Lecture One: Introduction - Machine Learning and Python

COMP3032 Introduction to Machine Learning

Unit Information

What is Machine Learning (ML)

- ML Definition
- ML Examples
- Why ML

Machine learning algorithms

- Supervised learning
- Unsupervised learning
- Reinforcement learning

Introduction to Python

- Data types and operators
- Statements, functions and modules
- Library NumPy and Pandas
- Maths Revision

Unit Information

This unit provides a comprehensive introduction to machine learning, including fundamental concepts, algorithms and essential techniques of Machine Learning.

- 2 hour lecture weekly
- 2 hour practical weekly from the 2nd week
- 3 Quizzes (15%)
- 2 Assignments (45%)
- Examination (40%)
- Refer to LG for more details.

What is ML - ML Definition

Machine Learning is the science (and art) of programming computers so they can learn from data.

- Arthur Samuel (1959): Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed.
- Tom Mitchell (1998): A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Example ML Program: spam filters

Spam email filters are ML program that can learn to flag spam by being given examples of spam emails (e.g., flagged by users)and examples of regular (i.e., nonspam or ham) emails

- training set: the examples that the system uses to learn
- trailning instance: each training example

In this example:

- task T: flagging emails as spam or not spam
- experience E: watching user flag emails as spam or not spam
- performance P: the ratio (or number) of correctly classified emails as spam or not spam

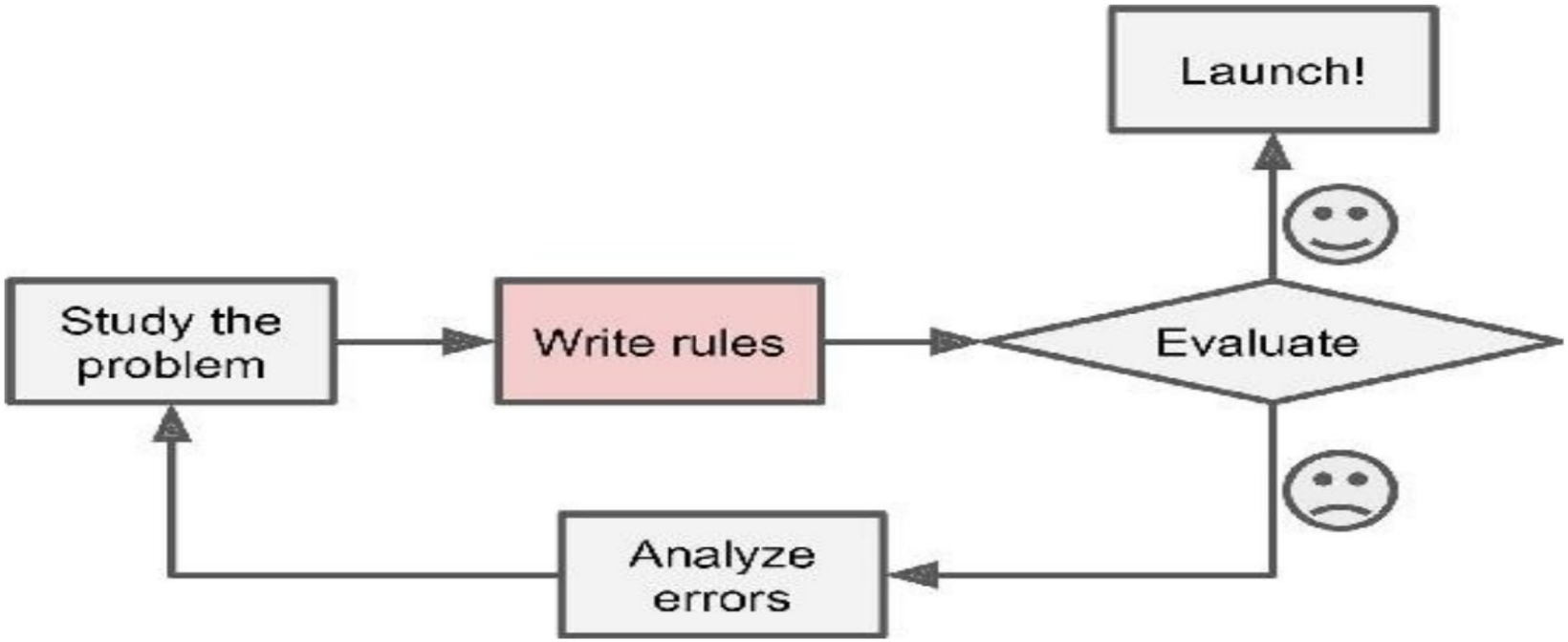
Why ML

Almost any application that involves understanding data or signals that come from the real world can be best addressed using machine learning

- Predict (estimate) prices for items (e.g., houses, cars) based on properties of that item
- Distinguish spam emails from useful emails
- Handwriting recognition, Natural Language Processing (NLP)
- Distinguish different kinds of objects in images
- Amazon, Netflix product recommendations
- data mining: discover patterns from large amounts of data that were not immediately obvious: Web click data, medical records, biology

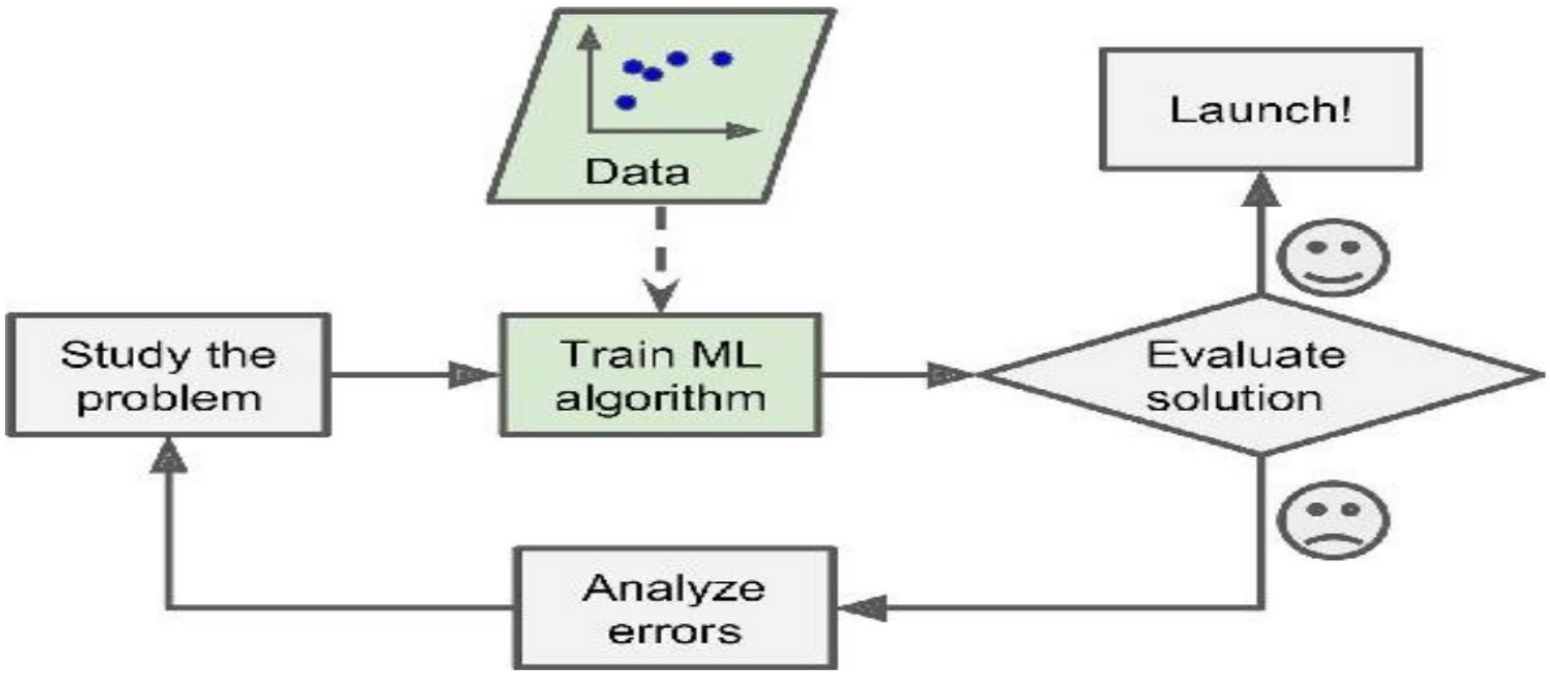
Traditional Programming

The implementation of spam filters through traditional programming techniques:



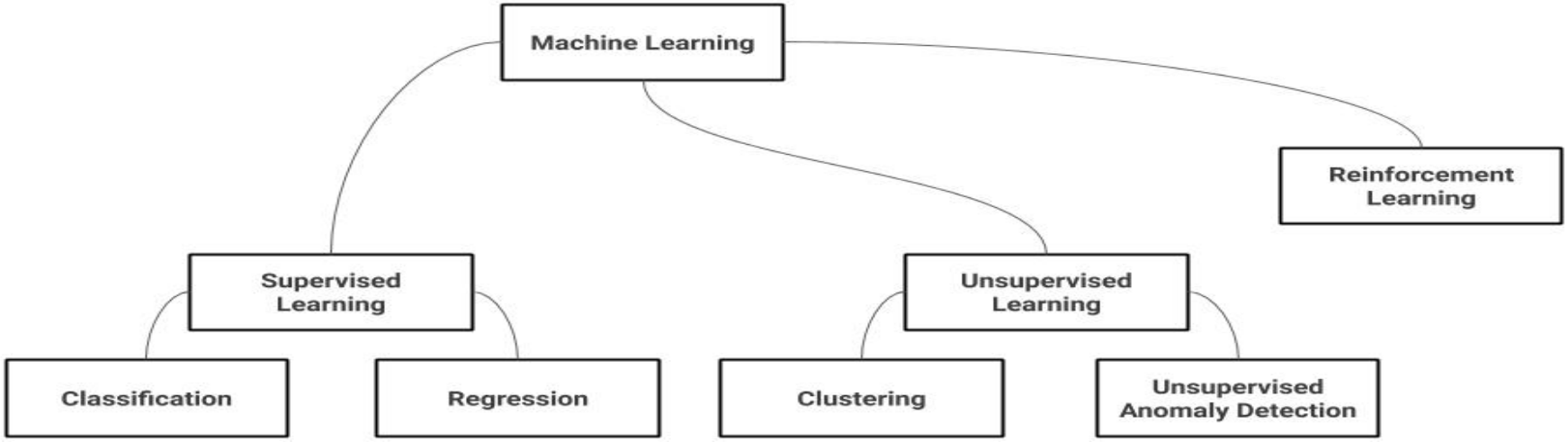
ML

An ML based spam filter automatically "learns" which word patterns are good predictors for spam:



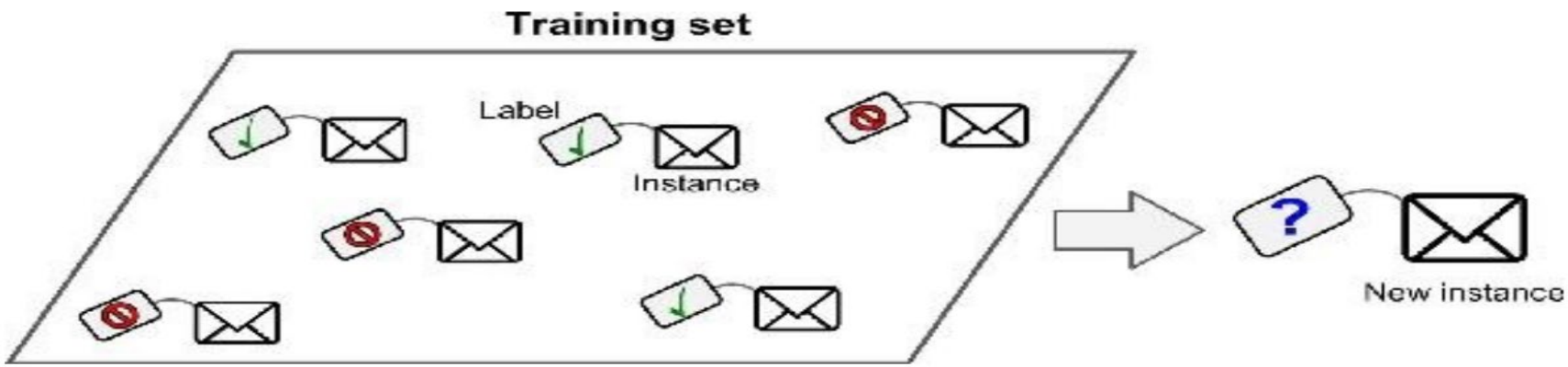
Machine learning algorithms

- Supervised learning
- Unsupervised Learning
- Reinforcement Learning



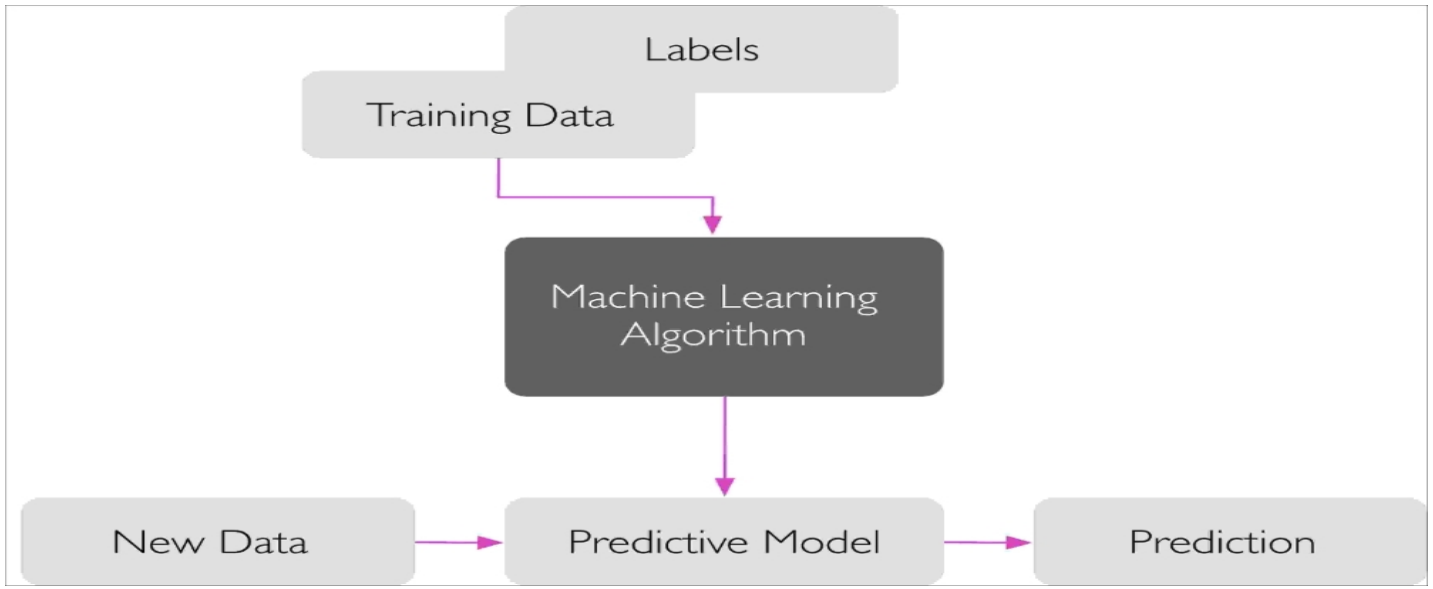
Supervised learning

In supervised learning, the training data contains the preferred answers



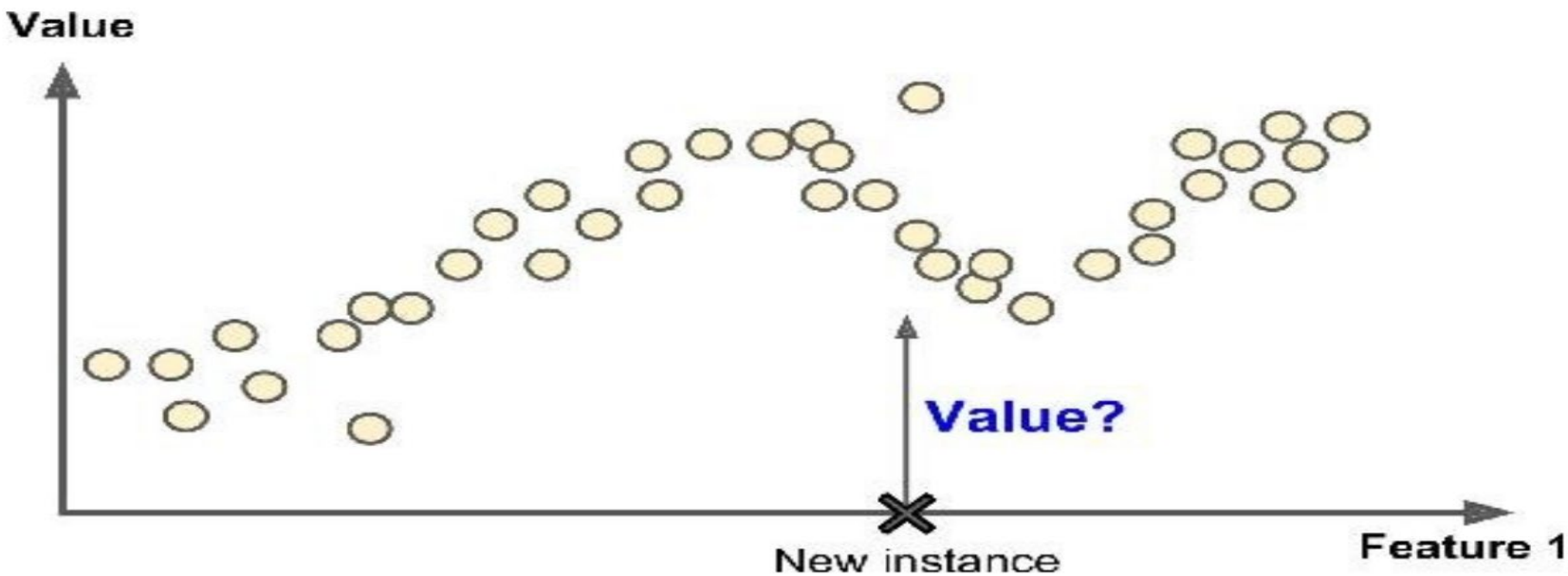
Supervised learning: predict future

The main goal in supervised learning is to learn a model from labeled training data that allows us to make predictions about unseen or future data.



Supervised learning: regression

Regression predict continuous valued output
Example: predict car prices based on a set of given features (mileage, age, brand, etc.)



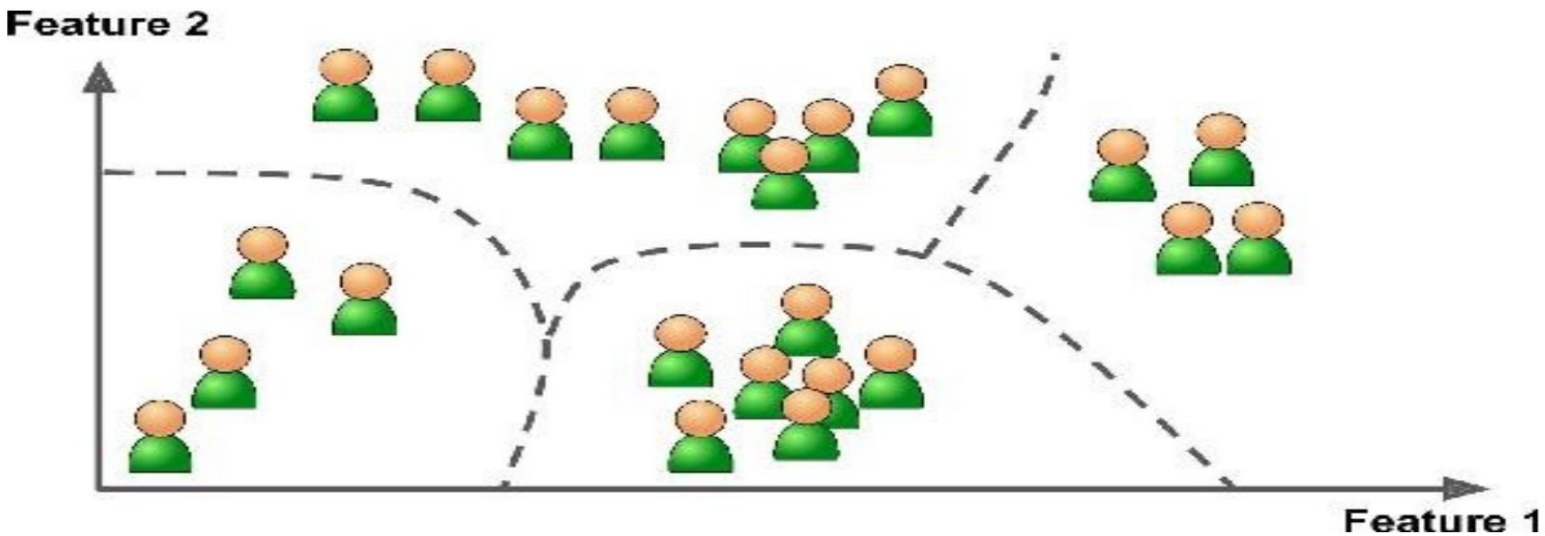
Supervised learning: classification

Classification predict discrete valued output

- spam email filter: spam or not spam
- predict tumor type: benign or malignant
- image recognition : table, chair, lamp or neither

Unsupervised learning

Unsupervised learning starts with training data without labels.
Clustering: grouping unlabelled data based on some connections



Clustering Algorithms

Clustering algorithms, in essence, try to put the data samples into clusters so that it minimizes the intracluster distances and maximizes the intercluster distances.

- Samples in the same cluster to be as similar as possible
- Samples from different clusters to be as different as possible

Dimensionality Reduction (DR)

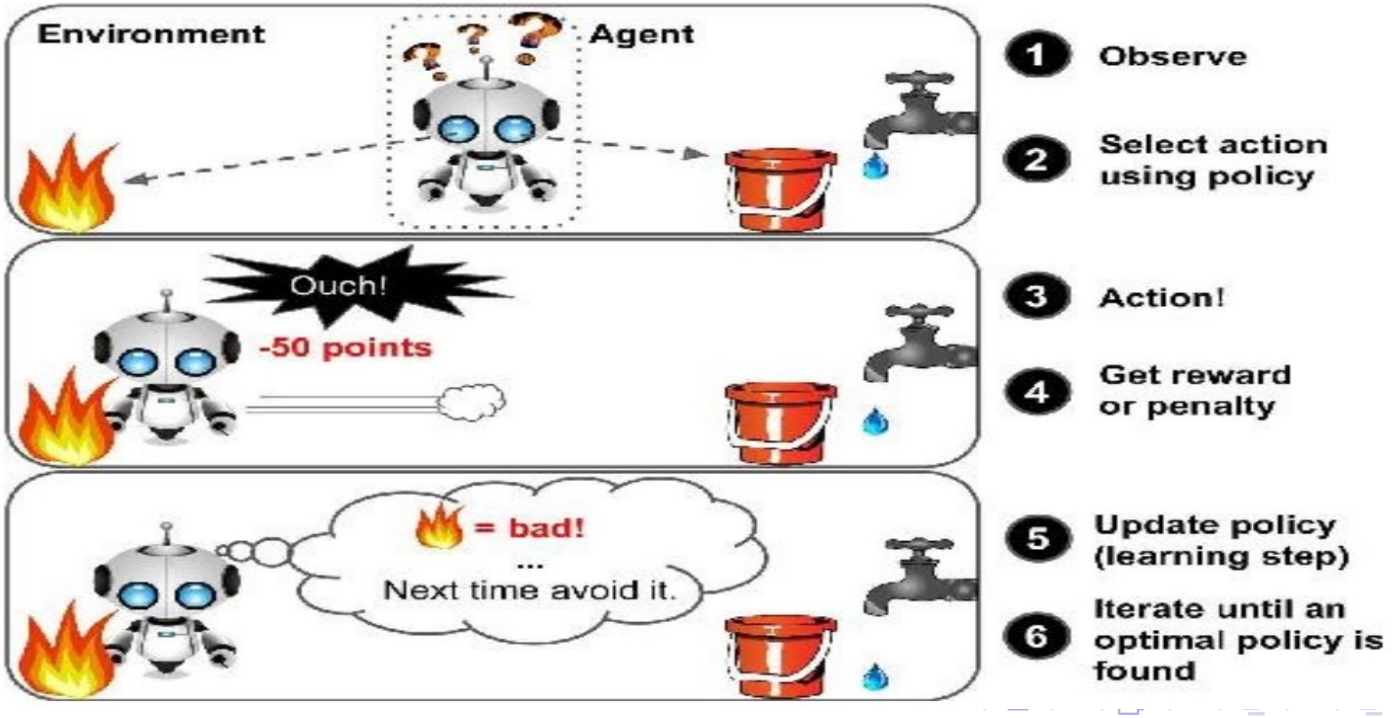
- Many machine learning problems involve thousands or even millions of features
- Make training extremely slow, and much harder to find a good solution
- Difficult to visualize high dimensional data set
- Dimensionality Reduction (DR): merge several correlated features into one
- E.g. car's mileage correlated with age so merge into one feature

Manifold Learning (MAL)

- Main Approaches for DR: Projection and Manifold Learning
- MAL: seeks to describe datasets as low-dimensional manifolds embedded in high-dimensional spaces
- An example: a two-dimensional piece of paper put/embedded in a three-dimensional world (can be bent and rolled)

Reinforcement learning

- can observe the environment
- select and perform actions
- get rewards and penalties in return
- learn by itself what is the best strategy (policy) to get the most reward over time



Differences

- Supervised learning: labeled data, have feedback, predict future/outcome
- Unsupervised learning: no labels, no feedback, find hidden structures in data
- Reinforcement learning: no labels, reward system, learn from series of actions

Introduction to Python

- Anaconda https://www.anaconda.com/products/individual
  - Spider
  - Jupyter Notebook
  - Anaconda Prompt
- Interpreted language: Python programs are executed by an interpreter.
  - interactive mode
  - script mode

Basic Data types

int	$1, 2, 100, ...$
float	2.5, 7.8
boolean	True, False
string	'hi there'
lists	['apple','grape', 'kiwi'], [1,2,-8,100]

Strings

A string is made up of characters. It must be in quotes, single quotes or double quotes: 'Good morning!', "hi there"

```
+: concatenation, 'hello'+' '+'world'
    'hello world'
*: repetition (multiplication), 'ha'*3
    'hahaha'
index: indexes start at 0, 'hello'[0] # h.
    # Spaces inside a string are counted.
\:  can be used to escape quotes
```

List

A list is a comma-separated values (items) between square brackets.
fruits=['apple','grape','kiwi']

- Lists might contain items of different types, but usually the items all have the same type.
- use index to access each element: fruits[0] # apple
- can be sliced: fruits[:], fruits[-1]
- +: concatenation of two lists
- It is possible to nest lists (lists containing other lists)

Variables

A varialbe is a name that refers to a value.

- starts with a letter, can contain letters, numbers and underscore characters.
- normally starts with a lowercase letter
- can be arbitrarily long
- choose names that are meaningful
- function *type*() gives the type of the value it refers to

Example:

```
>>> pi = 3.1415926535897932
>>> type(pi)
<class  'float'>
```

Arithmetic operators

- $+$ add: 2+4
- $-$ subtract: 50-90
- $*$ multiply: 2*4
- $/$ divide: 6/5 (1.2, division always returns a floating point number)
- $//$ floor division: 6//5 (1, discards the fractional part)
- $**$ exponentiation: 2**4

function : *round*(*float_num*, *num_of _decimals*)
round() is a built-in function. It will return a float number that will be rounded to the decimal places which are given as input. If the decimal places to be rounded are not specified, it is considered as 0, and it will round to the nearest integer.

Comparison operators

==	equal to
!=	not equal to
>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to

logical operators

and $x > 0$ and $x < 10$

or $x\%2 == 0$ or $x\%3 == 0$

not not $x\%2 == 0$

Any nonzero number is interpreted as 'true': 5 and True

Conditional statements

Make a selection based on the conditions.
The if/elif/else statement:

```
if x > 0:
  print('x is positive')
elif x==0:
  print('x is 0')
else:
  print('x is negative')
```

- indented body
- pass statement: does nothing
- check conditions in order, and the first true branch executes
- One conditional can be nested within another

Loops

The while statement:

```
n=5
while n>0 :
    print('n is:', n)
n=n-1
```

- indented body
- break statement: jump out of the loop
- continue statement: continue with the next iteration

Loops

The for statement: iterates over the items of any sequence (a list or a string), in the order that they appear in the sequence.

```
>>> fruits = ['apple', 'grape', 'kiwi']
>>> for w in fruits:
...     print(w, len(w))
...
apple 5
grape 5
kiwi 4
```

- indented body
- break statement: jump out of the loop
- continue statement: continue with the next iteration of the loop

Functions

A function is a named sequence of statements that performs a computation.

```
def square(x):
    return x ** 2

result = square(4)
    # result is 16
```

- The keyword *def* introduces a function definition
- *def* must be followed by the function name and the parenthesized list of formal parameters
- The body of the function starts at the next line
- The body must be indented.

Modules

A module is a file containing Python definitions and statements.

- The file name is the module name with the suffix *.py* appended: *mymodule.py*
- A module can be imported into other modules
- python standard library contains many modules: math, random, time, calendar, os, urllib,
- math module provides most of the familiar mathematical functions

```
>>> import math
>>> r=0.7
>>> h=math.sin(r)
>>> print(h)
0.644217687237691
```

Modules

```
>>>import random
>>>random.randint(1,100)
# Generate a random number between 1-100

>>> import time
>>> time.tzname
# What timezone does your computer think it is in?

>>> import calendar
>>> calendar.prmonth(2022, 7)
# Print a calendar for this month!
```

Modules

```
>>> import os
>>> for file in os.listdir("/home/pi"):print(file)
# Print the names of all the files in a directory

>>> import urllib.request
>>> myurl = "http://www.google.com"
>>> data = urllib.request.urlopen(myurl).read()
>>> print(data)
# Open a web page and read it
```

An Example: let's play a game!

```
from random import randint
secret_number = randint(1, 10)
while True:
    guess = input("What number am I thinking of? ")
    if secret_number == int(guess):
        print("Yay! You got it.")
        break
    elif secret_number > int(guess):
        print("No, that's too low.")
    else:
        print("No, that's too high.")
```

NumPy

NumPy is a python library that is the core for scientific computation under python.

- Provides high-performance multidimensional array objects and tools
- Easier working with these arrays
- Usually the NumPy module is imported under the 'np' name

```
>>> import numpy as np
>>> arr1 = np.array([0,2,4,6,8,10,12,14,16,18])
>>> arr2 = np.array([[0,2],[4,6],[8,10],[12,14],
                                        [16,18]],dtype='int64')
>>> arr1
>>> arr2
>>> arr1[0]
>>> arr2[1]
```

Pandas DataFrames

Pandas (for Python Data Analysis Library) is a software library written primarily for the manipulation and analysis of data.

- It offers the data structures and operations for the manipulations of numerical tables and time series data
- Its key data structure is called DataFrame
- DataFrame is a two-dimensional, size mutable and potentially heterogeneous tabular data with labelled axes (rows and columns)
- It provides data manipulations at a high-level
- It is built from the numpy package

Pandas DataFrames

There are mainly two methods for creating DataFrames

- From Python list: start from standard python list and then declare them as DataFrames
- From importing external files (e.g., CSV files): use the read_csv('path') DataFrame method:

An Example: Polygons

```
# It's customary to call pandas pd when importing it
import pandas as pd

polygons = {
    'Name': [
        'Triangle', 'Quadrilateral', 'Pentagon', 'Hexagon',
        'Heptagon', 'Octagon', 'Nonagon', 'Decagon',
        'Hendecagon', 'Dodecagon', 'Tridecagon',
        'Tetradecagon'],
     # Range parameters are the start, the end of the
     # range and the step
     'Sides': range(3, 15, 1),
}
polygons_data_frame = pd.DataFrame(polygons)
polygons_data_frame.sort_values('Name').head(5)
```

Polygons

	Name	Sides
7	Decagon	10
9	Dodecagon	12
8	Hendecagon	11
4	Heptagon	7
3	Hexagon	6

Scalar, Vector and Matrix

- Scalar: a single number
- Vector: a list of numbers
  - a row or a column
  - machine learning: a column by default
- Matrix: a two dimensional array of numbers (m rows and n columns)
  - Transpose
  - Multiplication (Dot Product)