

Môn: **Nhập môn Trí tuệ nhân tạo**

Nhóm: **6**

Giảng viên: **Nguyễn Thành An**

Mã nhóm: **PBL**

#	Họ và tên	MSSV	Email
1	Võ Phước Thịnh	52000807	
2	Nguyễn Trung Nghĩa	52000693	nguyentrungnghiacmp528@gmail.com
3	Hồ Thịnh Phát	51900821	hothinhphatvd4@gmail.com
4	Trần Đình Phúc	52000109	phuctran12421@gmail.com

BẢNG PHÂN CÔNG CÔNG VIỆC

#	Công việc	Người thực hiện	Đánh giá
1	Câu 1	Nguyễn Trung Nghĩa	Hoàn thành
2	Câu 2	Trần Đình Phúc	Hoàn thành
3	Câu 3	Nguyễn Trung Nghĩa	Hoàn thành
4	Câu 4	Hồ Thịnh Phát	Hoàn thành
5	Câu 5	Hồ Thịnh Phát	Hoàn thành
6	Kiểm thử, chỉnh sửa lại	Trần Đình Phúc	Hoàn thành

#	Họ và tên	MSSV	% Đánh giá
1	Võ Phước Thịnh	52000807	0
2	Nguyễn Trung Nghĩa	52000693	100
3	Hồ Thịnh Phát	51900821	90
4	Trần Đình Phúc	52000109	100

Workspace: <https://github.com/trngbro/AI> (Private during deadline time)

Backtracking search

```
function BACKTRACKING-SEARCH(csp) returns a solution, or failure
  return RECURSIVE-BACKTRACKING({}, csp)
function RECURSIVE-BACKTRACKING(assignment, csp) returns a solution, or failure
  if assignment is complete then return assignment
  var ← SELECT-UNASSIGNED-VARIABLE(Variables[csp], assignment, csp)
  for each value in ORDER-DOMAIN-VALUES(var, assignment, csp) do
    if value is consistent with assignment according to Constraints[csp] then
      add { var = value } to assignment
      result ← RECURSIVE-BACKTRACKING(assignment, csp)
      if result ≠ failure then return result
      remove { var = value } from assignment
  return failure
```

Tham khảo giải thuật Backtracking trong slide bài giảng lesson 05

Câu 1 (2.0 điểm): Constraint Satisfaction


YC1.1

EightQueenSolver
+ board: [8x8]
+ solve(): str + backtracking(col: int): boolean + isValid(row: int, col:int): boolean + printBoard(): str

- Khởi tạo bàn cờ 8x8 trống
- ***solve()*** sẽ gọi đệ quy backtracking(số cột sẽ tăng dần)
- ***backtracking()*** được đệ quy tối đa 8 lần để điền 8 quân hậu vào bàn cờ
- ***isValid()*** kiểm tra tính hợp lệ của quân hậu khi điền vào với trạng thái bàn cờ hiện tại
- ***printBoard()*** từ danh sách bàn cờ sẽ in ra cho người dùng xem

Câu 1 (2.0 điểm): Constraint Satisfaction

YC1.2

 NQueenSolver
+ board: [NxN] + N: int
+ solve(): str + backtracking(col: int): boolean + isValid(row: int, col: int): boolean + printBoard(): str

- Khởi tạo bàn cờ với NxN ô trống
- ***solve()*** sẽ gọi đệ quy backtracking(số cột sẽ tăng dần)
- ***backtracking()*** được đệ quy tối đa 8 lần để điền 8 quân hậu vào bàn cờ
- ***isValid()*** kiểm tra tính hợp lệ của quân hậu khi điền vào với trạng thái bàn cờ hiện tại
- ***printBoard()*** từ danh sách bàn cờ sẽ in ra cho người dùng xem

Câu 1 (2.0 điểm): Constraint Satisfaction

Đánh giá hoàn thành

Tiêu chí	Điểm	Đánh giá
YC1-1	---	Hoàn thành
YC1-2	---	Hoàn thành
Tổng điểm đánh giá		2 điểm

YC2.1

1. Mở với tên tệp đã cho để đọc.
2. Đọc dòng đầu tiên của tệp và gán giá trị cho `e` và `l` tương ứng.
3. Khởi tạo một từ điển trống có tên là `nodes_dict`.
4. Đối với tôi trong phạm vi (`e`), hãy làm như sau:
 - a. Đọc dòng tiếp theo của tệp và chia thành `a` và `b`.
 - b. Nếu `a` chưa phải là khóa trong `nodes_dict`, hãy thêm nó làm khóa mới với giá trị `Node(a)`.
 - c. Nếu `b` chưa phải là khóa trong `nodes_dict`, hãy thêm nó làm khóa mới với giá trị `Node(b)`.
 - d. Nối giá trị của `nodes_dict[b]` vào danh sách kế vị của `nodes_dict[a]`.
5. Đối với tôi trong phạm vi (`l`), hãy làm như sau:
 - a. Đọc dòng tiếp theo của tệp và chia nó thành nút và giá trị.
 - b. Đặt giá trị của `nodes_dict[node]` thành `int(value)`.
6. Đặt `self.root` thành `nodes_dict["n00"]`.
7. Đặt `self.terminalStates` thành một cách hiểu từ điển để lọc ra các nút có người kế vị và tạo một từ điển ánh xạ số nhận dạng của chúng với giá trị của chúng.
8. Đặt `self.successors` thành một cách hiểu từ điển để ánh xạ từng nút trong `nodes_dict` vào danh sách kế vị của nó.

YC2.2

```
def print_tree(self, node, level=0):  
    print("\t" * level, node)  
    for succ in node.successors:  
        self.print_tree(succ, level+1)
```

Cài đặt hàm print()

YC2.3

Minimax Algorithm

```
function MINIMAX-DECISION(state) returns an action  
   $v \leftarrow \text{MAX-VALUE}(\textit{state})$   
  return the action in SUCCESSORS(state) with value v
```

```
function MAX-VALUE(state) returns a utility value  
  if TERMINAL-TEST(state) then return UTILITY(state)  
   $v \leftarrow -\infty$   
  for a, s in SUCCESSORS(state) do  
     $v \leftarrow \text{MAX}(v, \text{MIN-VALUE}(s))$   
  return v
```

```
function MIN-VALUE(state) returns a utility value  
  if TERMINAL-TEST(state) then return UTILITY(state)  
   $v \leftarrow \infty$   
  for a, s in SUCCESSORS(state) do  
     $v \leftarrow \text{MIN}(v, \text{MAX-VALUE}(s))$   
  return v
```


Cài đặt hàm run()

Câu 2 (2.0 điểm): Adversarial Search

Đánh giá hoàn thành


Tiêu chí	Điểm	Đánh giá
YC2-1	---	Hoàn thành
YC2-2	---	Hoàn thành
YC2-3	---	Hoàn thành
Tổng điểm đánh giá		2 điểm

YC3.1

 EightQueenSolver
+ board: [8x8] + solver: Glucose3()
+ add_clause(clause) + solve()

Cài đặt EightQueenSolver bằng Glucose3 với CNF

YC3.2

 NQueenSolver
+ N: int + board: [NxN] + solver: Glucose3()
+ add_clause(clause) + solve()

Cài đặt NQueenSolver bằng Glucose3 với CNF

Câu 3 (2.0 điểm): Logical Agents

Đánh giá hoàn thành

Tiêu chí	Điểm	Đánh giá
YC3-1	---	Hoàn thành
YC3-2	---	Hoàn thành
Tổng điểm đánh giá		2 điểm

YC4.1

Decision Tree

```
# Flatten data
train_X = train_X.reshape((train_X.shape[0], -1))
test_X = test_X.reshape((test_X.shape[0], -1))

# Học dữ liệu trên tập training
dt_clf = DecisionTreeClassifier()
dt_clf.fit(train_X, train_y)

# Tính độ chính xác trên tập training và tập test
train_y_pred = dt_clf.predict(train_X)
test_y_pred = dt_clf.predict(test_X)

train_acc_dt = accuracy_score(train_y, train_y_pred)
test_acc_dt = accuracy_score(test_y, test_y_pred)

print(f"Độ chính xác trên tập train: {train_acc_dt}")
print(f"Độ chính xác trên tập test: {test_acc_dt}")

# Lưu mô hình xuống tập tin
joblib.dump(dt_clf, 'dt_clf.joblib')

# Load mô hình lên từ tập tin
dt_clf = joblib.load('dt_clf.joblib')

# Chạy inference (tính prediction) cho ít nhất 05 input samples.
samples = test_X[:5]
predictions = dt_clf.predict(samples)
print(f"Kết quả tính prediction cho 5 input: {predictions}")

# Trực quan hoá cây với hàm tree.plot_tree().
plt.figure(figsize=(20,10))
plot_tree(dt_clf, filled=True)

print("\n\n\n\nTrực quan hoá cây:")
plt.show()
```

Sử dụng

```
from sklearn.tree import DecisionTreeClassifier, plot_tree
```

YC4.2

Naïve Bayes classifier

```
# Học dữ liệu trên tập training
nb_clf = GaussianNB()
nb_clf.fit(train_X, train_y)

# Tính độ chính xác trên tập training và tập test
train_y_pred = nb_clf.predict(train_X)
test_y_pred = nb_clf.predict(test_X)

train_acc_nb = accuracy_score(train_y, train_y_pred)
test_acc_nb = accuracy_score(test_y, test_y_pred)

print(f"Độ chính xác trên tập train: {train_acc_nb}")
print(f"Độ chính xác trên tập test: {test_acc_nb}")

# Lưu mô hình xuống tập tin
joblib.dump(nb_clf, 'nb_clf.joblib')

# Load mô hình lên từ tập tin
nb_clf = joblib.load('nb_clf.joblib')

# Chạy inference (tính prediction) cho ít nhất 05 input samples.
samples = test_X[:5]
predictions = nb_clf.predict(samples)
```

Sử dụng

```
from sklearn.naive_bayes import GaussianNB
```


YC4.3

KNeighborsClassifier

```
# Flatten data
train_X = train_X.reshape(train_X.shape[0], -1)
test_X = test_X.reshape(test_X.shape[0], -1)

# Học dữ liệu trên tập training với k=5
knn = KNeighborsClassifier(n_neighbors=5)

# huấn luyện mô hình
knn.fit(train_X, train_y)

# dựa đoán nhãn
train_y_pred = knn.predict(train_X)
test_y_pred = knn.predict(test_X)

# Tính độ chính xác trên tập training và tập test
train_acc_knn = accuracy_score(train_y, train_y_pred)
test_acc_knn = accuracy_score(test_y, test_y_pred)

print("Độ chính xác trên tập train:", train_acc_knn)
print("Độ chính xác trên tập test: ", test_acc_knn)

# Lưu mô hình xuống tập tin
dump(knn, 'knn_model.sav')

# Load mô hình lên từ tập tin
loaded_knn = load('knn_model.sav')

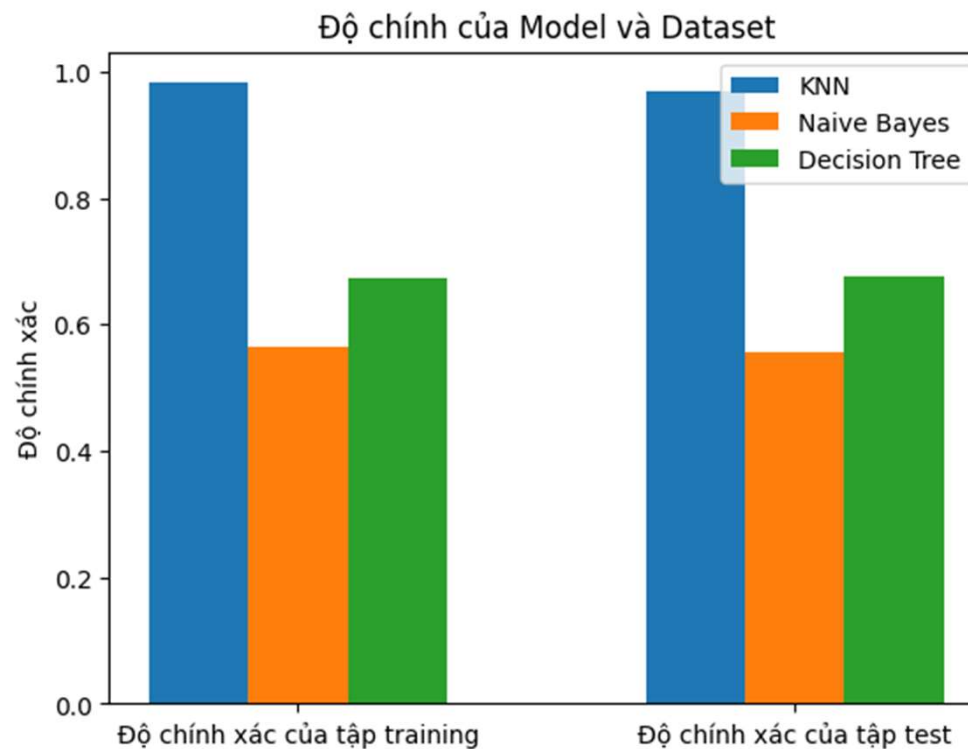
# Chạy inference (tính prediction) cho ít nhất 05 input samples.
random_test_samples = np.random.choice(test_X.shape[0], 5)
```

Sử dụng

```
from sklearn.neighbors import KNeighborsClassifier
```

Câu 4 (2.0 điểm): Machine Learning

YC4.4



Thu thập các training và test accuracy ta làm được biểu đồ độ chính xác này

Câu 4 (2.0 điểm): Machine Learning

Đánh giá hoàn thành

Tiêu chí	Điểm	Đánh giá
YC4-1	---	Hoàn thành
YC4-2	---	Hoàn thành
YC4-3	---	Hoàn thành
YC4-4	---	Hoàn thành
Tổng điểm đánh giá		2 điểm

Đánh giá tổng quát

Tiêu chí	Điểm	Đạt được
Câu 1	2 điểm	2 điểm
Câu 2	2 điểm	2 điểm
Câu 3	2 điểm	2 điểm
Câu 4	2 điểm	2 điểm
Câu 5	2 điểm	---
Tổng		---

Thuận lợi

- Đa số các thành viên làm việc ăn ý
- Đa số các thành viên có kiến thức về Git
- Đa số thành viên nhiệt tình tham gia tìm hiểu đề tài
- Vấn đề nghiên cứu được phân chia bài bản
- Các thành viên có nền tảng lập trình tốt

Khó khăn

- Số lượng thành viên nghiên cứu thực tế tương đối ít
- Mất nhiều thời gian để ôn tập lại python do các thành viên không xuất phát điểm từ các ngành khoa học máy tính
- Còn nhiều thiếu sót về phân tích đề
- Lượng công việc cho từng thành viên khá lớn
- Dù khó khăn nhưng nhóm vẫn hoàn thành toàn bộ bài tập

References

Book

- [1] *Stuart Russell, Peter Norvig. Artificial Intelligence : A Modern Approach, 2nd edition. New Jersey, McGraw-Hill, 2005*
- [2] Tom Mitchell. Machine Learning. New York, McGraw-Hill, 1997.
- [3] Manning and Schuetze, Foundations of Statistical Natural Language Processing, MIT Press. Cambridge, MA, 1999.

Website

- [1] sakai.it.tdt.edu.vn