TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA Công Nghệ Thông Tin
BỘ MÔN: Công Nghệ Phần Mềm

ĐỀ THI VÀ BÀI LÀM

Tên học phần: Trí tuệ nhân tạo
Mã học phần: Hình thức thi: *Tự luận có giám sát*
Đề số: **00002** Thời gian làm bài: 70 phút *(không kể thời gian chép/phát đề)*
Được sử dụng tài liệu khi làm bài.

Họ tên: Võ Tuấn Mạnh Hùng **Lớp**: 19TCLC_DT1 **MSSV**: 102190017

Sinh viên làm bài trực tiếp trên tệp này, lưu tệp với định dạng MSSV_HọTên.pdf và nộp bài thông qua MSTeam:

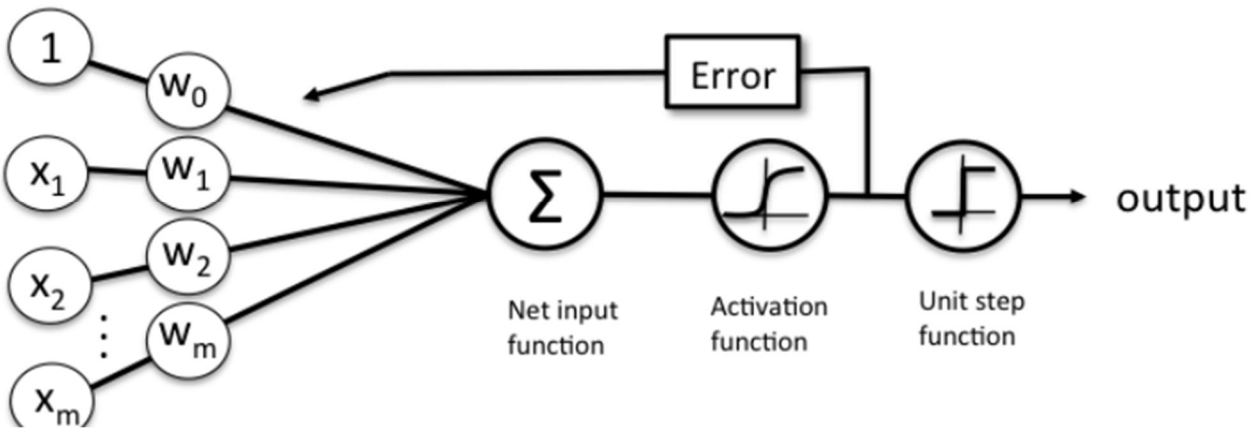
Câu 1 (*4 điểm*): Cho tập dữ liệu input.csv với 80 mẫu dữ liệu, mỗi mẫu có 4 đặc trưng (chiều dài đài hoa, chiều rộng đài hoa, chiều dài cánh hoa, chiều rộng cánh hoa) và tên loài hoa tương ứng.

a) *(3 điểm)* Hãy viết chương trình phân loại hoa sử dụng Logistic Regression. Nêu rõ mô hình thức phân loại trong chương trình như thế nào (Ví dụ: có bao nhiêu tế bào nơ-ron, mỗi nơ-ron phụ trách công việc gì, làm sao để phân loại,…)?

 # Trả lời: Dán code vào bên dưới

```
import numpy as np
import pandas as pd
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
data = pd.read_csv('input.csv')
data.head()
X=data.drop(['Labeling'], axis=1)
y=data["Labeling"]

print(X.shape)
print(y.shape)
logreg = LogisticRegression()
logreg.fit(X, y)
```

 # Trả lời: Mô tả mô hình phân loại bằng hình ảnh hoặc bằng lời.



b) *(1 điểm)* Hãy thực thi chương trình và cho biết nhãn của 10 mẫu dữ liệu trong output2.csv

 # Trả lời: Dán code thực thi thành công

```
#data test
test=pd.read_csv("/content/output_02.csv")
#print(test)
X_test=test.drop(['Labeling'], axis=1)
print(X_test.shape)
kq=logreg.predict(X_test)
print(kq)
test["Labeling"]=kq
test
```

 # Trả lời: Dán kết quả nhãn ứng với 10 mẫu dữ liệu

	sepal_length	sepal_width	petal_length	petal_width	Labeling
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.7	3.0	4.2	1.2	Iris-versicolor
6	5.7	2.9	4.2	1.3	Iris-versicolor
7	6.2	2.9	4.3	1.3	Iris-versicolor
8	5.1	2.5	3.0	1.1	Iris-versicolor
9	5.7	2.8	4.1	1.3	Iris-versicolor

Câu 2 (2 *điểm*): Cho không gian Oxyz với 6 điểm có tọa độ tương ứng (0,4,1), (4,1,0) (3,2,0),(2,3,2),(2,4,1) và (2,3,4).

a) *(1 điểm)* Viết hàm thực thi thuật toán *k*-means

 # Trả lời: Dán code vào bên dưới

```
def kmeans_init_centers(X, k):
    # randomly pick k rows of X as initial centers
    return X[np.random.choice(X.shape[0], k, replace=False)]


def kmeans_assign_labels(X, centers):
    # calculate pairwise distances btw data and centers
```

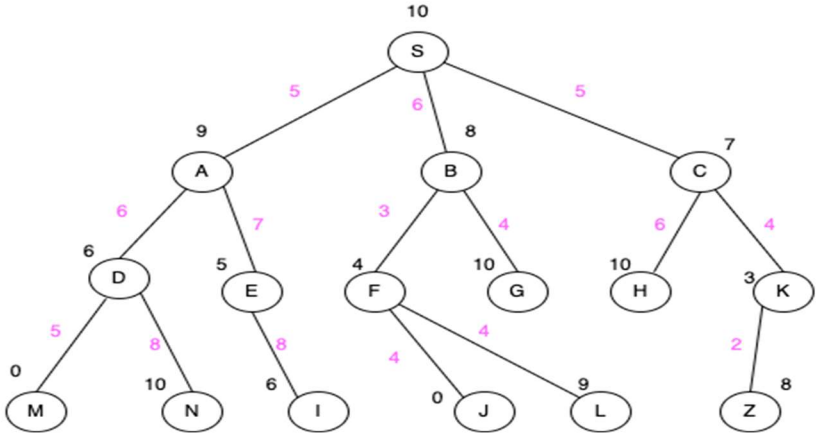
```
    D = cdist(X, centers)
    # return index of the closest center
    return np.argmin(D, axis=1)


def kmeans_update_centers(X, labels, K):
    centers = np.zeros((K, X.shape[1]))
    for k in range(K):
        # collect all points assigned to the k-th cluster
        Xk = X[labels == k, :]
        # take average
        centers[k, :] = np.mean(Xk, axis=0)
    return centers


def has_converged(centers, new_centers):
    # return True if two sets of centers are the same
    return (set([tuple(a) for a in centers]) == set([tuple(a) for a in new_cen
ters]))


def kmeans(X, K):
    centers = [kmeans_init_centers(X, K)]
    labels = []
    it = 0
    while True:
        labels.append(kmeans_assign_labels(X, centers[-1]))
        new_centers = kmeans_update_centers(X, labels[-1], K)
        if has_converged(centers[-1], new_centers):
            break
        centers.append((new_centers))
        it += 1
    return (centers, labels, it)
```

b) *(1 điểm)* Nếu sử dụng thuật toán *k*-means với k = 3 thì kết quả phân nhóm sẽ như thế nào? (các điểm thuộc mỗi nhóm, trọng tâm của mỗi nhóm).

 # Trả lời: viết câu trả lời vào bên dưới

(0,4,1) thuộc nhóm 1
(2,4,1) thuộc nhóm 1.
Trọng tâm nhóm 1: (1,4,1)
(4,1,0) thuộc nhóm 2
(3,2,0) thuộc nhóm 2
Trọng tâm nhóm 2: (3.5,1.5,0)
(2,3,2) thuộc nhóm 3
(2,3,4) thuộc nhóm 3
Trọng tâm nhóm 3: (2,3,3)

Câu 3 (*4 điểm*): Cho cây G = (V,E) như hình vẽ với V là tập đỉnh và E là tập cạnh, trọng số tại đỉnh là hàm ước lượng khoảng cách từ đỉnh đó đến trạng thái đích(giá trị các nhỏ thì càng gần trạng thái đích), trọng số trên cạnh thể hiện chi phí phải trả khi đi qua cạnh.



a) *(2 điểm)* Hãy viết đoạn code biểu diễn đồ thị trên bằng cách khởi tạo tập đỉnh V, tập cạnh E, trọng số trên đỉnh và trọng số trên cạnh.

 # Trả lời: Dán code vào bên dưới

```
class Node:

    def __init__(self, label, goal_cost):

        self.label = label

        self.cost = 10000

        self.goal_cost = goal_cost

        self.save_cost = None

        self.pr = []

        self.chld = []

    def __repr__(self):

        return str(dict({

            "label": self.label,

            "cost" : self.cost,

            "goal cost": self.goal_cost

        }))

    def __eq__(self, other):

        return self.label == other.label
```

```
    def __lt__(self, other):
        if self.save_cost == 10000:
            return self.goal_cost + self.cost < other.goal_cost + other.cost
        else:
            return self.cost < other.cost


    def get_label(self):
        return self.label


    def neighbors(self):
        return self.chld + self.pr


class Tree:
    def __init__(self):
        self.nodes = []
        self.edges = []


    def add_nodes(self, data):
        for node in data:
            self.nodes.append(node)


    def add_node(self, node):
        self.nodes.append(node)
```

```
    def get_index(self, node):
        for i, n in enumerate(self.nodes):
            if n.get_label() == node.get_label():
                return i
        return -1


    def add_edges(self, tuple_edges):
        for t in tuple_edges:
            start_label = t[0]
            end_label = t[1]
            w = t[2]
            index_start_label = self.get_index(Node(start_label, None))
            index_end_label = self.get_index(Node(end_label, None))


            self.nodes[index_start_label].chld.append(self.nodes[index_end_label])
            self.nodes[index_end_label].pr.append(self.nodes[index_start_label])
            self.edges.append((self.nodes[index_start_label], self.nodes[index_end_label],t[2]))


    def show_nodes(self):
        return [node.get_data() for node in self.nodes]
    def get_edge(self, start_node, end_node):
        try:
```

```
            return [edges for edges in self.edges if edges[0] == start_node

                    and edges[1] == end_node][0]

        except:

            return None

def update_cost(tree, current_node, prev_node):

    if tree.get_edge(prev_node, current_node) is not None:

        #print("OK")

        if current_node.cost > prev_node.cost + tree.get_edge(prev_node, curren
t_node)[2]:

            current_node.cost = prev_node.cost + tree.get_edge(prev_node, curre
nt_node)[2]



if __name__ == "__main__":

    tree = Tree()

    tree.add_nodes([

        Node("S", 10),

        Node("A", 9),

        Node("B", 8),

        Node("C", 7),

        Node("D", 6),

        Node("E", 5),

        Node("F", 4),

        Node("G", 10),

        Node("H", 10),

        Node("K", 3),

        Node("M", 0),

        Node("N", 10),
```

```
        Node("I", 6),

        Node("J", 0),
        Node("L", 9),

        Node("Z", 8)

    ])

    tree.add_edges([

        ("S", "A", 5),

        ("S", "B", 6),

        ("S", "C", 5),

        ("A", "D", 6),

        ("A", "E", 7),

        ("B", "F", 3),

        ("B", "G", 4),

        ("C", "H", 6),

        ("C", "K", 4),

        ("D", "M", 5),

        ("D", "N", 8),

        ("E", "I", 8),

        ("F", "J", 4),

        ("F", "L", 4),
         ("K", "Z", 4)

    ])

    tree.nodes[0].cost = 0
```

b) *(2 điểm)* Hãy viết chương trình sử dụng thuật toán **A*** để tìm đường đi từ đỉnh "S" đến các đỉnh có trọng số trên đỉnh bằng 0. Trong chương trình, hãy in ra thứ tự đỉnh khám phá trong quá trình tìm kiếm.

Trả lời: Dán code vào bên dưới

```
import heapq
def A_Star(tree, start, end):

    frontier = [start]

    heapq.heapify(frontier)

    explored = []

    while len(frontier) > 0:

        state = heapq.heappop(frontier)

        explored.append(state)

        print(state)

        if state == end:

            return explored

        for child in state.neighbors():

            update_cost(tree, child, state)

            if child.get_label() not in list(set(node.get_label() for node in f
rontier + explored)):

                heapq.heappush(frontier, child)

    return False
if __name__ == "__main__":

    print("Dinh thu 1-------\n")
    result = A_Star(tree, tree.nodes[0], tree.nodes[10])

    if result:

        s = 'explored: '

        for i in result:

            s += i.label + " "

            print(s);

    else:

        print('404 Not Found!')

    print("Dinh thu 2-------\n")
    result = A_Star(tree, tree.nodes[0], tree.nodes[13])
```

```
    if result:
        s = 'explored: '
        for i in result:
            s += i.label + " "
            print(s);
    else:
        print('404 Not Found!')
```

Trả lời: Dán kết quả thực thi vào bên dưới

```
Dinh thu 1-------

{'label': 'S', 'cost': 0, 'goal cost': 10}
{'label': 'A', 'cost': 5, 'goal cost': 9}
{'label': 'C', 'cost': 5, 'goal cost': 7}
{'label': 'B', 'cost': 6, 'goal cost': 8}
{'label': 'K', 'cost': 9, 'goal cost': 3}
{'label': 'F', 'cost': 9, 'goal cost': 4}
{'label': 'G', 'cost': 10, 'goal cost': 10}
{'label': 'H', 'cost': 11, 'goal cost': 10}
{'label': 'D', 'cost': 11, 'goal cost': 6}
{'label': 'E', 'cost': 12, 'goal cost': 5}
{'label': 'Z', 'cost': 13, 'goal cost': 8}
{'label': 'J', 'cost': 13, 'goal cost': 0}
{'label': 'L', 'cost': 13, 'goal cost': 9}
{'label': 'M', 'cost': 16, 'goal cost': 0}
explored: S
explored: S A
explored: S A C
explored: S A C B
explored: S A C B K
explored: S A C B K F
explored: S A C B K F G
explored: S A C B K F G H
explored: S A C B K F G H D
explored: S A C B K F G H D E
explored: S A C B K F G H D E Z
explored: S A C B K F G H D E Z J
explored: S A C B K F G H D E Z J L
explored: S A C B K F G H D E Z J L M
```

```
Dinh thu 2-------

{'label': 'S', 'cost': 0, 'goal cost': 10}
{'label': 'A', 'cost': 5, 'goal cost': 9}
{'label': 'C', 'cost': 5, 'goal cost': 7}
{'label': 'B', 'cost': 6, 'goal cost': 8}
{'label': 'K', 'cost': 9, 'goal cost': 3}
{'label': 'F', 'cost': 9, 'goal cost': 4}
{'label': 'G', 'cost': 10, 'goal cost': 10}
{'label': 'H', 'cost': 11, 'goal cost': 10}
{'label': 'D', 'cost': 11, 'goal cost': 6}
{'label': 'E', 'cost': 12, 'goal cost': 5}
{'label': 'Z', 'cost': 13, 'goal cost': 8}
{'label': 'J', 'cost': 13, 'goal cost': 0}
explored: S
explored: S A
explored: S A C
explored: S A C B
explored: S A C B K
explored: S A C B K F
explored: S A C B K F G
explored: S A C B K F G H
explored: S A C B K F G H D
explored: S A C B K F G H D E
explored: S A C B K F G H D E Z
explored: S A C B K F G H D E Z J
```

Đà Nẵng, ngày 5 tháng 06 năm 2022

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)