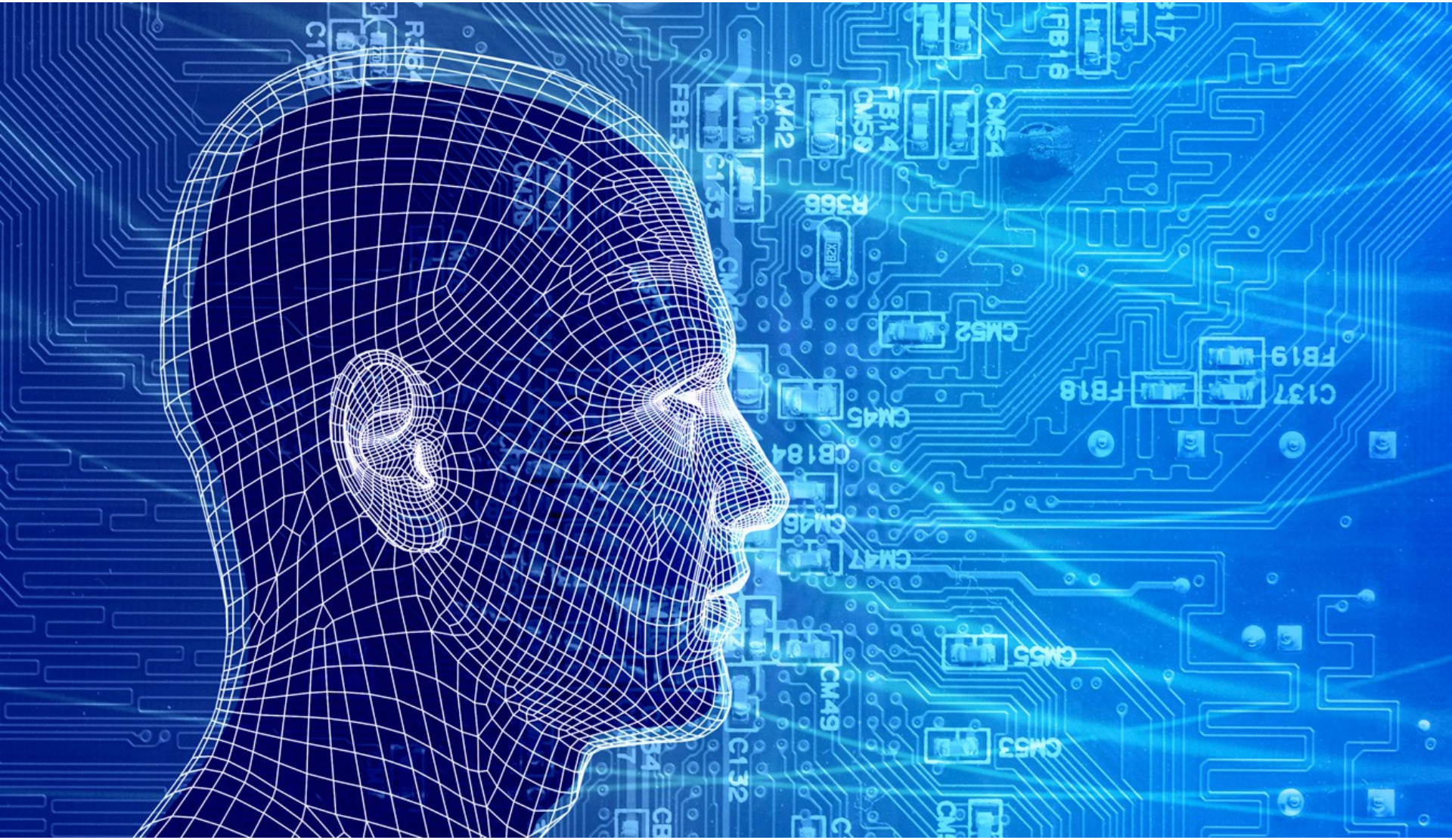
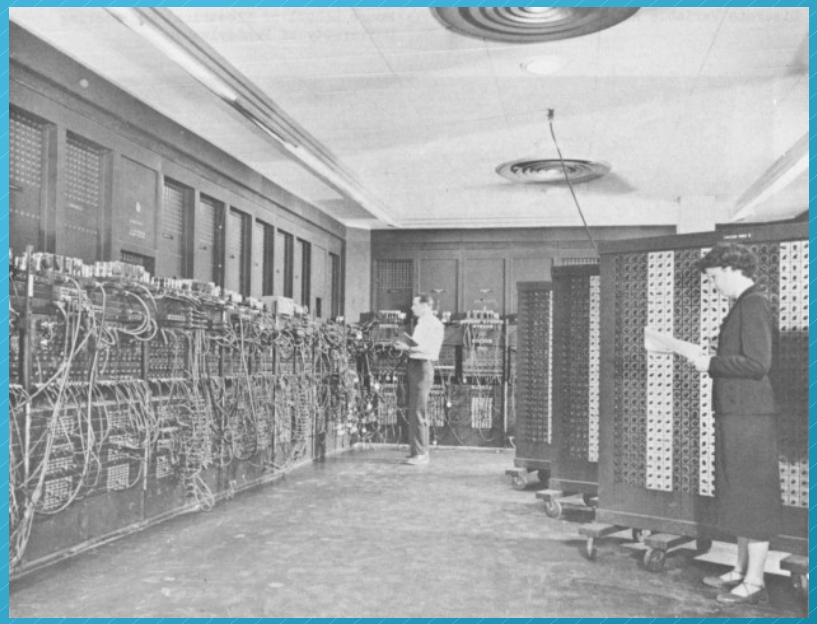
Can machines think like human beings and beyond!

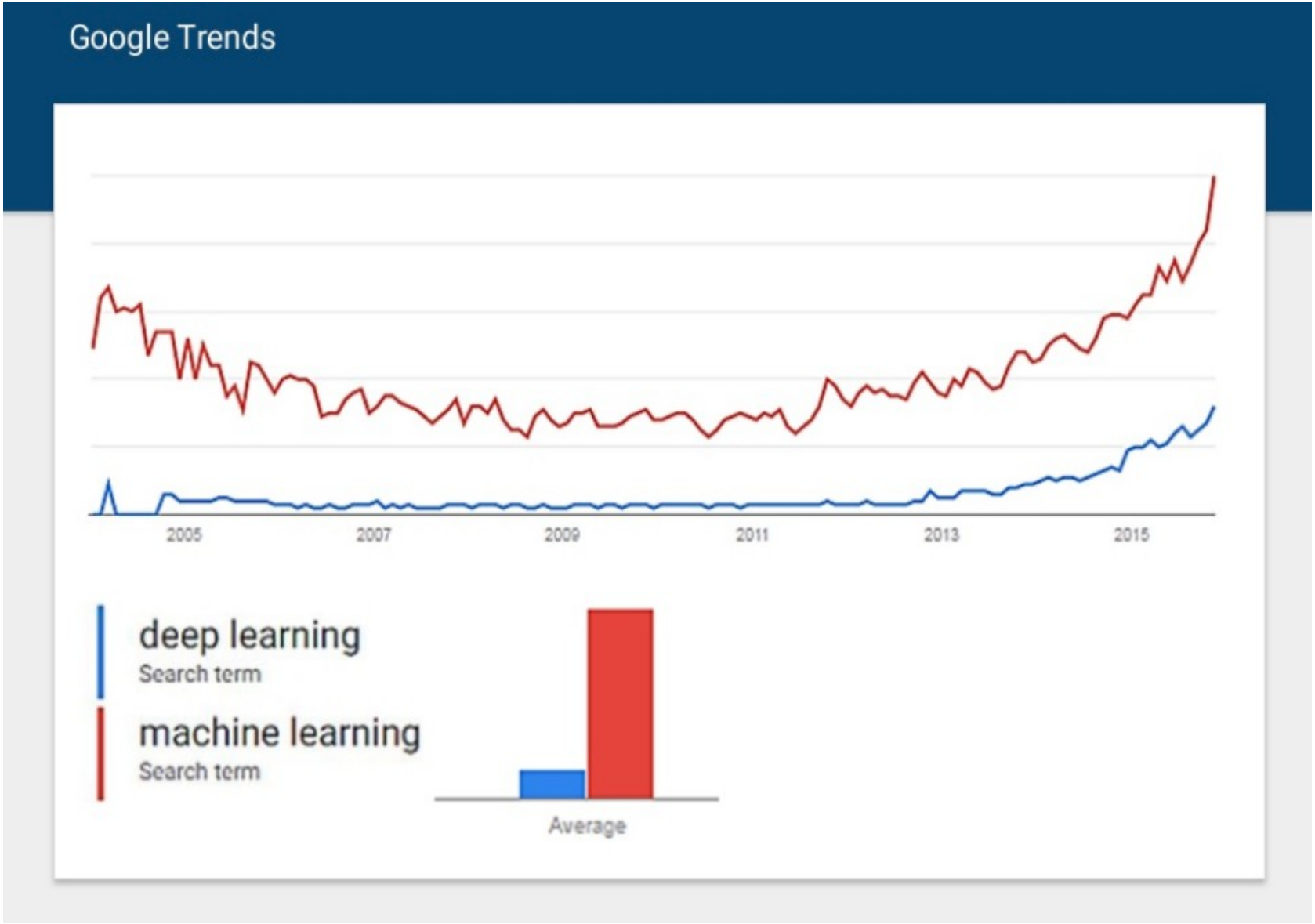


1941-First Electronic Computer

- ENIAC - Electronic Numerical Integrator and Computer
- An innovation that revolutionized the world!

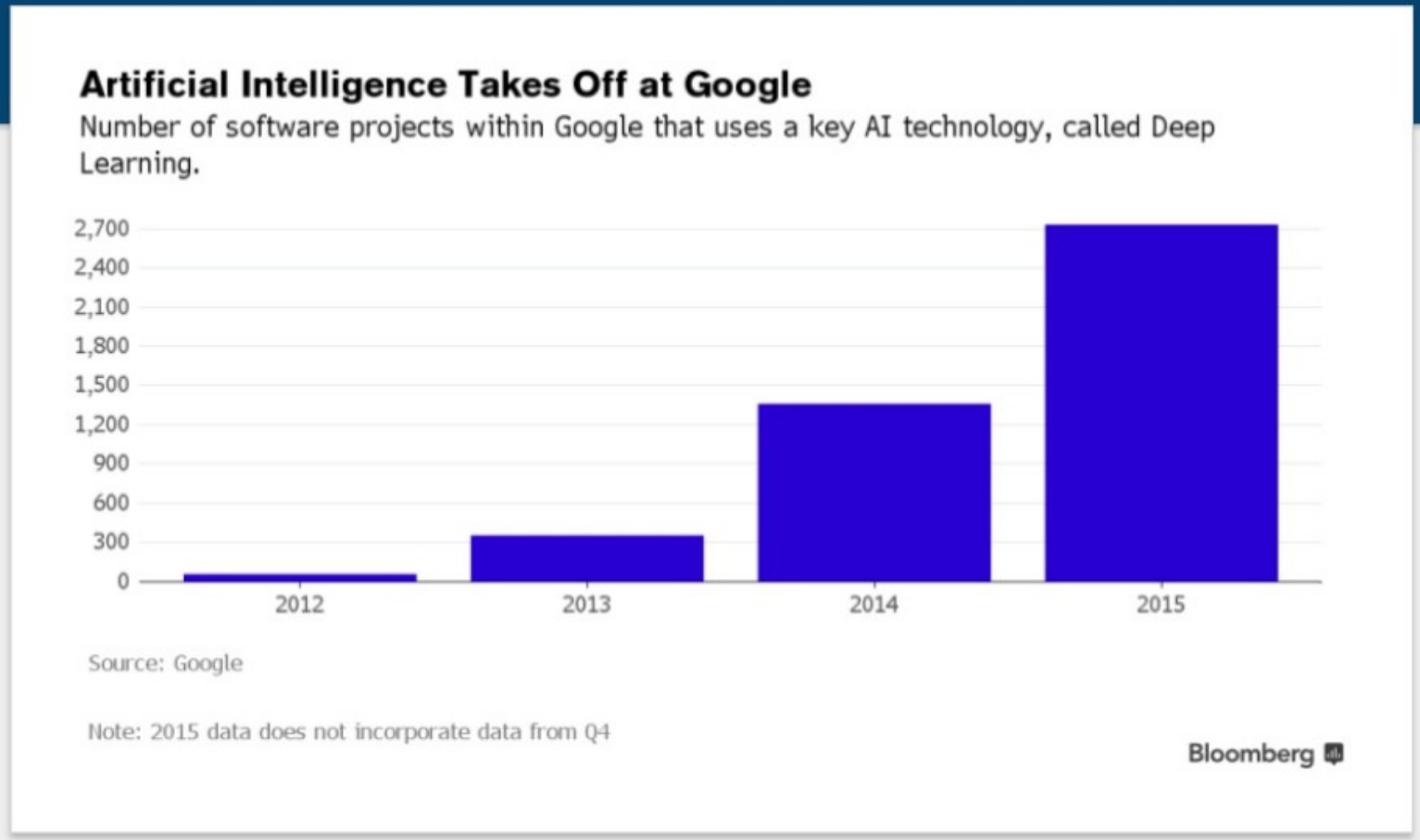


Who's interested?



AI projects at google

Google

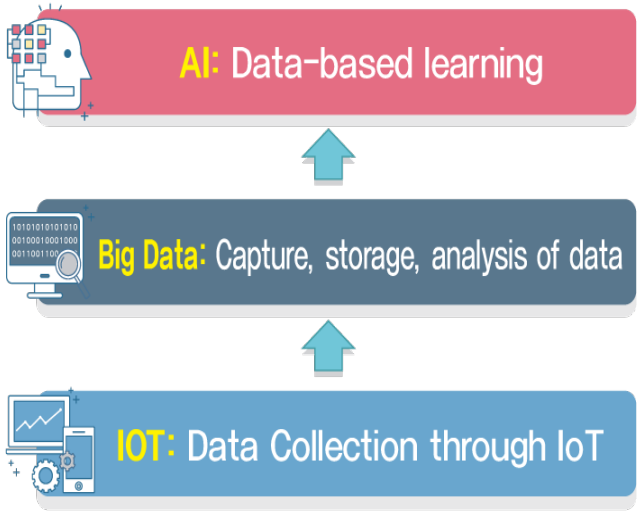


IOT and AI: Are they related?

Internet of Things is the concept of primarily connected devices that can be used to perform a given set of actions. For example, a home automation system that can control your entire power supply of your home devices.

The Internet of Things (IoT) is a system of interrelated computing devices, mechanical and digital machines, objects, animals or people that are provided with unique identifiers and the ability to transfer data over a network without requiring human-to-human or human-to-computer interaction.

Artificial intelligence is the simulation of human intelligence processes by machines, especially computer systems.



AI is a machine's ability to take decisions based on given input. For example, watch what you do repeatedly for a few days and copy that action.

If you link these together, the applications are numerous. A home automation system that knows when to switch on and off your devices based on your current activity is a combination of IoT and AI.

Domains for AI....

AI definition(simple)

Artificial intelligence (AI) - is a branch of computer science and engineering that deals with intelligent behavior, learning, and adaptation in machine

AI- Putting it to good use

Interested in watching it
In Action!

The Dartmouth Conference and the Name Artificial Intelligence

J. McCarthy, M. L. Minsky, N. Rochester, and C.E. Shannon. August 31, 1955. "We propose that a 2 month, 10 man study of artificial intelligence be carried out during the summer of 1956 at Dartmouth College in Hanover, New Hampshire. The study is to proceed on the basis of the conjecture that every aspect of learning or any other feature of intelligence can in principle be so precisely described that a machine can be made to simulate it."

And , AI was born!

The Origins of AI Hype

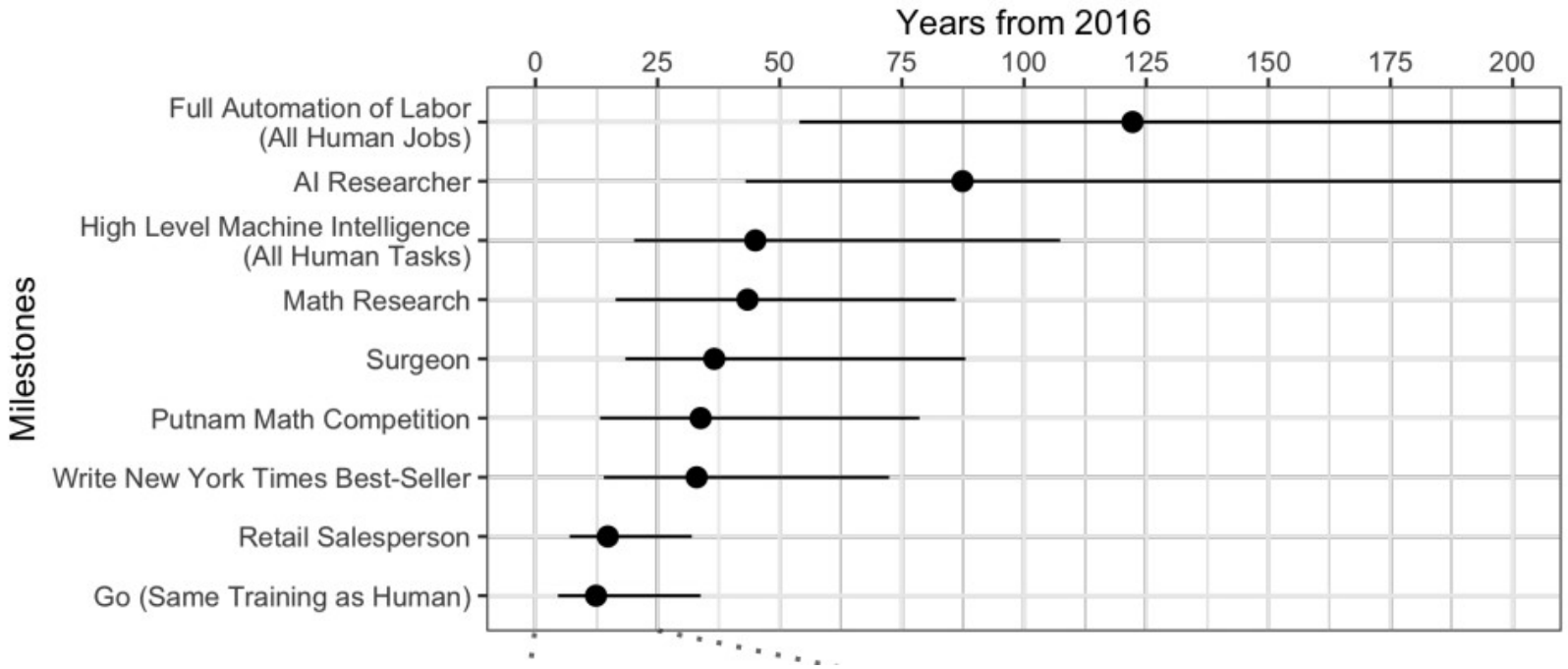
1950 Turing predicted that in about fifty years "an average interrogator will not have more than a 70 percent chance of making the right identification after five minutes of questioning".

1957 Newell and Simon predicted that "Within ten years a computer will be the world's chess champion, unless the rules bar it from competition."

AI History

1943	McCulloch & Pitts: Boolean circuit model of brain
1950	Turing's "Computing Machinery and Intelligence"
1952–69	Look, Ma, no hands!
1950s	Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine
1956	Dartmouth meeting: "Artificial Intelligence" adopted
1965	Robinson's complete algorithm for logical reasoning
1966–74	AI discovers computational complexity Neural network research almost disappears
1969–79	Early development of knowledge-based systems
1980–88	Expert systems industry booms
1988–93	Expert systems industry busts: "AI Winter"
1985–95	Neural networks return to popularity
1988–	Resurgence of probabilistic and decision-theoretic methods Rapid increase in technical depth of mainstream AI "Nouvelle AI": ALife, GAs, soft computing

Will Intelligent machines surpass human beings



Will Intelligent machines surpass human beings

Ellon Musk says ““Robots will do everything better than us“

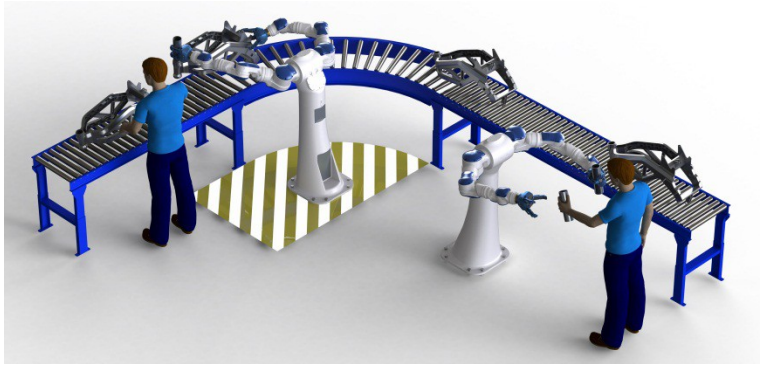
"There certainly will be job disruption. Because what's going to happen is robots will be able to do everything better than us. ... I mean all of us," saidÂ
Musk, speaking to the National Governors AssociationÂ in July. "Yeah, I am not sure exactly what to do about this. This is really the scariest problem to me, I will tell you."

Reality time for those Sci Fi(s)

Continental AG's SMART Factory

- Active RFID tags and Geo-location are used to move the tire components throughout the factory

- Collaborative robots
  - Robots are "shown" how to do a task once and then they can repeat that action
  - Reduces risks of injuries and reduces the need for additional assisting employees



What is Machine Learning?

Aspect of AI: creates knowledge

Definition:

“changes in [a] system that ... enable [it] to do the same task or tasks drawn from the same population more efficiently and more effectively the next time." (Simon 1983)

There are two ways that a system can improve:

1. By acquiring new knowledge
   - acquiring new facts
   - acquiring new skills
2. By adapting its behavior
   - solving problems more accurately
   - solving problems more efficiently

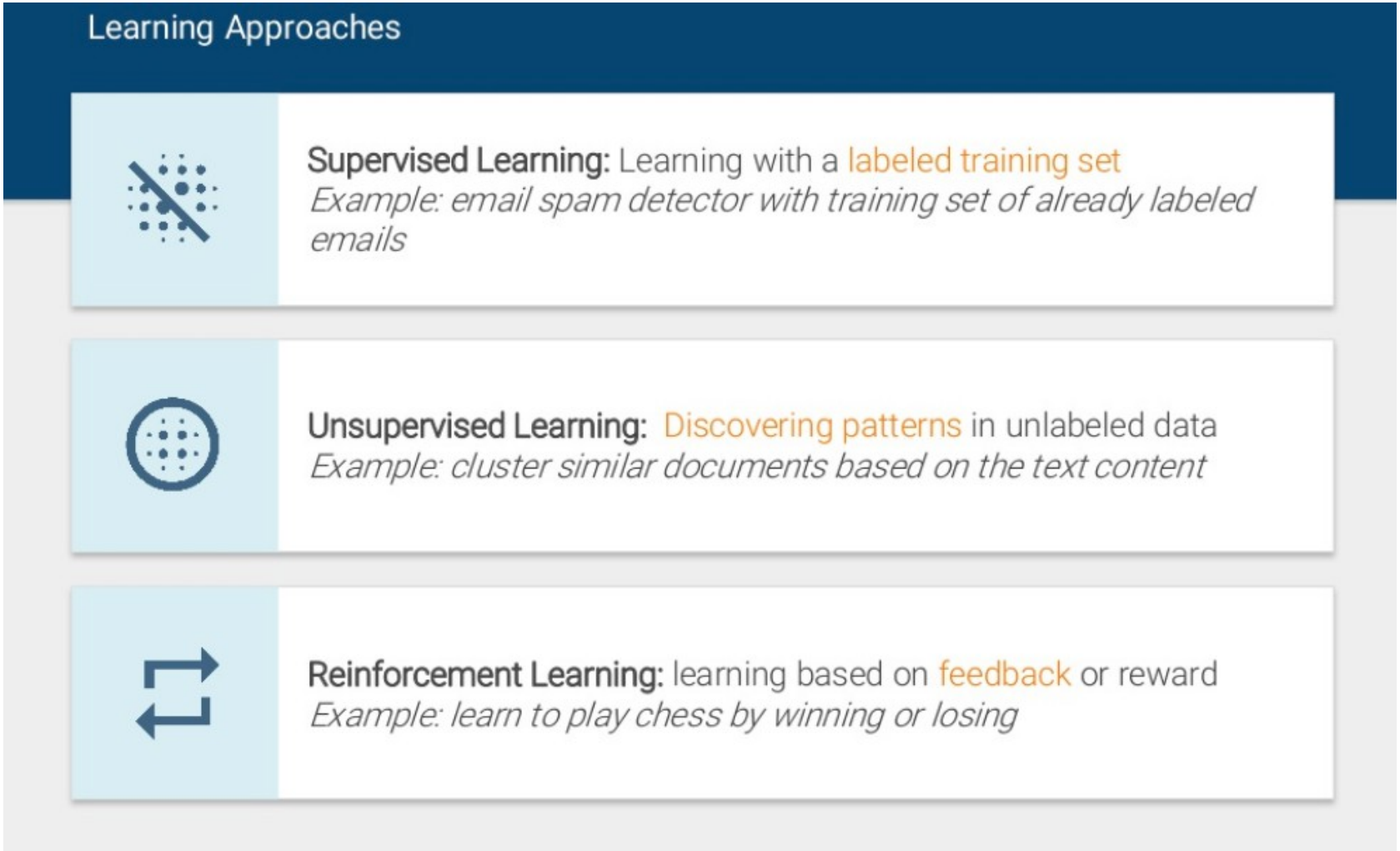
What is Learning?

Herbert Simon: “Learning is any process by which a system improves performance from experience.”

What is the task?

- Classification
- Categorization/clustering
- Problem solving / planning / control
- Prediction
- others

How can machines learn?



Inductive (Supervised) Learning

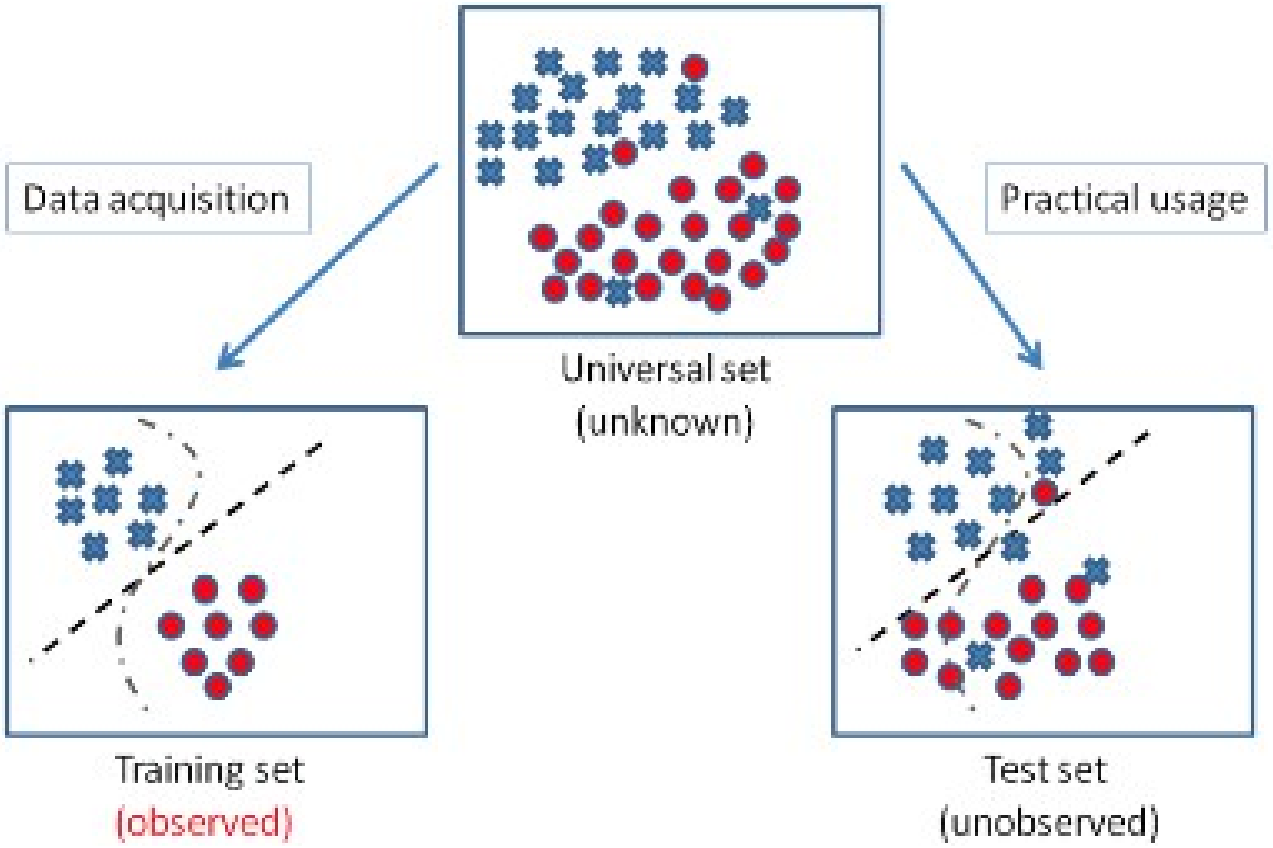
Basic Problem: Induce a representation of a function (a systematic relationship between inputs and outputs) from examples.

target function $f: X \rightarrow Y$
example $(x, f(x))$
hypothesis $g: X \rightarrow Y$ *such that* $g(x) = f(x)$

x = set of attribute values (***attribute-value representation***)
x = set of logical sentences (*first-order representation*)

Y = set of discrete labels (***classification***)
Y = ! (*regression*)

Learning : Training and Test Set



Lets Learn Python

Course Contents

- About Python, Why Python?
- Running Python
- Working with basic types
- Types and Operators
- Basic Statements
- Functions
- Scope Rules (Locality and Context)
- Classes & objects, operator overloading
- Some Useful Packages and Resources

History [edit]

Main article: History of Python

Guido van Rossum, the creator of Python

Python was conceived in the late 1980s,[29] and its implementation began in December 1989[30] by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC language (itself inspired by SETL)[31] capable of exception handling and interfacing with the Amoeba operating system.[7] Van Rossum remains Python's principal author. His continuing central role in Python's development is reflected in the title given to him by the Python community: *Benevolent Dictator For Life* (BDFL).

On the origins of Python, Van Rossum wrote in 1996:[32]

> ...In December 1989, I was looking for a "hobby" programming project that would keep me occupied during the week around Christmas. My office ... would be closed, but I had a home computer, and not much else on my hands. I decided to write an interpreter for the new scripting language I had been thinking about lately: a descendant of ABC that would appeal to Unix/C hackers. I chose Python as a working title for the project, being in a slightly irreverent mood (and a big fan of *Monty Python's Flying Circus*).
>
> — Guido van Rossum

Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode. With this release, the development process became more transparent and community-backed.[33]

Python 3.0 (initially called Python 3000 or py3k) was released on 3 December 2008 after a long testing period. It is a major revision of the language that is not completely backward-compatible with previous versions.[34] However, many of its major features have been backported to the Python 2.6.x[35] and 2.7.x version series, and releases of Python 3 include the `2to3` utility, which automates the translation of Python 2 code to Python 3.[36]

Python 2.7's end-of-life date was initially set at 2015, then postponed to 2020 out of concern that a large body of existing code could not easily be forward-ported to Python 3.[37][38]

Python 3.6 had changes regarding UTF-8 (in Windows, PEP 528 and PEP 529) and Python 3.7.0b1 (PEP 540) adds a new "UTF-8 Mode" (and overrides POSIX

Why Use Python?

Python is object-oriented

Structure supports such concepts as polymorphism, operation overloading, and multiple inheritance

It's free (open source)

Downloading and installing Python is free and easy

Source code is easily accessible

Free doesn't mean unsupported! Online Python community is huge

It's portable

Python runs virtually every major platform used today

As long as you have a compatible Python interpreter installed, Python programs will run in exactly the same manner, irrespective of platform

It's powerful

Dynamic typing

Built-in types and tools

Library utilities

Third party utilities (e.g. Numeric, NumPy, SciPy)

Automatic memory management

Why Use Python?

It's mixable

Python can be linked to components written in other languages easily

- Linking to fast, compiled code is useful to computationally intensive problems
- Python is good for code steering and for merging multiple programs in otherwise conflicting languages

Python/C integration is quite common

It's easy to use

Rapid turnaround: no intermediate compile and link steps as in C or C++

Python programs are compiled automatically to an intermediate form called *bytecode*, which the interpreter then reads

This gives Python the development speed of an interpreter without the performance loss inherent in purely interpreted languages

It's easy to learn

Structure and syntax are pretty intuitive and easy to grasp

Running Python

```
Apples-MacBook-Pro:objectDetectionAI apple$ python
Python 3.6.4 |Anaconda, Inc.| (default, Jan 16 2018, 12:04:33)
[GCC 4.2.1 Compatible Clang 4.0.1 (tags/RELEASE_401/final)] on
darwin
Type "help", "copyright", "credits" or "license" for more information.
>>>
>>> print('hello')
hello
>>>
```

In addition to being a programming language, Python is also an interpreter. The interpreter reads other Python programs and commands, and executes them. Note that Python programs are compiled automatically before being scanned into the interpreter. The fact that this process is hidden makes Python faster than a pure interpreter.

First pieces of code

Playing with the interpreter

```
>>> 3+4
7
>>> 270*5+3*(200/56)
1360.7142857142858
>>> 12*7+13*5
149
```

A Code Sample

```
x = 34 - 23              # A comment.
y = "Hello"              # Another one.
z = 3.45
if z == 3.45 or y == "Hello":
    x = x + 1
    y = y + " World"     # String concat.
print x
print y
```

Running Python Script file

Python scripts can be written in text files with the suffix .py. The scripts can be read into the interpreter in several ways:

Examples:

```
$ python script.py
# This will simply execute the script and return to the terminal afterwards
$ python -i script.py
# The -i flag keeps the interpreter open after the script is finished running
$ python
>>> execfile('script.py')
# The execfile command reads in scripts and executes them immediately, as though they had been typed into the interpreter directly
$ python
>>> import script  # DO NOT add the .py suffix.  Script is a module here
# The import command runs the script, displays any unstored outputs, and creates a lower level (or context) within the program.
```

Run first Python Script

Suppose the file helloPython.py contains the following lines:

```
print 'Hello world'
myTestArray = [0,1,2,3,4]
```

Let's run this script in each of the ways described on the last slide:

```
$ python helloPython.py
Hello world
$
```

The script is executed and the interpreter is immediately closed. x is lost.

```
$ python -i script.py
Hello world
>>> print(x)
[0,1,2,3,4]
>>>
```

“Hello world” is printed, x is stored and can be called later, and the interpreter is left open

Types and Operators: Types of Numbers

Python supports several different numeric types

Integers

Examples: `0, 1, 1234, -56`

Integers are implemented as C longs

Note: dividing an integer by another integer will return only the integer part of the quotient, e.g. typin

Long integers

Example: `99999999999999999999L`

Must end in either `l` or `L`

Can be arbitrarily long

Floating point numbers

Examples: `0., 1.0, 1e10, 3.14e-2, 6.99E4`

Implemented as C doubles

Division works normally for floating point numbers: `7./2. = 3.5`

Operations involving both floats and integers will yield floats:

`6.4 - 2 = 4.4`

Types and Operators: Operations on Numbers

Basic algebraic operations

Four arithmetic operations: `a+b, a-b, a*b, a/b`

Exponentiation: `a**b`

Other elementary functions are not part of standard Python, but included in packages like NumPy and SciP

Comparison operators

Greater than, less than, etc.: `a < b, a > b, a <= b, a >= b`

Identity tests: `a == b, a != b`

Bitwise operators

Bitwise or: `a | b`

Bitwise exclusive or: `a ^ b` # Don't confuse this with exponentiation

Bitwise and: `a & b`

Shift `a` left o

Other

Not surprisingly, Python follows the basic PEMDAS order of operations

Python supports mixed-type math. The final answer will be of the most complicated type used.

Whitespace is so important!

Whitespace

Whitespace is meaningful in Python: especially indentation and placement of newlines.

- **Use a newline to end a line of code.**
  - Use `\` when must go to next line prematurely.
- **No braces `{ }` to mark blocks of code in Python… Use *consistent* indentation instead.**
  - The first line with *less* indentation is outside of the block.
  - The first line with *more* indentation starts a nested block
- **Often a colon appears at the start of a new block. (E.g. for function and class definitions.)**

Thank You!