Short Course in Artificial Intelligence

October 2023

Prof. Bernard Merialdo

Professor Emeritus

Institut Eurecom, France

About myself

- Majoring in Maths
- PhD in Artificial Intelligence
- Worked in IBM on research for *Speech Recognition* and *Natural Language Processing*
- Then became Professor, working on *Image and Video analysis*
- Now Professor Emeritus consulting Vietnamese companies on AI

Short Course in Artificial Intelligence

October 2023

Lecture 1: Complexity

Why a course on AI ?

- ***"The A.I. Revolution Will Change Work. Nobody Agrees How."***
  - *New-York Times, June 2023*
- Recent improvements on Generative AI and LLM (ChatGPT, DALLE…)
- Some new tools will change the way to do Software Development

Why a course on AI ?

"Hollywood writers have been on strike for 100 days — and there's no end in sight" (8 August 2023)

5

Why a course on AI ?

"Hollywood writers have been on strike for 100 days — and there's no end in sight" (8 August 2023)

AI may create movie scenarios
AI may create videos of artificial characters

6

Why a course on AI ?

- **ChatGPT?:** *Write a python code to sort a list of 5 numbers without any loop or a predefined function.*

```
def sort_numbers(numbers):
    if numbers[0] > numbers[1]:
        numbers[0], numbers[1] = numbers[1], numbers[0]
        sort_numbers(numbers)
    if numbers[1] > numbers[2]:
        numbers[1], numbers[2] = numbers[2], numbers[1]
        sort_numbers(numbers)
    if numbers[2] > numbers[3]:
        numbers[2], numbers[3] = numbers[3], numbers[2]
        sort_numbers(numbers)
    if numbers[3] > numbers[4]:
        numbers[3], numbers[4] = numbers[4], numbers[3]
        sort_numbers(numbers)
    return numbers

if __name__ == "__main__":
    numbers = [5, 2, 9, 1, 7]
    sorted_numbers = sort_numbers(numbers.copy())
    print(sorted_numbers)
```


Why a course on AI ?

- Many jobs will be impacted by AI
- Software Development will be impacted too
- We don't know how much (yet)
  - This is currently happening
- Things will be changing quickly…
  - New skills such as prompt engineering

RECAP IN VIETNAMESE

Cách mạng “AI”

- Cuộc cách mạng AI có tiềm năng thay đổi công việc, nhưng không có sự đồng thuận về cách diễn biến cụ thể.
- Các tiến bộ gần đây trong Generative AI và LLM (ChatGPT, DALLE...) có khả năng thay đổi cách phát triển phần mềm.
- Cuộc đình công của nhà viết kịch Hollywood kéo dài 100 ngày và chưa có dấu hiệu kết thúc. AI có thể tạo ra kịch bản phim và video với nhân vật nhân tạo.

Cách mạng "AI"

- Nhiều công việc sẽ bị ảnh hưởng bởi AI, bao gồm cả phát triển phần mềm, nhưng mức độ tác động chưa được xác định.
- Sự thay đổi xảy ra nhanh chóng và yêu cầu các kỹ năng mới như kỹ thuật prompt.

About Computers

- The computer on which I learned programming

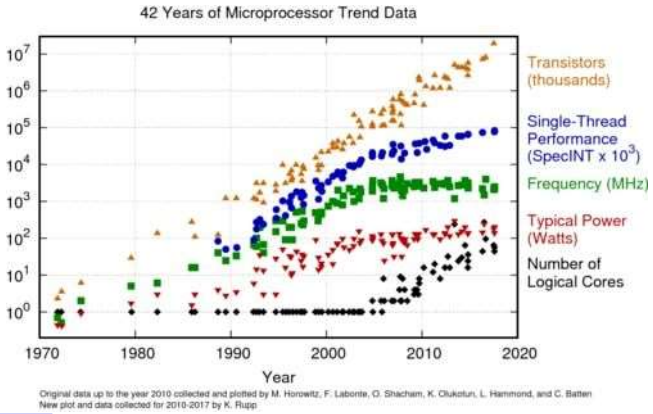
- 32 KB RAM, 5 MB hard disk

About Computers

- A Smartphone today:
  - 8 GB Ram
  - 256 GB storage
  - 2340x1080 display
  - 2.5 in x 5.7 in
  - cameras: 3 rear, 1 front
  - phone, wifi, bluetooth
  - GPS….

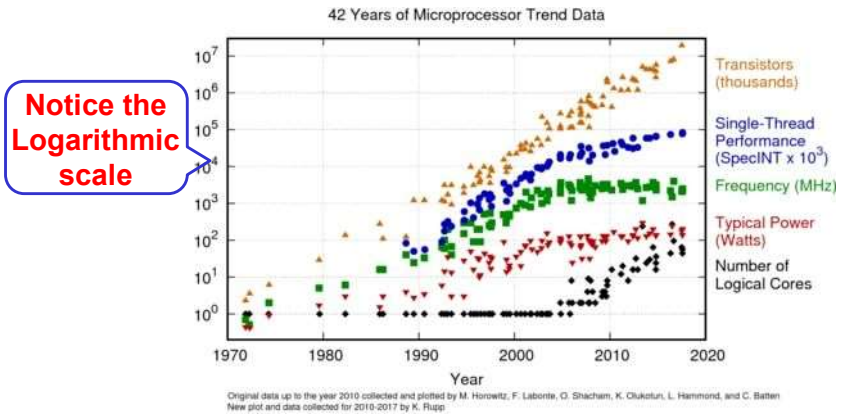
About Computers – Moore's law

- **"Every two years, the number of transistors on microchips will double"** (1965)



About Computers – Moore's law

- **"Every two years, the number of transistors on microchips will double"** (1965)



About Computers – Moore's law

- **"Every two years, the number of transistors on microchips will double"** (1965)

- Increase in computing speed
- Increase in memory storage
- Increase in disk storage
- Increase in communication
- Increase in number of users
- Reduction in cost

About Computers

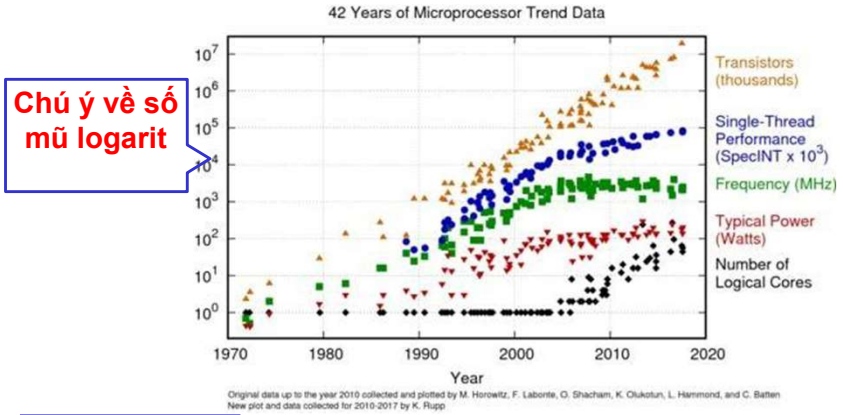
Why is it important ?

- When you learn Maths, Maths will be the same in 10 years
  - Once you know Maths, you are done
- When you learn English, English will be the same in 10 years
  - Once you know English, you are done
- When you learn computers, technologies in 10 years will be different
  - You will have to keep learning new technologies
  - You have to learn how to learn

RECAP IN VIETNAMESE

Định luận Moore

- **"Mỗi hai năm, số lượng transistor trên vi mạch sẽ tăng gấp đôi" (1965)**



Định luận Moore

- **"Mỗi hai năm, số lượng transistor trên vi mạch sẽ tăng gấp đôi" (1965)**

  - Sự tăng tốc độ tính toán.
  - Sự tăng cường khả năng lưu trữ bộ nhớ.
  - Sự tăng cường khả năng lưu trữ đĩa.
  - Sự tăng cường khả năng truyền thông.
  - Sự tăng số lượng người dùng.
  - Sự giảm chi phí.

Artificial Intelligence

- Can a machine act / think as a person ?
- A very old idea (dream): 1769: Von Kempelen

Artificial Intelligence

- Can a machine act / think as a person ?
- A very old idea: 1769: Von Kempelen

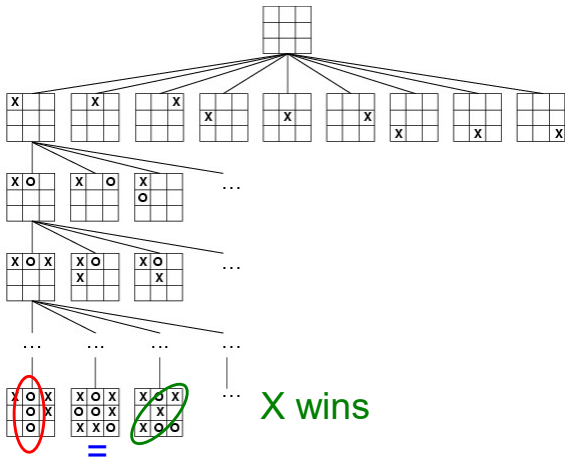
Artificial Intelligence: the Early Days

- The First Computer: ENIAC 1943
- First AI programs: 1950-

- "Solving complex problems"
  - complex = many possible solutions
  - problem = find the best solution
  - too many solutions to check them all → heuristics
- Typical Applications
  - Theorem Proving, Logic
  - Find best path in a tree/graph
  - Games

Artificial Intelligence

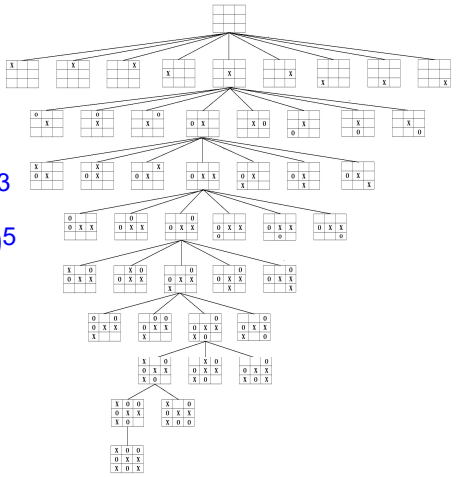
- Game complexity: Tic-tac-toe



Artificial Intelligence

- Game complexity: Tic-tac-toe (caro)

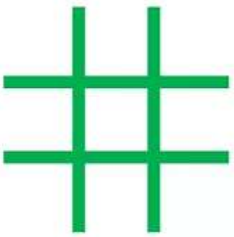
Board size: 9
Number of moves: ~ 4
Number of positions: ~ 10^3
Number of games : ~ 10^5



Artificial Intelligence

- Games and intelligence:
  - Games complexity:

TIC-TAC-TOE

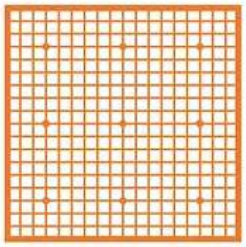


3x3

CHESS

8x8

GO



19x19

Artificial Intelligence

- Games and intelligence:
  - Games complexity:

Game	Board size	Number of moves (average)	Number of positions	Number of games	Average game length
Tic-tac-toe	9	4	10^3	10^5	9
Chess	64	35	10^{47}	10^{123}	80
Go (19x19)	361	250	10^{170}	10^{360}	150

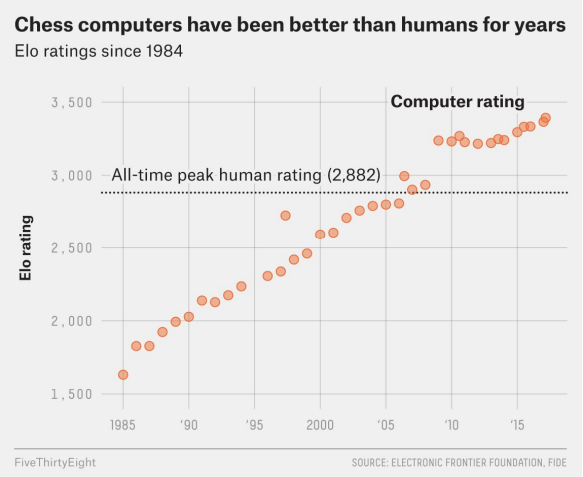
(Number of atoms in the Universe ≈ 10^{80})

Artificial Intelligence

- Chess was considered a very difficult game:
  - 1997: Kasparov vs DeepBlue

Artificial Intelligence

- Chess was considered a very difficult game:

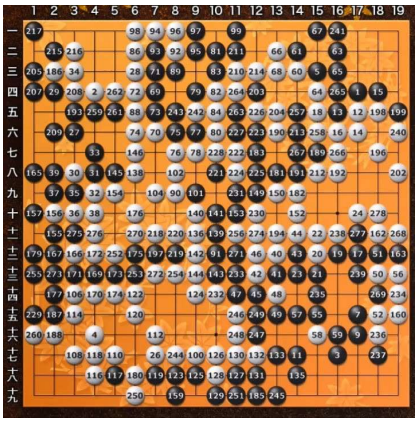


Artificial Intelligence

- The progress of Computer Chess were due to the increase in computer power, which allowed programs to enumerate more possibilities.
- Despite this increase, Go was considered infeasible because of the gap in complexity

Artificial Intelligence

- Go was considered infeasible but:
  - 2017 AlphaGo



Rank	Name	Elo
1	Google DeepMind AlphaGo	3608
2	Ke Jie	3608
3	Park Junghwan	3593
4	Lee Sedol	3550
5	Iyama Yuta	3536
6	Mi Yuting	3528
7	Shi Yue	3509
8	Kim Jiseok	3504
9	Lian Xiao	3504
10	Tuo Jiaxi	3501

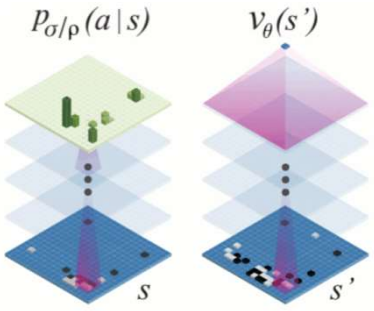
RECAP IN VIETNAMESE

Trí tuệ nhân tạo

- Máy tính đầu tiên: ENIAC (1943).
- Chương trình trí tuệ nhân tạo đầu tiên: từ năm 1950.
- Mục tiêu: Giải quyết vấn đề phức tạp.
- Ứng dụng phổ biến:
  - Logic và chứng minh định lý.
  - Tìm đường đi tốt nhất trong cây/đồ thị.
  - Các trò chơi phức tạp: Cờ vua, cờ vây..

AlphaGo

- Chess was solved because computers became faster
- Go was solved because of a new learning technique: Deep Learning
  - Policy (Deep) network
  - Value (Deep) network

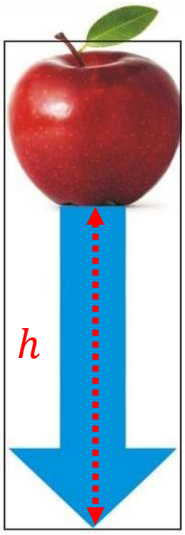


Artificial Intelligence - Machine Learning

- In traditional programming, the software developer describes each step of the processing
- In Machine Learning, the AI scientist provides a model and examples, then lets the machine find the best instance of the model (this is the training). When the model is trained, it can be used to process new data.

Artificial Intelligence - Machine Learning

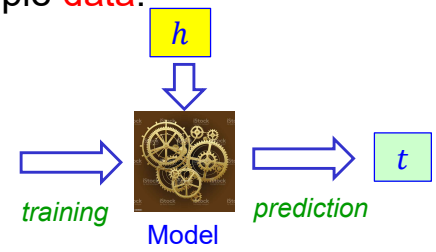
- Problems are usually solved by a formula: $h = \frac{1}{2}gt^2$ so $t = \sqrt{\frac{2h}{g}}$

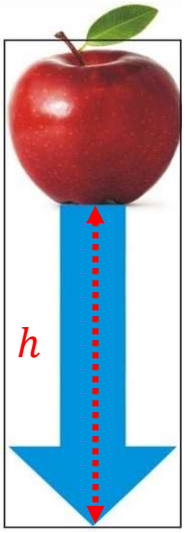


Artificial Intelligence - Machine Learning

- Problems are usually solved by a formula: $h = \frac{1}{2}gt^2$ so $t = \sqrt{\frac{2h}{g}}$
- Machine Learning builds a model from example data:

h	t
1m	0.45s
2m	0.64s
3m	0.78s
4m	0.90s
…	…





RECAP IN VIETNAMESE

Trí tuệ nhân tạo - Machine Learning

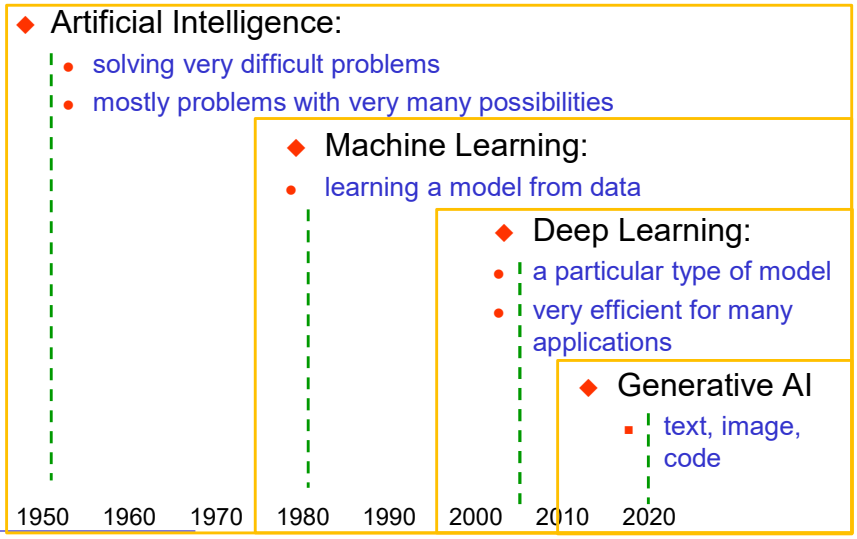
- Thông thường: Cần mô tả từng bước xử lý.
- Trong Machine Learning: chỉ cần cung cấp mô hình và ví dụ

⇨
- Máy tính sẽ tự tìm ra phiên bản tốt nhất của mô hình thông qua quá trình huấn luyện

Artificial Intelligence

- Why is AI so important today ?
- Because of this new technique called « Deep Learning », which is one instance of Machine Learning
- It has been shown to be VERY effective on many difficult problems:
  - Image recognition
  - Natural Language Processing
  - Data Analysis
- Recently, Generative AI allows to produce realistic content: text, images, code…

Artificial Intelligence



RECAP IN VIETNAMESE

Tại sao AI lại quan trọng?

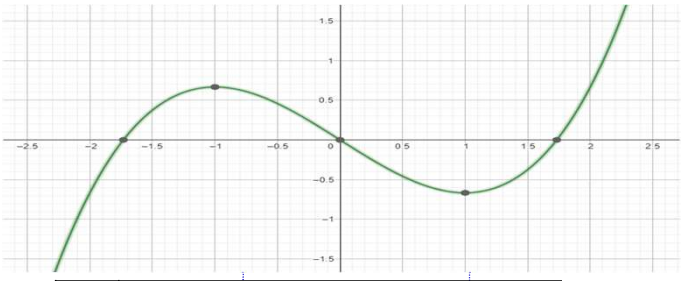
- Trí tuệ nhân tạo (AI):
  - Giải quyết các vấn đề rất khó khăn
  - Đặc biệt giải quyết các vấn đề có rất nhiều khả năng
- Học máy (Machine Learning):
  - Học một mô hình từ dữ liệu
- Deep Learning:
  - sử dụng một kiểu mô hình đặc biệt
  - rất hiệu quả trong nhiều ứng dụng khác nhau
- Trí tuệ nhân tạo sinh sản (Generative AI):
  - Tạo ra văn bản, hình ảnh, mã code

Maths for AI

- If you want to understand the techniques used in AI, it is useful to have some basic knowledge in Mathematics:
- Calculus
  - Derivatives, maximum and minimum
- Linear Algebra
  - Vectors, matrices, dimension
- Probability – Statistics
  - Probability distribution, conditional probabilities, mean, variance

Maths for AI - Calculus

- Remember derivatives $f'(x)$



x		-1		0		$+1$	
$f'(x)$	+	0	-		-	0	+
$f(x)$	↗		↘	0	↘		↗

local maximum *local minimum*

Maths for AI – Probabilities, Statistics

- Probability distribution:

$$P(100\$) = \frac{4}{18} = 0.222$$

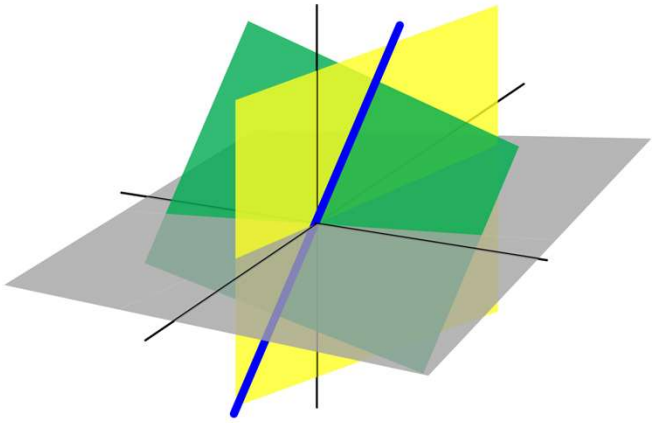
$$P(200\$) = \frac{2}{18} = 0.111$$

$$\dots$$

$$P(1000\$) = \frac{1}{18} = 0.055$$

Maths for AI - Linear Algebra

- Vectors, matrices and vector spaces



Toán trong AI

- Một số kiến thức toán được sử dụng trong AI:
- Calculus (Phép tính):
  - Đạo hàm, giá trị tối đa và tối thiểu.
- Linear Algebra (Đại số tuyến tính):
  - Vector, ma trận, chiều của ma trận.
- Probability - Statistics (Xác suất - Thống kê):
  - Phân phối xác suất, xác suất có điều kiện, trung bình, phương sai.

Practice 1

Practice 1: Tic-Tac-Toe

- We provide you with a Jupyter notebook which contains a program to play Tic-Tac-Toe
- You can modify the program by commenting/uncommenting some lines
- Initially, the program plays randomly, but it is able to learn how to play better
- You can experiment with the program, and later play against it.

Practice 1: Tic-Tac-Toe

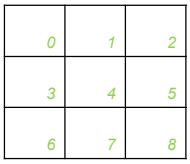
- Board 3x3
- Python representation:

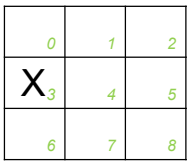
  board = '.....'
- Players X and O:

  players = ['X','O']

  player = 'X'
- Player X plays in slot 3:

  board = '...X.....'





Practice 1: Tic-Tac-Toe

- List empty slots:

```
def Empty_slots(board):
        return [x for x in range(len(board)) if board[x] == '.']
```

- Random Player:

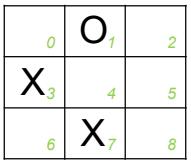
```
def Random_player(board):
        return random.choice(Empty_slots(board))
```

- Test if winning position:

```
def Win(board, player):
    for x in [[0,1,2],[3,4,5],[6,7,8],[0,3,6],[1,4,7],[2,5,8],[0,4,8],[2,4,6]]:
        if (board[x[0]]==player) and (board[x[1]]==player) and
(board[x[2]]==player):
            return True
    return False
```

Practice 1: Tic-Tac-Toe

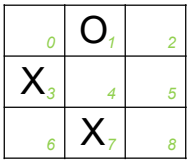
- List empty slots:




```
def Empty_slots(board):
        return [x for x in range(len(board)) if board[x] == '.']
        return [x for x in range(len(board)) if board[x] == '.']
                                        '.O.X...X.'
        return [x for x in range(len(board)) if board[x] == '.']
                                        9
        return [x for x in range(len(board)) if board[x] == '.']
                                [0, 1, 2, 3, 4, 5, 6, 7, 8]
        return [x for x in range(len(board)) if board[x] == '.']
                                    [0, 2, 3, 5, 6, 8]
```


Practice 1: Tic-Tac-Toe

- Random Player:




```
def Random_player(board):
        return random.choice(Empty_slots(board))
        return random.choice(Empty_slots(board))
                                        '.O.X...X.'
        return random.choice(Empty_slots(board))
                                    [0, 2, 4, 5, 6, 8]
        return random.choice(Empty_slots(board))
```

for example 2

Practice 1: Tic-Tac-Toe

- Playing:

```
def Board_after_play(board,player,slot):
   return board[:slot] + player + board[slot + 1:]
```

Example:

```
Board_after_play(board,       player, slot):
Board_after_play('.O.X...X.', 'O',       2):
   return board[:slot] + player + board[slot + 1:]
                '.O'       +  'O'     +  'X...X.'
   return board[:slot] + player + board[slot + 1:]
                '.OOX...X.'
```


Practice 1: Tic-Tac-Toe

- Trained Player:

Assume that valuesX[board] is the "value" for X of position board

```
def PlayerX(board):
        boards = [Board_after_play(board,'X',x)] for x in
Empty_slots(board)]
        v = [valuesX[b] for b in boards]
        return v.index(max(v))
```

- Training:
  - Initially, all valuesX[board] are zero
  - We play an automatic game,
    - if X wins, valuesX are increased for the boards used
    - if X looses, valuesX are decreased for the boards used

Practice 1: Tic-Tac-Toe

Suggested exercise:

(you just have to comment/uncomment some lines)

1. Run with learning_rate alpha = 0
2. Run with learning rate alpha = 0.1
3. With alpha = 0.1, save the values at the end of training
4. Change PlayerX to HumanPlayer, load the values and play against the computer

5. Later, you may try to test by yourself
   - try other values of the learning_rate
   - try a different number of games for learning
   - write your own Player routine

RECAP IN VIETNAMESE

Bài tập 1: Tic-Tac-Toe

- Trò chơi Tic Tac Toe trên một bàn cờ 3x3 được viết bằng Python và sử dụng ký hiệu 'X' và 'O' để đại diện cho hai người chơi.
- Cung cấp các hàm kiểm tra vị trí thắng, xác định danh sách vị trí trống trên bàn cờ, và chuyển đổi bàn cờ sau mỗi nước đi.
- Đề xuất một cách huấn luyện người chơi X: tăng giảm giá trị của các bàn cờ dựa trên kết quả của trò chơi, thắng thì +, thua thì -
- Mô tả quá trình huấn luyện ban đầu với giá trị ban đầu của bàn cờ là 0 và các giá trị này sẽ được điều chỉnh tăng hoặc giảm dựa trên kết quả của trận đấu

Bài tập 1: Tic-Tac-Toe

Dưới đây là một số yêu cầu cần thực hiện trong bài tập này:

1. Chạy chương trình với giá trị learning_rate alpha = 0. Sau đó, quan sát và ghi nhận kết quả.
2. Tiếp theo, chạy chương trình với giá trị learning_rate alpha = 0.1. Quan sát và ghi nhận sự khác biệt so với khi alpha = 0.
3. Với alpha = 0.1, lưu giá trị cuối cùng sau quá trình training.
4. Thay thế PlayerX bằng HumanPlayer và reload lại code.
5. Sau khi hoàn thành các yêu cầu trên, bạn có thể thử nghiệm tiếp bằng cách:
   - Thay đổi giá trị learning_rate để xem sự ảnh hưởng của nó đến quá trình học.
   - Thay đổi giá trị number of games được sử dụng để huấn luyện để xem liệu nó có ảnh hưởng đến hiệu suất của mô hình hay không.
   - Viết một quy trình Player riêng của bạn và thử nghiệm nó trong trò chơi để xem liệu nó có hoạt động tốt hay không.