Medical Data Science, WS 2023/2024

Prof. Dr. Nico Pfeifer

22.11.2023

Oğuz Ata Çal 6661014

Edward Beach 5451904

Assignment 3

Problem 1

a) Multi-task Learning is an extension of Dual-task learning. There are different models trained for different domains (both source and target domains), and the pairwise weight-vector difference of each model is minimized. In Dual-task learning, the same is done with only 2 domains (target and source). As can be seen from slide 4, pages 12 and 14, Dual-task learning performs better when the source and target domain are close (i.e., the task is similar). Multi-task learning performs on par for one of the organisms (A.thaliana) but performs worse than Dual-task learning for all other organisms. So, for this particular example, Dual-learning is best.

b) The MHC is a genomic region found in all vertebrates, and it plays a crucial role in the immune system. HLA specifically refers to the MHC in humans. Therefore, HLA is a subset of the MHC. The main function of HLA I is to present intracellular antigens (peptides derived from proteins within the cell) to cytotoxic T cells.

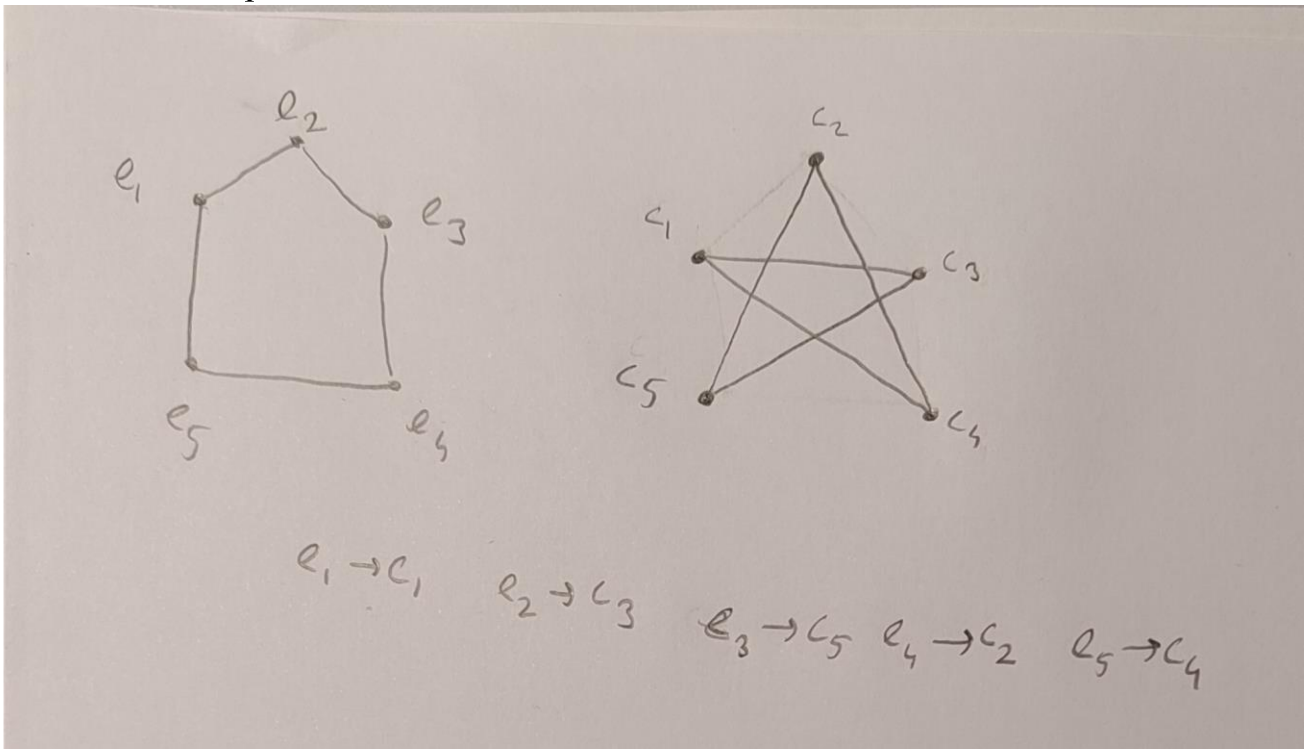
c) Leveraging is encoding other information as features such as HLA information, supertype information, and so on. This leads to high-dimensional feature spaces. These features can be used with several learning models.

d) We encounter graphs for example in a protein function prediction task. We can predict the function of the protein from the structure of it, which can be modeled as a graph. So, taking similarities and analyzing the underlying graphs of proteins is interesting. In neuroscience, graphs are employed to model and analyze brain connectivity networks. This helps in understanding how different regions of the brain interact and how disruptions in these networks may be associated with neurological disorders.

e) The complexity is a problem in the computation of graph similarity because of the intricate structural characteristics, and size of graphs, coupled with challenges in isomorphism detection. The problem of computing graph similarity is currently addressed through a combination of approximation techniques, heuristics, graph kernels, subgraph isomorphism algorithms, and machine learning-based approaches. Two approaches to handle this problem are

Graph kernels and graph edit. Graph kernels count common substructures for local pattern similarity, while graph edit distance measures the minimum editing cost for global and local differences. Kernels offer flexibility but may be sensitive, while edit distance is robust but computationally expensive for large graphs. It is difficult to say that one is better than the other overall.

f) Two graphs are isomorphic if they are structurally identical, meaning their vertices can be one-to-one mapped in a way that preserves the edge relationship. Here is an example.


e_1
e_2
e_3
e_4
e_5
c_1
c_2
c_3
c_4
c_5
$e_1 \to c_1$ $e_2 \to c_3$ $e_3 \to c_5$ $e_4 \to c_2$ $e_5 \to c_4$

The graphs are isomorphic since by the mapping provided, they have the same number of cycles, and the nodes are adjacent to the same nodes as in the map on the left.

g) A size-k graphlet of G is defined as a subgraph of G of size k. Both a graphlet kernel and previously discussed kernels operate in the same principle that they are the inner product of the occurrence matrices of two concepts. The difference is that in the previously discussed kernels, we were working on all of the data (all of the DNA sequence) but here because comparing the entire structure of graphs is too complex, we transform the data and look at only a part of the graph (subgraphs of size k i.e., graphlets).

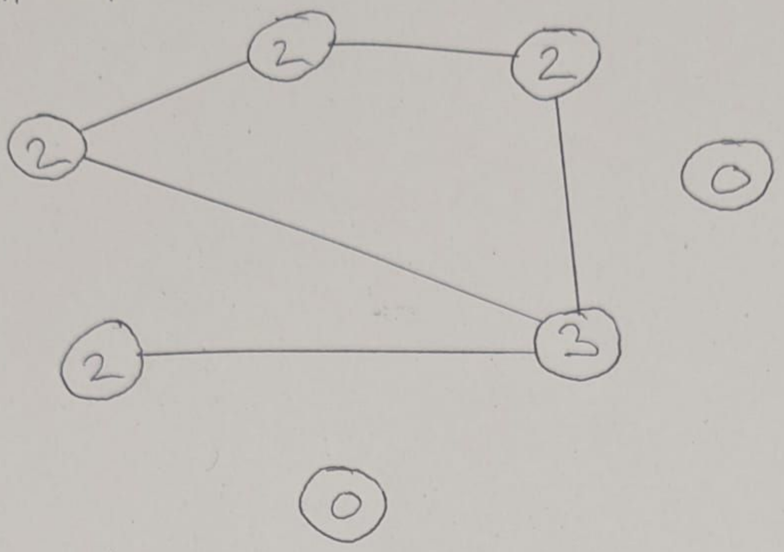
h) WL subtree kernel utilizes the WL graph isomorphism test, iteratively refining vertex labels and counting the common refined labels between two graphs. WL edge kernel counts the matching pairs of edges with identically labeled endpoints in two graphs and thus provides a measure of how similar the graphs are based on edge label correspondence.

Problem 2

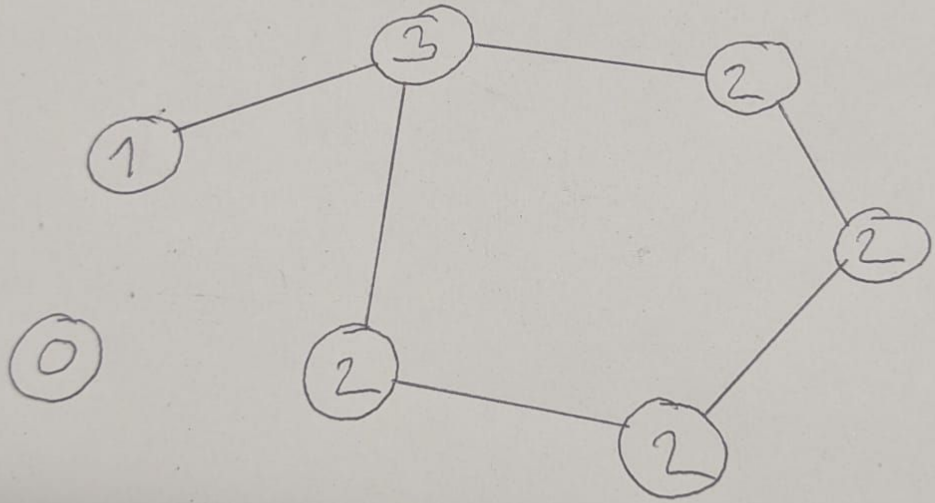
The answers along with the code are provided in the part2.ipynb file.

Problem 3

a) Graph G_1:



Graph G_2:



b) Iteration 1: Relabling each node with their respective multiset (Here in order of the nodes in the adjacency matrix):

Graph G_1: 2,23 ; 2,22 ; 2,23 ; 0 ; 3,122 ;

0 ; 1,3

G_2: 1,3 ; 3,122 ; 2,23 ; 2,22 ; 2,22 ;

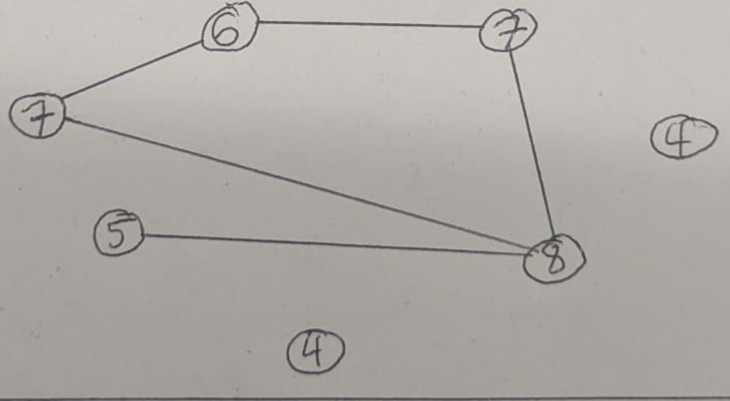
2,23 ; 0

Mapping each label to a new label:

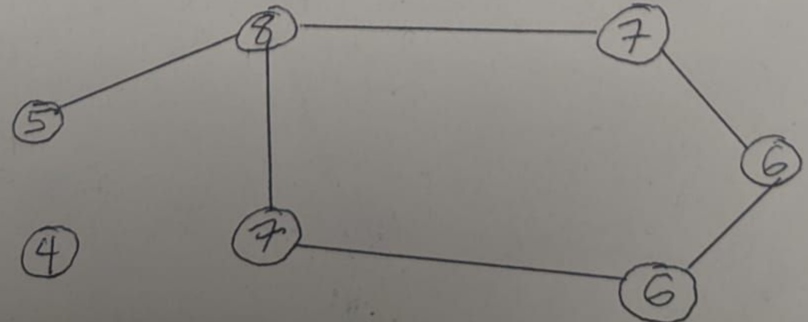
$0 \rightarrow 4$	$2,23 \rightarrow 7$
$1,3 \rightarrow 5$	$3,122 \rightarrow 8$
$2,22 \rightarrow 6$	

Relabeling the graphs:

G_1:



G_2:



Iteration 2: Relabeling each node with their multiset:

G_1: 7,68 ; 6,77 ; 7,68 ; 4 ; 8,577;
4 ; 5,8

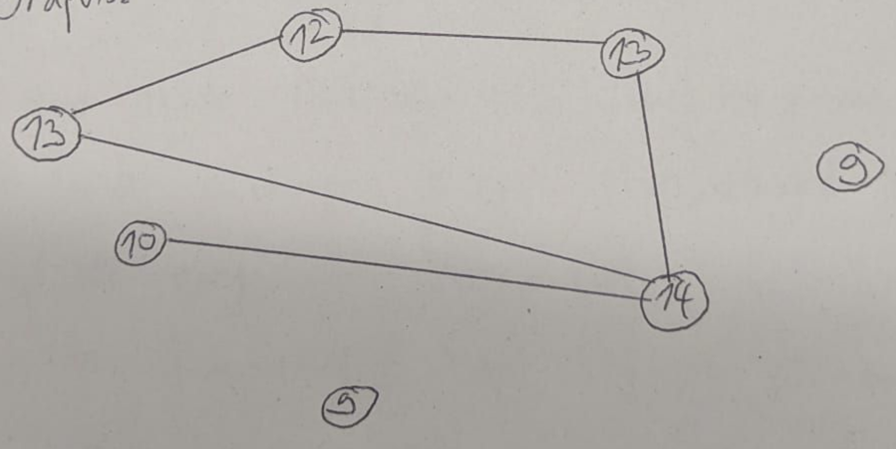
G_2: 5,8 ; 8,577 ; 7,68 ; 6,67 ; 6,67;
7,68 ; 4

Mapping multiset labels to new labels:

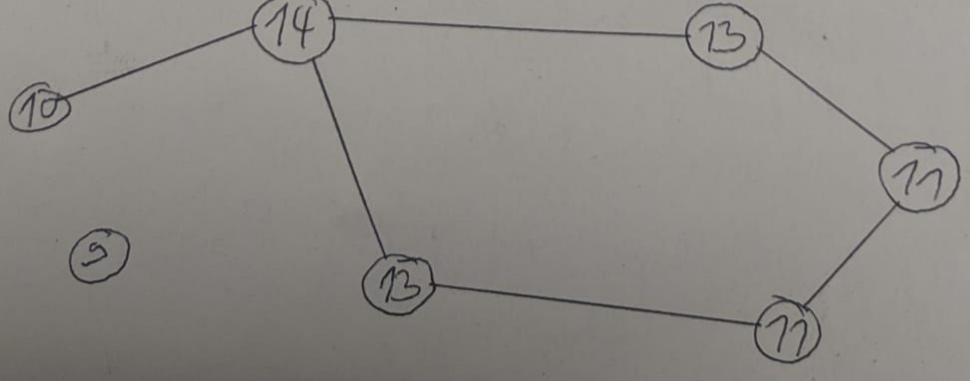
$4 \rightarrow 9$	$6,77 \rightarrow 12$
$5,8 \rightarrow 10$	$7,68 \rightarrow 13$
$6,67 \rightarrow 11$	$8,577 \rightarrow 14$

Relabeled Graphs:

G_1:



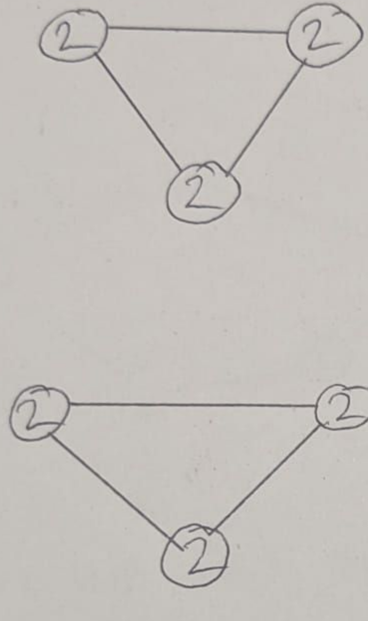
G_2:



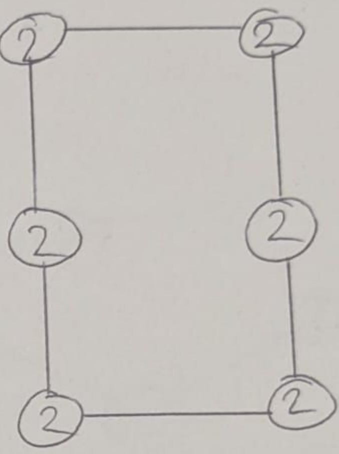
The algorithm terminates after iteration 2, since the graphs have a different set of labels $\Rightarrow G_1$ and G_2 are not an isomorphism.

c) Counterexample 1:

G_1:



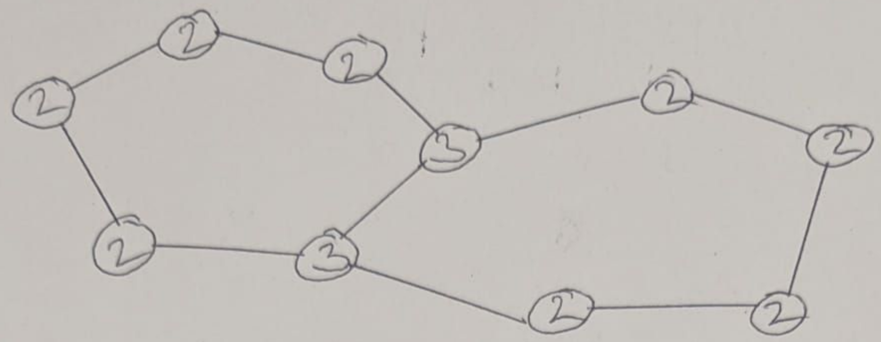
G_2:



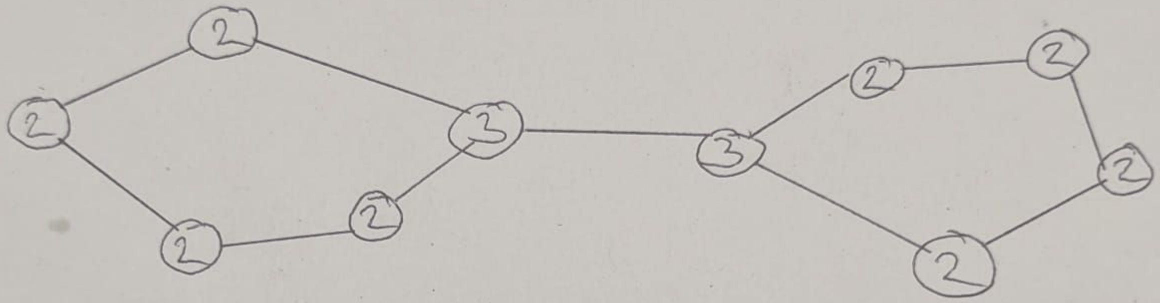
Since every node has exactly two neighbours, the algorithm will always assign the exact same multisets / relabeling and thus run indefinitely. However, since G_1 is disconnected and G_2 is connected, they are not an isomorphism.

Counter example 2

G_1:



G_2:



Here G_1 and G_2 are not an isomorphism, diffrent to counter example 1, both Graphs are connected graphs. Here the algorithm will also run infinitely, Since it will never assign diffrent multisets/relabeling to the diffrent graphs.