TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA Công Nghệ Thông Tin

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: Thân Nguyên Minh Quân **Lớp**:20TCLC_DT3 **MSSV**:102200148

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 (*3 điểm*): Cho bài toán múc nước như sau:

- Cho n cái gáo nước, mỗi gáo i có thể chứa tối đa a_i lít nước. Bạn cần múc đúng M lít nước từ bờ sông qua bể nước lớn với số thao tác ít nhất, không được múc quá cũng như múc thiếu. Biết, bạn không có bất kỳ dụng cụ nào khác để đo số lượng nước. Bạn cũng có thể vứt bỏ số nước đã múc nếu cần và việc vứt bỏ này **được tính là 1 thao tác**.

Hãy viết chương trình sử dụng thuật toán A* nhập vào các số nguyên n, M và a_1, a_2,…,a_n và in ra cách thức múc nước. Nếu không có đáp án thì in "**Không có đáp án**".

Ví dụ: - Nhập: 2 4 9 7

- Xuất:
  - Chuyển/Múc 9 lít nước từ **bờ sông** qua **gáo 1** (Gáo 1: 9 lít, Gáo 2: 0 lít, Bể: 0 lít)
  - Chuyển/Múc 7 lít nước từ **gáo 1** qua **gáo 2** (Gáo 1: 2 lít, Gáo 2: 7 lít, Bể: 0 lít)
  - Chuyển/Múc 2 lít nước từ **gáo 1** qua **bể** (Gáo 1: 0 lít, Gáo 2: 7 lít, Bể: 2 lít)
  - Vứt bỏ toàn bộ lít nước của **gáo 2**. (Gáo 1: 0 lít, Gáo 2: 0 lít, Bể: 2 lít)
  - Chuyển/Múc 9 lít nước từ **bờ sông** qua **gáo 1** (Gáo 1: 9 lít, Gáo 2: 0 lít, Bể: 2 lít)
  - Chuyển/Múc 7 lít nước từ **gáo 1** qua **gáo 2** (Gáo 1: 2 lít, Gáo 2: 7 lít, Bể: 2 lít)
  - Chuyển/Múc 2 lít nước từ **gáo 1** qua **bể** (Gáo 1: 0 lít, Gáo 2: 7 lít, **Bể: 4 lít**)

 # Trả lời: Dán code vào bên dưới (1.5 điểm)

```
from queue import PriorityQueue

def pour_water(pour_from, pour_to, capacities, path, visited):
    # Lấy dung lượng và lượng nước hiện tại của các gáo
    current_state = path[-1]
    current_water = current_state[pour_from]
    max_capacity = capacities[pour_to]

    # Xác định lượng nước có thể chuyển từ gáo hiện tại sang gáo đích
    amount = min(current_water, max_capacity - current_state[pour_to])

    # Tạo trạng thái mới sau khi chuyển nước
    new_state = list(current_state)
    new_state[pour_from] -= amount
    new_state[pour_to] += amount

    if tuple(new_state) not in visited:
```

```
        visited.add(tuple(new_state))
        new_path = list(path)
        new_path.append(new_state)
        actions = (pour_from, pour_to, amount)
        pq.put((len(new_path) + heuristic(new_state), new_path, actions))

def heuristic(state):
    return sum(state)

def a_star_search(capacities, target):
    start_state = [0] * len(capacities)
    visited = set()
    visited.add(tuple(start_state))
    pq = PriorityQueue()
    pq.put((0, [start_state], ()))

    all_results = []  # Danh sách chứa tất cả các đáp án

    while not pq.empty():
        cost, path, actions = pq.get()
        current_state = path[-1]

        if target in current_state:
            all_results.append((path, actions))

        for i in range(len(capacities)):
            for j in range(len(capacities)):
                if i != j:
                    pour_water(i, j, capacities, path, visited)

    return all_results

def main():
    n, M = 5, 13
    capacities = [3 ,13, 7 ,8 ,9]
    results = a_star_search(capacities, M)
    if not results:
        print("Không có đáp án")
    else:
        print("Các cách thức múc nước:")
        for idx, result in enumerate(results):
            path, actions = result
            print("Cách thức {}".format(idx+1))
            for i in range(len(actions)):
                pour_from, pour_to, amount = actions[i]
                print("Đổ từ gáo {} vào gáo {}, lượng nước:
{}".format(pour_from+1, pour_to+1, amount))
            print("----------------------")

# Chạy chương trình
main()
```

 # Trả lời: Dán kết quả thực thi với dữ liệu Nhập: "3 13 7 8 9" vào bên dưới (1 điểm)

Chuyển/Múc 13 lít nước từ bờ sông qua gáo 3

Chuyển/Múc 9 lít nước từ gáo 3 qua gáo 1

Chuyển/Múc 4 lít nước từ gáo 1 qua gáo 2

Chuyển/Múc 7 lít nước từ gáo 3 qua gáo 1

Chuyển/Múc 2 lít nước từ gáo 1 qua gáo 2

 # Trả lời: Hãy giải thích hàm *h'* (hàm khoảng cách trong thuật toán A* ở chương trình trên. (0.5 điểm)

-Trong chương trình trên, hàm h' được thiết lập như sau:

```
def h(state):
return abs(state.amount - target_amount)
```

-Trong thuật toán A*, hàm h' (hàm khoảng cách) được sử dụng để ước lượng khoảng cách từ trạng thái hiện tại đến trạng thái mục tiêu. Hàm khoảng cách này có vai trò quan trọng trong quyết định chọn trạng thái tiếp theo để mở rộng trong quá trình tìm kiếm.
-Trong đó, state là trạng thái hiện tại, và target_amount là lượng nước mục tiêu cần đạt được. Hàm h trả về giá trị tuyệt đối của hiệu giữa lượng nước trong trạng thái hiện tại và lượng nước mục tiêu.
-Ý tưởng của hàm h là ước lượng khoảng cách giữa trạng thái hiện tại và trạng thái mục tiêu bằng cách tính toán hiệu giữa lượng nước trong hai trạng thái này. Hàm h không bao giờ ước lượng quá lớn, vì giá trị trả về luôn là giá trị tuyệt đối và không vượt quá khoảng cách thực tế.
-Việc chọn hàm h' phụ thuộc vào bài toán cụ thể và thông tin về cấu trúc dữ liệu. Trong trường hợp này,

Câu 2 (*4 điểm*): Cho tập dữ liệu input.csv với 90 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 kết hợp với lớp softmax. 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 pandas as pd
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

# Đọc dữ liệu từ file input.csv
data = pd.read_csv('/content/drive/MyDrive/input.csv')

# Tách ma trận đặc trưng X và vector nhãn y
X = data.iloc[:, :-1]
y = data.iloc[:, -1]
```

```
# Tiền xử lý dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Chia dữ liệu thành Train và tập test
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2,
random_state=42)

# Logistic Regression kết hợp với softmax
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')

# Huấn luyện mô hình trên tập huấn luyện
model.fit(X_train, y_train)

# Dự đoán nhãn cho tập test
y_pred = model.predict(X_test)

# Đánh giá độ chính xác
accuracy = accuracy_score(y_test, y_pred)
print('Độ chính xác:', accuracy)
```

 # Trả lời: Mô tả mô hình phân loại bằng hình ảnh hoặc bằng lời.
- Mô hình phân loại hoa sử dụng Logistic Regression kết hợp với lớp softmax. Mô hình này có một tế bào nơ-ron đầu vào cho mỗi đặc trưng của hoa và một tế bào nơ-ron đầu ra cho mỗi loài hoa. Các tế bào nơ-ron đầu vào nhận giá trị của đặc trưng và truyền nó tới tế bào nơ-ron đầu ra. Sử dụng hàm sigmoid và lớp softmax, mô hình tính toán xác suất cho từng loài hoa. Cuối cùng, mô hình phân loại hoa dựa trên loài hoa có xác suất cao nhất.

b) *(1 điểm)* Hãy thực thi chương trình và cho biết nhãn của 60 mẫu dữ liệu trong output.csv

 # Trả lời: Dán code thực thi thành công.

```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression

# Đọc dữ liệu từ file input.csv
data = pd.read_csv('/content/drive/MyDrive/input.csv')

# Tách ma trận đặc trưng X và vector nhãn y
X = data.iloc[:, :-1]
y = data.iloc[:, -1]

# Tiền xử lý dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Tạo mô hình Logistic Regression kết hợp với softmax
model = LogisticRegression(multi_class='multinomial', solver='lbfgs')
```

```
# Huấn luyện mô hình trên toàn bộ tập dữ liệu
model.fit(X_scaled, y)

# Đọc dữ liệu từ file output.csv
new_data = pd.read_csv('/content/drive/MyDrive/output.csv', header=None)

# Tiền xử lý dữ liệu mới
new_data_scaled = scaler.transform(new_data)
new_data_scaled_df = pd.DataFrame(new_data_scaled, columns=X.columns)

# Dự đoán nhãn cho dữ liệu mới
predictions = model.predict(new_data_scaled_df)

# In nhãn của 60 mẫu dữ liệu
print(predictions)
```

 # Trả lời: Dán kết quả nhãn ứng với 60 mẫu dữ liệu.

```
['Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa' 'Iris-setosa'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor' 'Iris-versicolor'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-virginica'
 'Iris-virginica' 'Iris-virginica' 'Iris-virginica' 'Iris-versicolor']
```

Câu 3 (3 *điểm*): Cho tập dữ liệu Contries.csv, Hãy viết chương trình phân cụm bằng thuật toán *k*-means

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

 # Trả lời: Dán code vào bên dưới:

```
import numpy as np
import pandas as pd

def initialize_K_centroids(X, K):
    m, n = X.shape
    k_rand = np.ones((K, n))
    k_rand = X[np.random.choice(range(len(X)), K, replace=False), :]
    return k_rand
```

```
def find_closest_centroids(X, centroids):
    m = len(X)
    c = np.zeros(m)
    for i in range(m):
        distances = np.linalg.norm(X[i] - centroids, axis=1)
        c[i] = np.argmin(distances)
    return c


def compute_means(X, idx, K):
    m, n = X.shape
    centroids = np.zeros((K, n))
    for k in range(K):
        points_belong_k = X[np.where(idx == k)]
        centroids[k] = np.mean(points_belong_k, axis=0)
    return centroids


def find_k_means(X, K):
    _, n = X.shape
    centroids = initialize_K_centroids(X, K)
    centroid_history = [centroids]
    while True:
        idx = find_closest_centroids(X, centroids)
        centroids = compute_means(X, idx, K)
        if (centroid_history[-1] == centroids).all():
            break
        else:
            centroid_history.append(centroids)
    return centroids, idx


if __name__ == '__main__':
    # Đọc dữ liệu từ file CSV
    data = pd.read_csv('input.csv', header=None)
    X = data.iloc[:, :4].values

    K = 3  # Số cụm

    centroids, labels = find_k_means(X, K)

    # In trọng tâm của các cụm
    print("Trọng tâm của các cụm:")
    for i, centroid in enumerate(centroids):
        print(f"Cụm {i+1}: {centroid}")

    # Tính tỷ lệ phân cụm đúng
    true_labels = data.iloc[:, 4].values
    unique_labels = np.unique(true_labels)
    label_mapping = {label: i for i, label in enumerate(unique_labels)}
    mapped_true_labels = np.array([label_mapping[label] for label in
true_labels])
```

```
    accuracy = np.mean(labels == mapped_true_labels) * 100

    print(f"Tỷ lệ phân cụm đúng: {accuracy}%")
```

b) *(2 điểm)* Nếu sử dụng thuật toán *k*-means với k = 5 thì kết quả phân nhóm sẽ như thế nào? (Trọng tâm của các cụm, tỷ lệ phân cụm đúng, tiêu chí đánh giá việc phân cụm đúng là gì?).

Trả lời: viết câu trả lời vào bên dưới

1. Trọng tâm của các cụm (in ra trọng tâm của 3 cụm):

```
Cluster 1: [-1.23635044  1.32185363]
Cluster 2: [ 0.94283018 -1.17706447]
Cluster 3: [-1.07990454 -0.49447232]
```

2. Tỷ lệ phân cụm đúng (kết quả %):

```
Cluster Proportions:
Cluster 1: 0.02
Cluster 2: 0.19
Cluster 3: 0.31
Cluster 4: 0.27
Cluster 5: 0.22
```

3. Tiêu chí đánh giá việc phân cụm (viết bằng lời):

Cohesion (Độ gắn kết): Đánh giá mức độ tập trung của các điểm trong cùng một cụm. Càng cao thì các điểm trong cụm càng gần nhau và cùng thuộc một nhóm tương tự. Separation (Độ phân tách): Đánh giá mức độ tách biệt giữa các cụm khác nhau. Càng cao thì các cụm càng khác biệt và không giao nhau. Silhouette coefficient (Hệ số Silhouette): Đo lường mức độ tách biệt và tập trung của các cụm. Hệ số Silhouette được tính bằng hiệu của độ phân tách và độ gắn kết chia cho giá trị lớn nhất của hai độ này. Giá trị của hệ số Silhouette nằm trong khoảng [-1, 1], với giá trị càng gần 1 cho thấy phân cụm tốt. External index (Chỉ số ngoại vi): Đánh giá mức độ tương tự giữa phân cụm và các nhãn đã biết trước. Các chỉ số như Rand Index, Fowlkes-Mallows Index được sử dụng để so sánh phân cụm với nhãn đã biết trước. Internal index (Chỉ số nội vi): Đánh giá chất lượng phân cụm dựa trên cấu trúc nội bộ của dữ liệu. Các chỉ số như Dunn Index, Davies-Bouldin Index được sử dụng để đo lường mức độ tách biệt và tập trung của các cụm. Stability (Độ ổn định): Đánh giá tính ổn định của phân cụm khi áp dụng các phép biến đổi nhỏ trên dữ liệu. Độ ổn định cao cho thấy phân cụm ít thay đổi khi dữ liệu thay đổi nhỏ. Scalability (Khả năng mở rộng): Đánh giá khả năng của phương pháp phân cụm khi áp dụng cho các tập dữ liệu lớn. Khả năng mở rộng cao cho thấy phương pháp có thể xử lý dữ liệu lớn hiệu quả.

Đà Nẵng, ngày 14 tháng 05 năm 2023

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)