted with the .pdf report in the compressed .zip file. [**HINT:** normalize adjacency matrix before RWR: $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$, where $\mathbf{D}$ is an diagonal matrix s.t. $\mathbf{D}[i, i] = \sum_j \mathbf{A}[i, j]$] (3rd-party packages are **NOT allowed**)