Basic Concepts of Artificial Intelligence

Fundamentals of AI – Lecture 1

Part 1: Introduction to Course and Basic AI Concepts

AI in Society

- AI, in one form or another, has captured the imagination of society at large
- Different points of view exist about the contribution that AI makes to society
- On the one extreme a segment of society views it as science that will change almost every aspect of our lives
- Certainly, if we take into account technologies that make use of AI this point of view has great truth to it.

COVID-19 Guidelines

- Wear a mask in the class and building
  - Per UNT president's request
- UNT guidelines for COVID-19
  - https://healthalerts.unt.edu/

Syllabus

- Syllabus for CSCE 5210 Section 001 - Fundamentals of Artificial Intelligence (Fall 2021 1) (instructure.com)

Python Programming

- This course does *not* teach Python programming
- We will have a quick discussion on the basics of Python
- The best way of installing Python is through Anaconda:

- A comprehensive tutorial on Python is available from:

  Python Tutorial (w3schools.com)

Academic Offences

- Academic Offences include, but are not limited to:
  - **Cheating**
  - **Plagiarism**
  - **Misrepresentations**

- All students must read 06.003 Student Academic Integrity
  - https://policy.unt.edu/policy/06-003

Plagiarism

- All assignments and projects must be done individually:
  - You may not copy code directly from any other source
  - Plagiarism detection software will be used on all of the assignments
  - If you viewed code (from books, lecture notes or the Internet), you must include a reference in your project
  - You may not share code with any other students by transmitting completed functions to your peers
    - This restriction includes—but is not limited to—electronic and hard-copy sharing

Plagiarism

When one student copies from another student, both students are responsible

- Exceptions are made for outright theft

The penalty for plagiarism on an assignment is a mark of 0 on that assignment

All instances of plagiarism will be reported to UNT's Academic Integrity Office

Plagiarism

- The best way to avoid plagiarism is:
  - Make an early start on the projects
  - Understand the problem and the solution before you start coding

Seeking help

- Your first point of contact is any of the TAs.
- Typically, you will contact that for any issues on the Projects
- Any clarification on the lecture content should be directed to the course Instructor

AI Real World Applications

- One of the most commonly used AI apps is the electronic carburetor that automatically determines the *optimal* mix of fuel and air into a car's combustion chamber
- Another massively used app is the route finder that is used by virtually every driver to find "best" routes
- Robots that extract customer's order items from a warehouse and packs them for delivery
- Robots also perform many of the assembly tasks in a car manufacturing plant

About this course …..

- In this course we will cover some of the major algorithms that have contributed to its success
- AI is an umbrella term for any system that demonstrates a capability to take decisions and perform tasks that would be considered intelligent by humans
- Machine Learning is also a term used commonly in AI and the two areas both have the same fundamental goal but have differing approaches
- The two fields *complement* each other with no conflict between them

So, what distinguishes AI from Natural Intelligence?

- Two major differences:

1. The “skill” level – robot car assembly has high utility value but cannot be compared to a Michelangelo or an Einstein (not yet!)
2. AI systems do not have a fundamental understanding of their environment

Will AI ever reach human intelligence levels or even exceed them?

However….

- AI has achieved some stunning successes – refer to list given earlier
- In addition to those applications, it has made vast strides in *pattern recognition, image* and *audio recognition* and *natural language understanding*
- IBM's deep blue beat the reigning world champion Gary Kasparov for the world title
- More recently a deep learning program beat the world champion in the game Go
- Self driving cars have safely navigated freeways in the United States in full autonomous mode
- Image recognition rates on *benchmark image libraries* exceed human expert's performance

Video: Deep Blue

Deep Blue beats G. Kasparov (Chess contest) 1997 - Bing video

Video: Google Deep Mind vs Go Champion

AlphaGo and the future of Artificial Intelligence - BBC Newsnight - Bing video

Geoffrey Hinton on Deep Learning

A Fireside Chat with Turing Award Winner Geoffrey Hinton, Pioneer of Deep Learning (Google I/O'19) - Bing video

The road further ahead for AI

- In the immediate future – in a timeframe of 5 to 10 years from now, AI is going to be judged by some challenges that it has failed to meet up to now

1. It needs to learn how to safely navigate a car in crowded urban areas
2. Learn how to conduct an extended (not 1 minute!) intelligent conversation about everyday matters

    Alan: Siri, in the future can you call me an ambulance when I have severe difficulty in breathing

    Siri: Sure Alan, I'll call you ambulance from now on

Can AI ever exceed human intelligence?

- This is one of the most challenging questions of our times and there is no definitive answer
- What we can say with 100% certainty is that AI and human intelligence have very different modes of operation
- Humans are inherently creative and sometimes behave *irrationally* (i.e. go against basic logic reasoning) whereas AI is strictly *rational*
- Geniuses such as Einstein did not always behave rationally, his most creative work Relativity went against Newton's well established laws of dynamics

AI vs Humans

- In terms of intelligent behavior humans have some clear advantages over machines
  1. Human DNA is much more older than machine "DNA"
  2. Humans have the inherent ability to learn language
  3. Humans communicate with each other and such communications increase not just knowledge but also enhance learning
- So, is AI condemned to be a "*2nd class citizen*" to human intelligence?
- I firmly believe that the answer is ***No***, yet one needs to be realistic about AI's capabilities
- AI will outperform humans when the tasks it is assigned can be solved by rational thought – route finder problems, image recognition, … and so on.
- These are challenging, yet narrowly scoped problems – this is where AI will be ***unbeatable***

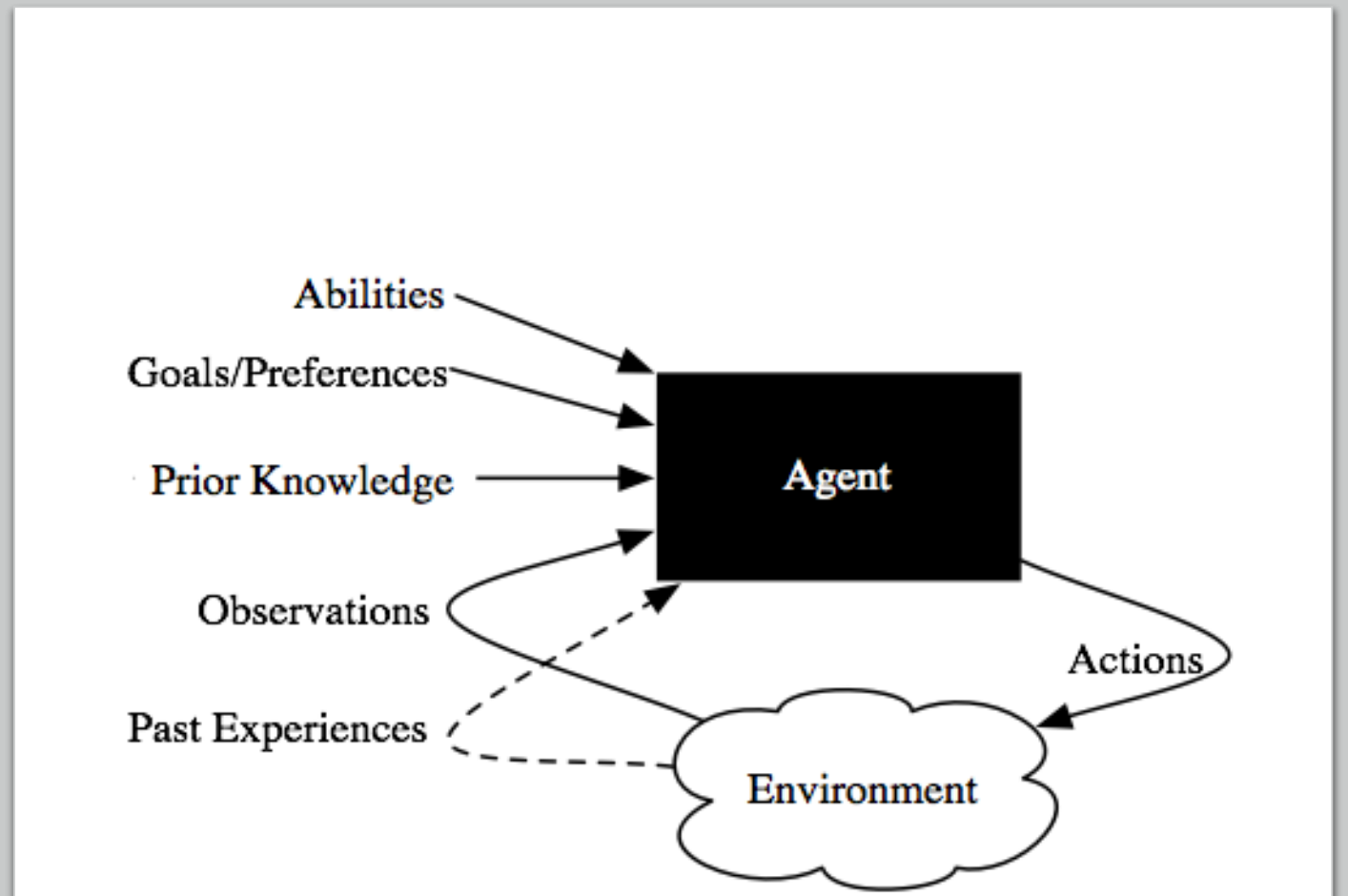
Rational Decision Making

- Rational decision making is done on the basis of optimizing an *utility function*
- Some examples of utility functions are:

1. Distance to target point from current location.
2. Score in a game
3. Correct recognition rate in an image recognition task
4. Prediction accuracy in weather forecasting (or any other type of forecasting)

Rational Decision Making

- In other words, AI excels when the problem is ***well defined***, ***narrow in scope*** and demands ***rational decision making***
- We can think of AI as an agent
- An agent interacts with its environment
- It uses sensors to perceive the environment and uses *actuators* such as robots, motors, conveyor belts and other mechanical devices that are driven by intelligent software



Paradigms of Artificial Intelligence

- Three main paradigms have emerged:

1. *Agent* based approaches based on rational decision making
2. *Learning* – system behavior is guided by using past data. This involves pattern recognition
3. *Reasoning under uncertainty* – this involves building *probabilistic models* that enable decision making

- This course will be structured around these 3 paradigms and will be delivered in three modules

Part 2: Python Programming Primer

Python

- Let’s get down to Python
- Some good sources of reference for Python are:
  - The Python Tutorial — Python 3.9.6 documentation
  - Python Tutorial (tutorialspoint.com)
  - Why Use Python for AI and Machine Learning? (steelkiwi.com)
- To get started with Python you first need to install the Python environment
- By far the easiest way to do this is by installing Anaconda which provides you with the latest version of Python and several powerful IDE’s such as Spyder, PyCharm and Jupyter notebook

Installing Python

- Download Anaconda from Anaconda | Individual Edition and accept the default option at each stage
- Select either Spyder or Pycharm as your IDE (we will use Jupyter notebook later on) and you should now be ready to start writing code in Python

Data Types in Python

- The basic data types are:
  1. int – stores 64-bit integers
     int num=23
  2. float – stores real numbers
     float num=12.7
  3. bool – stores Boolean values, either True or False
     bool sunny=True
- Can check for data type
  >>>type(5)
  int
  >>>type(2.8)
  float
- Can convert (cast) one type into another
  float(4) converts integer 4 to float 4.0
  int(7.9) truncates float 7.9 to integer 7

Arithmetic Operators in Python

- The basic operators are: +, -, * /, ** and % in most other languages
- a=2; b=8
- c=a*b; d=b/2
- e=a+b*c
- f=b**a
- g=b%2
- print(‘c=‘,c ,‘d=‘,d, ‘e=‘, e, ’f=‘, f, ‘g=‘,g)

Operator Precedence

- Arithmetic expressions are evaluated in the following order:
  1. Brackets ()
  2. Exponentiation **
  3. *, /, %
  4. +,-
- When operators of equal precedence appear they are executed in left to right order

  r=a*b/c
- In this case a is first multiplied by d and the result is then divided by c

Vector Operations in Python

- Python has direct support for vector operations

```
from numpy import array
# numpy is a library that provides support for fast operations on multidimensional arrays
v1=array([1,12,5])
v2=array([8,24,30])

v3=v1+v2  # vectors treated in the same way as ordinary (scalar) data types such as int or float
v4=v2-v1
v5=v1*v2
v6=v2/v1
print(‘Addition of vectors v1 and v2 is',v3)
print('Subtraction of vectors v1 and v2 is',v4)
print('Multiplication of vectors v1 and v2 is',v5)
print('Subtraction of vectors v1 and v2 is',v6)
```

String data type

- Text data is supported by the String data type

```
s1=‘The dog barked at the’
s2=‘the cat that was in the garden’
s3=s1+’ ‘+s2
s4='Hello'
len(s4)
s4[0:2] # string slicing (subsetting of a string)
s4[0:] # gets the first character of the string
s4[-1] # gets the last character
s4[-2] # gets the one before the last character
s4[::2] # gets every other character
```

Relational Operators

- Relational operators tests two variables against each other
- Suppose we have two variables v1 and v2 (can be int, float or string)
- All of the comparisons below will return either True or False

v1<v2

v1<=v2

v1>v2

v1>=v2

v1==v2

v1!=v2

Boolean Logic Operators

- Suppose that we have two Boolean variables b1 and b2

not a inverts the value of a; returns True if a is False, otherwise returns True

- The logical **and** operation **a and b** returns True if both a and b are true, otherwise it returns False
- The logical **or** operation **a or B** returns True if either a or B is True, otherwise it returns False
- The truth table below summarises the logic of these operators

a	b	a and b	a or b
True	True	True	True
True	False	False	True
False	True	False	False
False	False	False	False

Boolean Logic Examples

- Comparing houses according to price and number of bedrooms

house1_price=275000

house2_price=321000

house1_beds=3

house2_beds=2

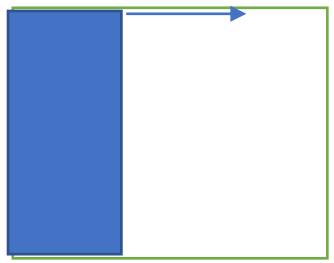
suitability=house1_price<house_price and house1_beds>=house2_beds

The Bool variable *suitability* returns True as both conditions are True, indicating that house 1 is a better buy if we are only interested in price and number of rooms

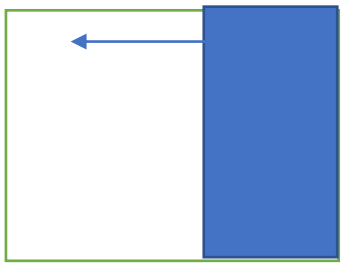
Branching in Python

- Decision making is a key part of any program and Python provides a simple method to implement it through its **if statement**
- At a corner:

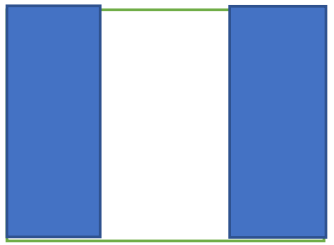
If left is blocked
go right



if right is blocked
go left



if both left and right are blocked
walk down



Branching in Python

- Python has 3 different versions of the **if** statement

```
If condition:
    action 1
    ……
    ……
    action n
```

```
If condition:
    action 1
    ……
    ……
    action n
else:
    action n+1
    ……..
    ……..
    action m
```

```
If condition:
    action 1
    ……
    ……
    action n
else:
    action n+1
    ……
    ……
    action m
elif:
    action m+1
    ……
    action o
```

Branching in Python

- Note the indentation of actions. This is required in Python. Without indentation the Python interpreter will signal a syntax error.
- Suppose that we wish to grade after a test

```
 if (marks>92):
   grade='A'
elif (marks>89):
    grade='A-'
 elif (marks>86):
     grade='B+'
 elif (marks >82):
      grade='B'
  elif(marks>79):
        grade='B-'
   elif(marks>76):
        grade='C+'
   elif(marks>72):
       grade='C'
  elif(marks>69) :
       grade='C-'
   else:
       grade='D'
```

Iteration

- In many situations we may need to execute a set of instructions multiple times – e.g., the grading process needs to be done for each student
- In these situations, Python provides two types of iterative statements, namely the **While** loop and the **For** loop.

```
while (condition)
        action 1
        …….
        action n
```

- The looping continues until the condition is false.

Populating/Initializing Arrays

- 1 D array:

```
arr_num = [0] * 10
print(arr_num)

arr_str = ['P'] * 10
print(arr_str)
```

- 2D array:

```
x = np.empty(shape=(10,10))
x.fill(var)
```

While Loop

- Suppose that we are searching for a given number in an array until either we find the number or run out of elements in the array

```
from numpy import array
num=array([15,6,71,87,19,2,56,43,97,101,4,62])
number = int(input("Enter a number:"))
index=0
found=False
while(not(found) and index<12):
    if(num[index]==number):
        found=True
    else:
        index=index+1 # advance the pointer by 1 to read the next element
if(index==12):
    print('Number not found in list')
else:
    print('Number found at index position',index)
```

For Loop

- The For loop iterator can be used with both integer and string values

```
weights=array[160,101,50,181,65]
sum=0
for i in range(0,5)
      sum=sum+weights[i]
print('Sum of weights is:',sum)

for c in s
   print(c)
```

Multi Dimensional Arrays

- Python provides support for n dimensional arrays
- A table of values can be conveniently represented by a 2D array

29	145	64	53
43	212	675	12
31	56	77	43

table=array([[29,145,64,53],[43,212,675,12],[31,56,77,43]])

- Just as with 1D arrays we can slice 2D arrays as well

```
table[0:1,1:2] produces [[145,64],[212,675]]
table[0:1,:] produces [[29,145,64,53],[43,212,675,12]]
table[:,1:2] produces [[145,64],[212,675],[56,77]]
table[:,:] produces the entire array
```

Lists in Python

- Python has another data structure that stores multiple items in the form of a **list**

```
L1=[car,truck,bicycle]
```

- Lists are used in situations where the number of items cannot be predicted in advance, unlike an array where you need to declare the number of items when you set it up
- This means that lists are more efficient than arrays in terms of memory – a list only uses as much memory as the number of items currently in it
- A list item can itself be a list

```
L2=[[car,van],[truck,trailer],[bicycle,scooter]]
```

Lists in Python

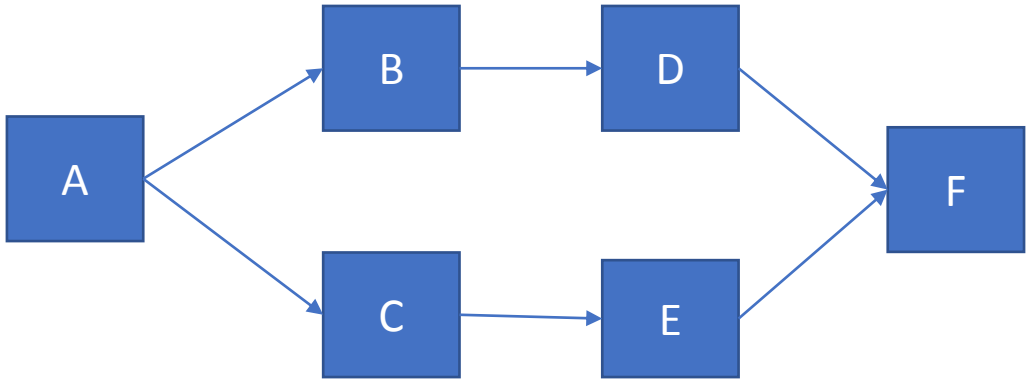
- Unlike arrays list items need not be homogeneous – i.e., items can be of different data types

```
L3=[12,'car',4.7]
```

- Common list operations include:
  1. Initialization `L=[]`
  2. Adding an item to a list at the end `L.append(item)`
  3. Removing an item from a list `L.remove(item)`
  4. Accessing an item `L  L[item]`

Using a list – a typical example

- Representing a graph, each node represent an object and each object is connected to a variable number of objects



- The data structure used is an *adjacency list* for each object. The list contains the objects that are reachable from any given object

Arrays versus Lists

Array	List
Storage space needs to be declared in advance – may waste memory	Storage is dynamic – only uses memory as needed
Arithmetic operations (*, /) can be performed on elements	No arithmetic operations allowed – will throw an error
Arrays need to be declared with an import statement to the numpy library	No declaration required as lists are built into Python

```
Import numpy as np
array = np.array([3, 6,
9, 12])
division = array/3
print(division)
print (type(division))
>>> [1. 2. 3. 4.]
```

```
list = [3, 6, 9, 12]
division = list/3
```

T>>>*TypeError Traceback (most recent call last) in () 1 list = [3, 6, 9, 12] ----> 2 division = list/3 TypeError: unsupported operand type(s) for /: 'list' and 'int'*