|  |
| --- |
| 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:**Võ Viết Trường **Lớp**:21TCLC\_Nhat2 **MSSV**:102210383

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 *ai* 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,…,*an* 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)  import heapq  def heuristic(*state*, *target*):      return abs(sum(*state*) - *target*)  def a\_star(*n*, *M*, *capacities*):      start = tuple([0]\**n*)      goal = *M*      open\_list = []      heapq.heappush(open\_list, (0 + heuristic(start, goal), 0, "", start))      closed\_list = set()      while open\_list:          \_, g, operations, state = heapq.heappop(open\_list)          if sum(state) == goal:              return operations          for i in range(*n*):              for j in range(*n*):                  if i != j:                      transfer = min(state[i], *capacities*[j] - state[j])                      if transfer > 0:                          new\_state = list(state)                          new\_state[i] -= transfer                          new\_state[j] += transfer                          new\_state = tuple(new\_state)                          if new\_state not in closed\_list:                              operation = f"Chuyển/Múc {transfer} lít nước từ gáo {i+1} qua gáo {j+1}"                              heapq.heappush(open\_list, (g + 1 + heuristic(new\_state, goal), g + 1, operations + "\n" + operation, new\_state))                              closed\_list.add(new\_state)          for i in range(*n*):              if state[i] < *capacities*[i]:                  new\_state = list(state)                  new\_state[i] = *capacities*[i]                  new\_state = tuple(new\_state)                  if new\_state not in closed\_list:                      operation = f"Chuyển/Múc {*capacities*[i]} lít nước từ bờ sông qua gáo {i+1}"                      heapq.heappush(open\_list, (g + 1 + heuristic(new\_state, goal), g + 1, operations + "\n" + operation, new\_state))                      closed\_list.add(new\_state)              if state[i] > 0:                  new\_state = list(state)                  new\_state[i] = 0                  new\_state = tuple(new\_state)                  if new\_state not in closed\_list:                      operation = f"Vứt bỏ {state[i]} lít nước từ gáo {i+1}"                      heapq.heappush(open\_list, (g + 1 + heuristic(new\_state, goal), g + 1, operations + "\n" + operation, new\_state))                      closed\_list.add(new\_state)      return "Không có đáp án"  print(a\_star(3, 15, [7, 8, 9]))  **# Trả lời:** Dán kết quả thực thi với dữ liệu Nhập: “3 15 7 8 9” vào bên dưới (1 điểm)    **# 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 heuristic (hàm h') được sử dụng để ước lượng khoảng cách từ trạng  thái hiện tại đến trạng thái mục tiêu. Hàm heuristic đóng vai trò quan trọng trong việc xác định  chi phí ưu tiên khi tìm kiếm đường đi tốt nhất trong không gian trạng thái.  - Hàm heuristic thường được thiết kế để đưa ra một ước lượng tối ưu về khoảng cách từ trạng thái  hiện tại đến trạng thái mục tiêu mà không yêu cầu tính toán chi tiết tất cả các bước trung gian - Trong bài toán này, hàm h' được thiết kế để đánh giá sự khác biệt giữa tổng lượng nước hiện tại trong các gáo và lượng nước mục tiêu cần đạt. Hàm h(state, target) trả về giá trị tuyến tính của sự chênh lệch giữa tổng lượng nước hiện tại trong các gáo và lượng nước mục tiêu. Ví dụ, nếu trạng thái hiện tại là (5, 3, 2) và mục tiêu là 13, thì hàm h sẽ trả về giá trị 3, thể hiện khoảng cách giữa trạng thái hiện tại và trạng thái mục tiêu. Hàm h' không xem xét chi phí di chuyển cụ thể giữa các trạng thái, nhưng nó giúp hướng dẫn thuật toán A\* tìm kiếm theo hướng có thể dẫn đến lời giải tối ưu trong thời gian ngắn nhất. |

***Câu 2*** (*4 điểm*): Cho tập dữ liệu [input.csv](https://drive.google.com/file/d/1BFZOngKU3CJrFu16MQWwDr1jfkXJMGYh/view?usp=sharing) 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.

1. *(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.  Cách thức hoạt động của mô hình Logistic Regression with Softmax như sau:  - Tính tổng có trọng số của đặc trưng đầu vào với ma trận trọng số w.  - Áp dụng hàm Softmax để tính toán xác suất dự đoán cho mỗi lớp.  - Dựa vào xác suất dự đoán, lớp với xác suất cao nhất được chọn là lớp dự đoán cho mẫu dữ liệu  đó.  Mô hình Logistic Regression with Softmax sử dụng hàm Softmax để tính toán các xác suất cho các lớp  khác nhau. Hàm Softmax được sử dụng để chuyển đổi giá trị đầu ra của mô hình thành các xác suất  tổng quát mà tổng các xác suất đó bằng 1. Điều này cho phép chúng ta dễ dàng xác định lớp có xác suất  dự đoán cao nhất. |

1. *(1 điểm)* Hãy thực thi chương trình và cho biết nhãn của 60 mẫu dữ liệu trong [output.csv](https://drive.google.com/file/d/1rloJhWAzl8r3weUKmUKqgBMFtkHYW3WR/view?usp=sharing)

|  |
| --- |
| **# 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-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-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica  Iris-virginica |

***Câu 3*** (3 *điểm*): Cho tập dữ liệu [Contries.csv](https://drive.google.com/file/d/1LdYTJ2aSl6y3zveaUArq7bCfEtwKtokt/view?usp=sharing), Hãy viết chương trình phân cụm bằng thuật toán *k*-means

1. *(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  import matplotlib.pyplot as plt  from sklearn.metrics import silhouette\_score  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*, *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  data = pd.read\_csv(r'D:\HocTap\Hocky5\TTNT\ThiCK\Countries*.*csv')  X = data.iloc[:, 1:3].values  K = 5  centroids, idx = find\_k\_means(X, K, *max\_iters*=10)  print(centroids)  data\_with\_clusters = data.copy()  data\_with\_clusters['Clusters'] = idx  pd.set\_option('display.max\_rows', None)  pd.set\_option('display.max\_columns', None)  print(data\_with\_clusters)  silhouette\_avg = silhouette\_score(X, idx)  print("Silhouette Score:", silhouette\_avg) |

1. *(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 5 cụm):    2. Tỷ lệ phân cụm đúng (kết quả %):    Tỉ lệ phần trăm: 44.46%  3. Tiêu chí đánh giá việc phân cụm (viết bằng lời):  - Đo độ tương phản của mỗi cụm có phân tách tốt hay không  -Một Silhouette Score cao (gần 1) cho thấy mỗi điểm dữ liệu trong một cụm gần các điểm dữ liệu khác trong cùng cụm và xa các điểm dữ liệu trong các cụm khác. |

Đà Nẵng, ngày 14 tháng 12 năm 2023

|  |  |
| --- | --- |
| **GIẢNG VIÊN BIÊN SOẠN ĐỀ THI** | **TRƯỞNG BỘ MÔN** |
|  | (đã duyệt) |