Machine Learning on Graphs: Foundations and Applications

EXERCISE 1

Group 29

Rishang Yadav 485041 | Risa Niranjan Chaudhari 485510 | Bhoomi Dhawan 485556

---

1. **Problem 1:** Given two directed trees, can we decide in polynomial time if the two trees are isomorphic? Can we also extend the algorithm to undirected trees? (5 Pt.)

Ans: Yes, we can decide if two directed or undirected trees are isomorphic in polynomial time. We can use the **1-WL algorithm** for the isomorphism test. Although we know that the algorithm fails on certain graphs, it is a complete test for trees due to their simple acyclic structures.

1. Polynomial Time Justification

- We know that the 1-WL algorithm can be computed in $\mathcal{O}(|E(G)| \cdot T)$ time, where T is the number of iterations.
- For any tree on n vertices, the number of edges is $|E| = n-1$, so $|E| \in \mathcal{O}(n)$.
- The algorithm is guaranteed to converge to a stable partition in at most n iterations, so $T \leq n$.
- Thus, the total runtime is bounded by $\mathcal{O}(n \cdot |E|) = \mathcal{O}(n \cdot n) = \mathcal{O}(n^2)$, which is polynomial.

2. Correctness for Trees

The 1-WL algorithm iteratively refines a colour $C_t^1(v)$ for each vertex v based on its previous colour and the multiset of its neighbours' previous colours:

$$C_t^1(v) := \text{Recolor}\left(\left(C_{t-1}^1(v), \{C_{t-1}^1(w) \mid w \in N(v)\}\right)\right)$$

- **Failure on Cyclic Graphs:** We know that 1-WL fails to distinguish a 6-cycle from two 3-cycles. This is because for both graphs, the local neighbourhood of every node looks identical, leading to the same colour vector.
- **Success on Acyclic Graphs (Trees):** Trees do not have this "local ambiguity". The colouring process propagates "inward" from the leaves

(nodes of degree 1). The colour $C^1_k(v)$ at iteration k becomes a unique, injective hash of the isomorphism type of the subtree of depth k rooted at v. Because there are no cycles, this "hash" is guaranteed to be unique for non-isomorphic structures.

- **Conclusion:** When the algorithm stabilizes after L iterations, the final vector of colors, $\phi^L_{\mathrm{WL}}(G)$, is a unique mapping for the tree's isomorphism type. Thus, if $\phi^L_{\mathrm{WL}}(G) = \phi^L_{\mathrm{WL}}(H)$, we can conclude that $G \simeq H$.

3. Extension to Directed and Undirected Trees

- **Undirected Trees:** The 1-WL algorithm as defined in the slides is already designed for undirected graphs. The proof of correctness above, which relies on the acyclic property, applies directly.
- **Directed Trees:** The algorithm is easily extended. We simply redefine the "neighborhood" $N(v)$ in the update rule. For instance, $N(v)$ can be a pair of multisets: one for in-neighbors and one for out-neighbors. The core logic of propagating structural information in an acyclic graph remains the same, and the polynomial-time guarantee holds.

2. **Problem 2:** Let $G = (V, E)$ be an undirected graph with $|V| = n$. Show that the 1-WL (color refinement) procedure always reaches a stable coloring after finitely many iterations. That is, prove there exists $t^* \geq 0$ such that for all $v, w \in V$

$$C^1_{t^*}(v) = C^1_{t^*}(w) \iff C^1_{t^*+1}(v) = C^1_{t^*+1}(w).$$

Give an upper bound on t^* in terms of n. (2 Pt.)

Ans: Let $G = (V, E)$ be an undirected graph with $|V| = n$. Denote by $C_t : V \to \mathcal{C}_t$ the coloring after t iterations of the 1-WL (color refinement) algorithm, where $\mathcal{C}_t$ is the set of colors present at iteration t. Each C_t divides the vertex set V into color classes. The update rule defines C_{t+1} by assigning each vertex $v \in V$ a new color based on the pair

$$\big(C_t(v),\ \text{multiset}\{\, C_t(u) : u \in N(v) \,\}\big),$$

where $N(v)$ denotes the neighbors of v.

Monotonicity of the refinement. At every step, the new coloring C_{t+1} refines the previous one C_t. This means that each color class in C_{t+1} is contained within some color class from C_t. Once two vertices receive different colors, they never merge again in later iterations. If two vertices are distinguished in step $t + 1$, they must have belonged to the same class in step t.

When the coloring changes, new colors appear. If C_{t+1} differs from C_t, at least one color class of C_t must have been split into smaller parts in C_{t+1}. Consequently,

$$|\mathcal{C}_{t+1}| > |\mathcal{C}_t|.$$

Hence, each time the refinement changes the coloring, the total number of color classes strictly increases.

Bounding the number of iterations. Since there are only n vertices, the number of distinct color classes can never exceed n. Because the sequence $|\mathcal{C}_0|, |\mathcal{C}_1|, \ldots$ increases strictly whenever the coloring changes, the process can make at most $n-1$ such changes. Therefore, the refinement must stabilize after at most $n-1$ iterations. Let t^* be the smallest integer such that

$$C_{t^*} = C_{t^*+1}.$$

At this point, the coloring no longer changes, and thus

$$t^* \leq n - 1.$$

Stability condition. By definition of t^*, for every pair of vertices $v, w \in V$, we have

$$C_{t^*}(v) = C_{t^*}(w) \iff C_{t^*+1}(v) = C_{t^*+1}(w).$$

This means the coloring has reached a stable state—further refinement steps will not produce any new distinctions.

3. **Problem 3:** Given an undirected graph G with adjacency matrix $A(G)$, show that the number of triangles in the graph G is equal to $\frac{1}{6}\,\text{trace}\left(A(G)^3\right)$. Here, the trace of a matrix is the sum of elements on the main diagonal. (2 Pt.)

Ans: Recall that for any integer $t \geq 1$, the (i,j)-entry of $A(G)^t$ equals the number of walks of length t from vertex i to vertex j.

In particular, the diagonal entry $(A^3)_{ii}$ counts the number of closed walks of length 3 that start and end at vertex i. Consider a triangle with vertices $\{i, j, k\}$. From vertex i there are exactly two closed walks of length 3 that traverse the triangle: $i \to j \to k \to i$ and $i \to k \to j \to i$. Thus each triangle contributes exactly 2 to each of the three diagonal entries $(A^3)_{ii}, (A^3)_{jj}, (A^3)_{kk}$, i.e. a total contribution of 6 to trace$(A^3) = \sum_i (A^3)_{ii}$.

Since different triangles contribute to disjoint sets of closed walks, summing over all vertices counts every triangle exactly 6 times. Therefore the number of (undirected) triangles in G is

$$\frac{1}{6}\,\mathrm{trace}\left(A(G)^3\right).$$

4. **Problem 4:** Is the matrix

$$K = \begin{pmatrix} 3 & 5 \\ 5 & 4 \end{pmatrix}$$

positive semi-definite? (1 Pt.)

Ans: A symmetric 2×2 matrix $\begin{pmatrix} a & b \\ b & d \end{pmatrix}$ is positive semi-definite iff $a \geq 0$ and $ad - b^2 \geq 0$.

For K, we have $a = 3 > 0$ and

$$ad - b^2 = 3 \times 4 - 5^2 = 12 - 25 = -13 < 0.$$

Hence, K is **not positive semi-definite**. It is an **indefinite** matrix since it has both positive and negative eigenvalues.

5. **Problem 5:**

Let G be a graph, consider another variant of the k-WL, which aggregates colors of adjacent k-tuples as follows,

$$\begin{aligned} M_{t,\square}(\mathbf{v}) := (&\{\{(C^k_{t,\square}(\phi_1(\mathbf{v}, w))) \mid w \in N(v_1)\}\}, \dots, \\ &\{\{(C^k_{t,\square}(\phi_k(\mathbf{v}, w))) \mid w \in N(v_k)\}\}), \end{aligned}$$

resulting in the coloring function $C^k_{t,\square}(\mathbf{v}) := \mathrm{Recolor}\Big((C^k_{(t-1,\square)}(\mathbf{v}), M_{(t-1,)}(\mathbf{v}))\Big)$. Everything else is defined in the same way as for the k-WL.

Implement, for general k,

1. the k-WL and
2. the above variant

in *Python* using NetworkX. Benchmark the algorithm on graphs generated via the Erdős–Rényi model with different number of edges. The Erdős–Rényi model, using a parameter $p \in [0, 1]$, is a random graph model where we connect two vertices by an edge with probability p.

What can you observe for the computation time of the two algorithms when varying p? (10 Pt.)

Ans: Python file attached.