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: **00001** Thời gian làm bài: 70 phút *(không kể thời gian chép/phát đề)*
Được sử dụng tài liệu khi làm bài.

Họ tên…Nguyễn Ngọc Thịnh….**Lớp**:21TCLC_NHAT2……..**MSSV**:…102210377……

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 **không tính là số 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 5 4 3
- Xuất:
  - Chuyển/Múc 4 lít nước từ **bờ sông** qua **gáo 1** (Gáo 1: 4 lít, Gáo 2: 0 lít, Bể: 0 lít)
  - Chuyển/Múc 4 lít nước từ **gáo 1** qua **bể** (Gáo 1: 0 lít, Gáo 2: 0 lít, Bể: 4 lít)
  - Chuyển/Múc 4 lít nước từ **bờ sông** qua **gáo 1** (Gáo 1: 4 lít, Gáo 2: 0 lít, Bể: 4 lít)
  - Chuyển/Múc 3 lít nước từ **gáo 1** qua **gáo 2** (Gáo 1: 1 lít, Gáo 2: 3 lít, Bể: 4 lít)
  - Chuyển/Múc 1 lít nước từ **gáo 1** qua **bể** (Gáo 1: 0 lít, Gáo 2: 3 lít, **Bể: 5 lít**)

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

```
import heapq

def h(state, target):
    return abs(sum(state) - target)

def A(n, M, capacities):
    start_state = tuple([0] * n)
    target = M
    open_states = []
    heapq.heappush(open_states, (0 + h(start_state, target), 0, "", start_state))
    closed_states = set()

    while open_states:
        _, g, actions, current_state = heapq.heappop(open_states)

        if sum(current_state) == target:
            return actions

        closed_states.add(current_state)
```

```
    for i in range(n):
        for j in range(n):
            if i != j:
                transfer = min(current_state[i], capacities[j] - current_state[j])
                if transfer > 0:
                    new_state = list(current_state)
                    new_state[i] -= transfer
                    new_state[j] += transfer
                    new_state = tuple(new_state)
                    if new_state not in closed_states:
                        action = f"Chuyển/Múc {transfer} lít nước từ gáo {i+1} qua gáo {j+1}"
                        heapq.heappush(open_states, (g + 1 + h(new_state, target), g + 1, actions + "\n" +
action, new_state))
                        closed_states.add(new_state)

    for i in range(n):
        if current_state[i] < capacities[i]:
            new_state = list(current_state)
            new_state[i] = capacities[i]
            new_state = tuple(new_state)
            if new_state not in closed_states:
                action = f"Chuyển/Múc {capacities[i]} lít nước từ bờ sông qua gáo {i+1}"
                heapq.heappush(open_states, (g + 1 + h(new_state, target), g + 1, actions + "\n" + action,
new_state))
                closed_states.add(new_state)

        if current_state[i] > 0:
            new_state = list(current_state)
            new_state[i] = 0
            new_state = tuple(new_state)
            if new_state not in closed_states:
                action = f"Vứt bỏ {current_state[i]} lít nước từ gáo {i+1}"
                heapq.heappush(open_states, (g + 1 + h(new_state, target), g + 1, actions + "\n" + action,
new_state))
                closed_states.add(new_state)

    return "Không có đáp án"

# Ví dụ: Nhập: 2 5 [4, 3]
print(A(3, 17, [7, 8, 9]))
```

 # Trả lời: Dán kết quả thực thi với dữ liệu Nhập: "3 17 7 8 9" vào bên dưới (1 điểm)

Chuyển/Múc 9 lít nước từ bờ sông qua gáo 3
Chuyển/Múc 8 lít nước từ bờ sông 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 thuật toán A*, hàm h được sử dụng để ước tính khoảng cách từ trạng thái hiện tại đến trạng thái mục tiêu. Hàm h không cung cấp giá trị chính xác mà chỉ là một ước tính để giúp thuật toán tìm kiếm đường dẫn tối ưu từ trạng thái hiện tại đến trạng thái mục tiêu.

Trong trường hợp bài toán múc nước, hàm h có thể được hiểu như sau:

Hàm h(state, target) tính toán khoảng cách từ trạng thái hiện tại (state) đến trạng thái mục tiêu (target). Trong bài toán múc nước, trạng thái là lượng nước trong các gáo hiện tại, và mục tiêu là tổng lượng nước mà bạn cần đạt được.

Hàm h(state, target) trong bài toán này tính toán giá trị tuyệt đối của hiệu số giữa tổng lượng nước trong trạng thái hiện tại và mục tiêu. Nó đo lường sự khác biệt giữa lượng nước hiện tại và lượng nước cần đạt được. Hàm này không phải lúc nào cũng cho kết quả chính xác, nhưng nó cung cấp một ước tính để giúp thuật toán tìm đường dẫn tối ưu nhanh hơn.

Hàm heuristic thường được sử dụng trong thuật toán A* để ước tính giá trị f* (f* = g + h), trong đó g là giá trị thực tế (số bước đã di chuyển) và h là giá trị ước tính (khoảng cách ước tính). Thuật toán A* sẽ ưu tiên các trạng thái có giá trị f* thấp hơn để tìm kiếm đường dẫn tối ưu đến mục tiêu.

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 nơron, mỗi nơron 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 numpy import genfromtxt
from sklearn.model_selection import train_test_split
def softmax(z):
    e_z = np.exp(z - np.max(z, axis=0))
    return e_z / np.sum(e_z, axis=0)

def logistic_softmax_regression(X, y, w_init, alpha, tol=1e-4, loop=10000):
    w = [w_init]
    N = X.shape[1]
    d = X.shape[0]
    K = len(np.unique(y))
    count = 0
    check_w = 20
    while count < loop:
        mix_id = np.random.permutation(N)
        for i in mix_id:
            xi = X[:, i].reshape(d, 1)
            yi = np.eye(K)[:, y[i]].reshape(K, 1)
            zi = softmax(np.dot(w[-1].T, xi))
            w_new = w[-1] + alpha * np.dot(xi, (yi - zi).T)
            count += 1
            if count % check_w == 0:
                if np.linalg.norm(w_new - w[-check_w]) < tol:
                    return w
            w.append(w_new)
    return w
test = []
with open("output.csv") as f:
```

```
    for val in f.readlines():
        l = val.strip()
        spl = l.split(",")
        if len(spl) > 1:
            test.append([float(i) for i in spl[:]])
test = np.array(test).T
test = np.vstack((np.ones((1, test.shape[1])), test))
x = []
y = []
with open("input.csv") as f:
    for val in f.readlines():
        l = val.strip()
        spl = l.split(",")
        if len(spl) > 1:
            x.append([float(i) for i in spl[:-1]])
            if spl[-1] == "Iris-setosa":
                y.append(0)
            if spl[-1] == "Iris-versicolor":
                y.append(1)
            if spl[-1] == "Iris-virginica":
                y.append(2)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=12, stratify=y)
x = np.array(x).T
y = np.array(y)
x = np.vstack((np.ones((1, x.shape[1])), x))
w = logistic_softmax_regression(x, y, np.zeros((x.shape[0], len(np.unique(y)))),
alpha=0.1)
y_train_pred = np.argmax(softmax(np.dot(w[-1].T, x)), axis=0)
y_train_pred
y_test_pred = np.argmax(softmax(np.dot(w[-1].T, test)), axis=0)
# print(y_test_pred)
for i in range(len(y_test_pred)):
    if y_test_pred[i] == 0:
        print('Iris-setosa')
    elif y_test_pred[i] == 1:
        print('Iris-versicolor')
    else:
        print('Iris-virginica')
```

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

1. Đọc X và Y trong file input.csv
2. One hot encoder Y bằng hàm convert_labels
3. Đưa vào class LogisticReg với learning rate là 0.0001, số iteration là 500, số lượng class là 2 (đãđược tính toán trước đó (num_class = len(np.unique(Y)), hàm activation là sigmoid.
4. Khởi tạo số lượng trọng số W bằng với số lượng đặc trưng X với giá trị random
5. Chạy vòng lặp, tính Y dự đoán và đạo hàm của Y dự đoán, từ đạo hàm ta cập nhật trọng số W
6. Cuối cùng, kết quả đầu vào thuộc nhãn nào bằng cách lấy argmax của Y dự đoán được

tính bằnggiá trị trọng số cuối cùng (chiều đầu ra sẽ bằng số lượng class).

7. Độ chính xác sẽ được tính bằng hàm accuracy

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 numpy as np
import pandas as pd
from numpy import genfromtxt
from sklearn.model_selection import train_test_split
def softmax(z):
 e_z = np.exp(z - np.max(z, axis=0))
 return e_z / np.sum(e_z, axis=0)
def logistic_softmax_regression(X, y, w_init, alpha, tol=1e-4, loop=10000):
 w = [w_init]
 N = X.shape[1]
 d = X.shape[0]
 K = len(np.unique(y))
 count = 0
 check_w = 20
 while count < loop:
 mix_id = np.random.permutation(N)
 for i in mix_id:
 xi = X[:, i].reshape(d, 1)
 yi = np.eye(K)[:, y[i]].reshape(K, 1)
 zi = softmax(np.dot(w[-1].T, xi))
 w_new = w[-1] + alpha * np.dot(xi, (yi - zi).T)
 count += 1
 if count % check_w == 0:
 if np.linalg.norm(w_new - w[-check_w]) < tol:
 return w
 w.append(w_new)
 return w
test = []
with open("output.csv") as f:
 for val in f.readlines():
 l = val.strip()
 spl = l.split(",")
 if len(spl) > 1:
 test.append([float(i) for i in spl[:]])
test = np.array(test).T
test = np.vstack((np.ones((1, test.shape[1])), test))
x = []
y = []
with open("input.csv") as f:
 for val in f.readlines():
 l = val.strip()
 spl = l.split(",")
 if len(spl) > 1:
```

```
 x.append([float(i) for i in spl[:-1]])
 if spl[-1] == "Iris-setosa":
 y.append(0)
 if spl[-1] == "Iris-versicolor":
 y.append(1)
 if spl[-1] == "Iris-virginica":
 y.append(2)
X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.3,
random_state=12, stratify=y)
x = np.array(x).T
y = np.array(y)
x = np.vstack((np.ones((1, x.shape[1])), x))
w = logistic_softmax_regression(x, y, np.zeros((x.shape[0], len(np.unique(y)))),
alpha=0.1)
y_train_pred = np.argmax(softmax(np.dot(w[-1].T, x)), axis=0)
y_train_pred
y_test_pred = np.argmax(softmax(np.dot(w[-1].T, test)), axis=0)
# print(y_test_pred)
for i in range(len(y_test_pred)):
 if y_test_pred[i] == 0:
 print('Iris-setosa')
 elif y_test_pred[i] == 1:
 print('Iris-versicolor')
 else:
 print('Iris-virginica')
```

 # 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-virginica
Iris-virginica
Iris-virginica
Iris-virginica
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-versicolor
Iris-virginica
Iris-virginica
Iris-virginica
```

Câu 3 (3 *điểm*): Cho tập dữ liệu input.csv với 90 mẫu dữ liệu như câu 2, 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
from sklearn.metrics import accuracy_score, adjusted_rand_score
from sklearn.preprocessing import LabelEncoder


def initialize_K_centroids(X, K):
    m, n = X.shape
    np.random.seed(123)
    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):
        distance = np.linalg.norm(X[i] - centroids, axis=1)
        c[i] = np.argmin(distance)
    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, max_iters=10):
    _, n = X.shape
    centroids = initialize_K_centroids(X, K)
    # centroid_history = np.zeros((max_iters, K, n))
    for i in range(max_iters):
        idx = find_closest_centroids(X, centroids)
        centroids = compute_means(X, idx, K)
    return centroids, idx


df = pd.read_csv('input.csv', header=None)
X = np.array(df[[0, 1, 2, 3]])
K = 3
centroids, idx = find_k_means(X, K, max_iters=10)
label_mapping = {"Iris-setosa": 2, "Iris-versicolor": 0, "Iris-virginica": 1}
labels_true = np.array(df[4].map(label_mapping))
print(centroids)
```

```
print(np.asarray(idx, int))
# Đánh giá độ chính xác bằng accuracy
accuracy = accuracy_score(labels_true, idx)
print("Accuracy:", accuracy)
```

b) *(2 điểm)* Nếu sử dụng thuật toán *k*-means với k = 3 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):

```
[[5.92368421 2.77894737 4.40789474 1.43157895]
 [6.83636364 3.09090909 5.67727273 2.08181818]
 [4.98666667 3.37666667 1.48333333 0.25      ]]
```

2. Tỷ lệ phân cụm đúng (kết quả %): 88%

3. Tiêu chí đánh giá việc phân cụm (viết bằng lời)
   1. Đầu tiên gán các label true thành 0, 1, 2 để tiến hành dự đoán với các cụm mà ta tìm được, sau đó ta tính độ chính xác accuracy giữa giá trị dự đoán và nhãn đúng
   2. Gán nhãn như sau: {"Iris-setosa": 2, "Iris-versicolor": 0, "Iris-virginica": 1}
   3. Accuracy là số nhãn dự đoán đúng trên tổng số nhãn dự đoán.

Đà Nẵng, ngày 14 tháng 12 năm 2023

GIẢNG VIÊN BIÊN SOẠN ĐỀ THI

TRƯỞNG BỘ MÔN
(đã duyệt)