台大資管營 微課程

Winter 2025

人工智慧導論 (Introduction to AI)

講師： 林杰

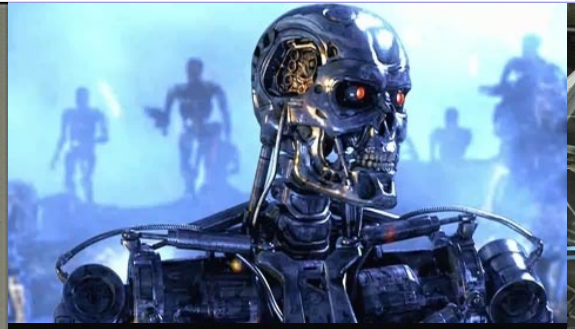
國立臺灣大學 資訊管理學系

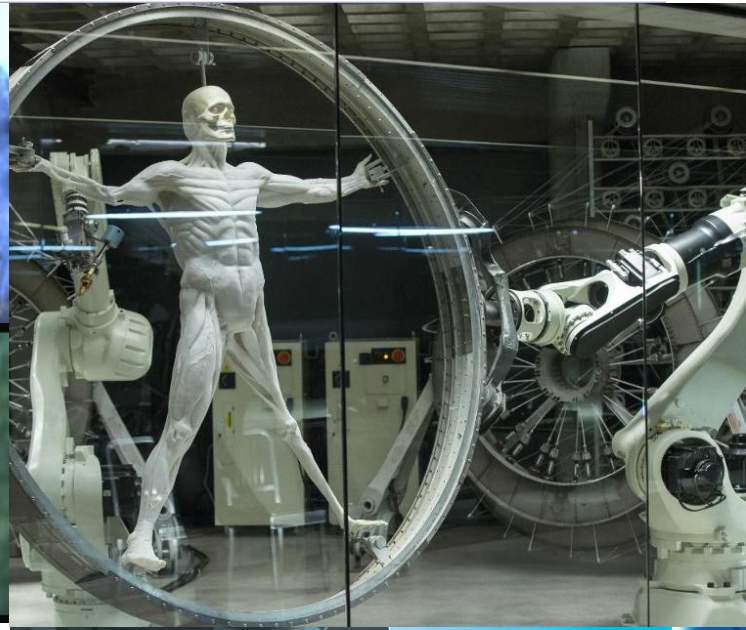
These slides are primarily adapted from those created by Dan Klein and Pieter Abbeel (CS188: Introduction to AI, UC Berkeley).

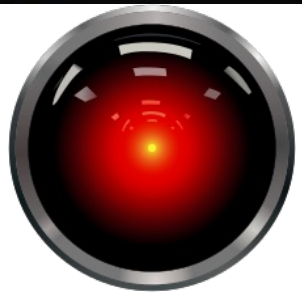
Additional adaptations were made from slides by:

- Hsin-Min Lu (IM5056: Statistical Learning and Deep Learning, NTU),
- Shang-Tse Chen and Vivian Chen (CSIE3005: Foundations of Artificial Intelligence, NTU).

AI in Science Fiction

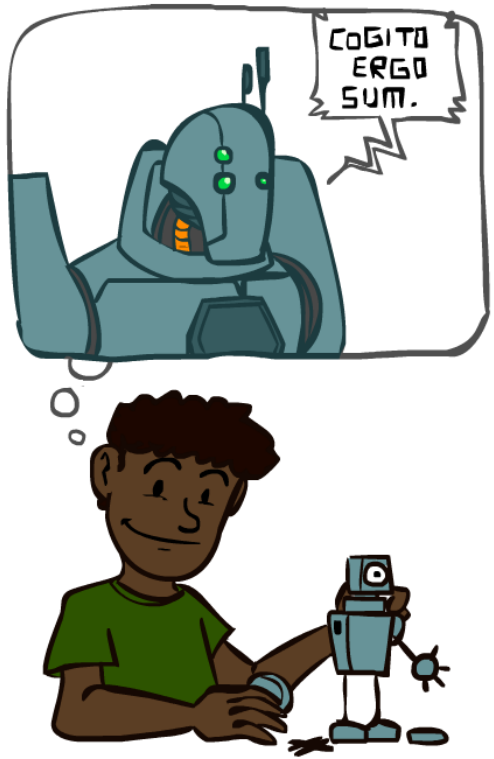






A (Short) History of AI

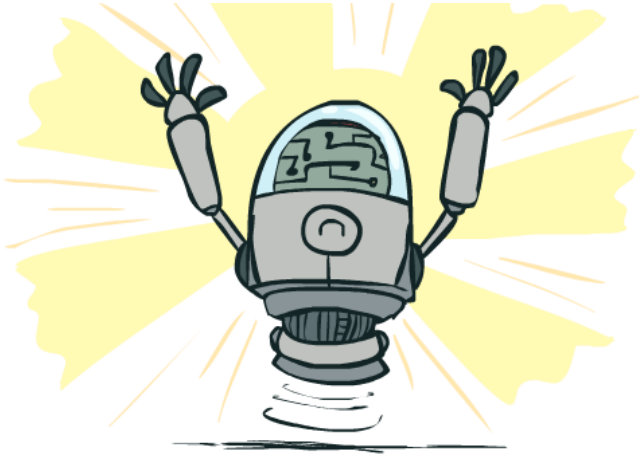
- 1940-1950: Early days
  - 1943: McCulloch & Pitts: Boolean circuit model of brain
  - 1950: Turing's "Computing Machinery and Intelligence"
- 1950—70: Excitement: Look, Ma, no hands!
  - 1950s: Early AI programs, including Samuel's checkers program, Newell & Simon's Logic Theorist, Gelernter's Geometry Engine
  - 1956: Dartmouth meeting: "Artificial Intelligence" adopted
  - 1965: Robinson's complete algorithm for logical reasoning
- 1970—90: Knowledge-based approaches
  - 1969—79: Early development of knowledge-based systems
  - 1980—88: Expert systems industry booms
  - 1988—93: Expert systems industry busts: "AI Winter"
- 1990—: Statistical approaches
  - Resurgence of probability, focus on uncertainty
  - General increase in technical depth
  - Agents and learning systems… "AI Spring"?
- 2000—: Where are we now?



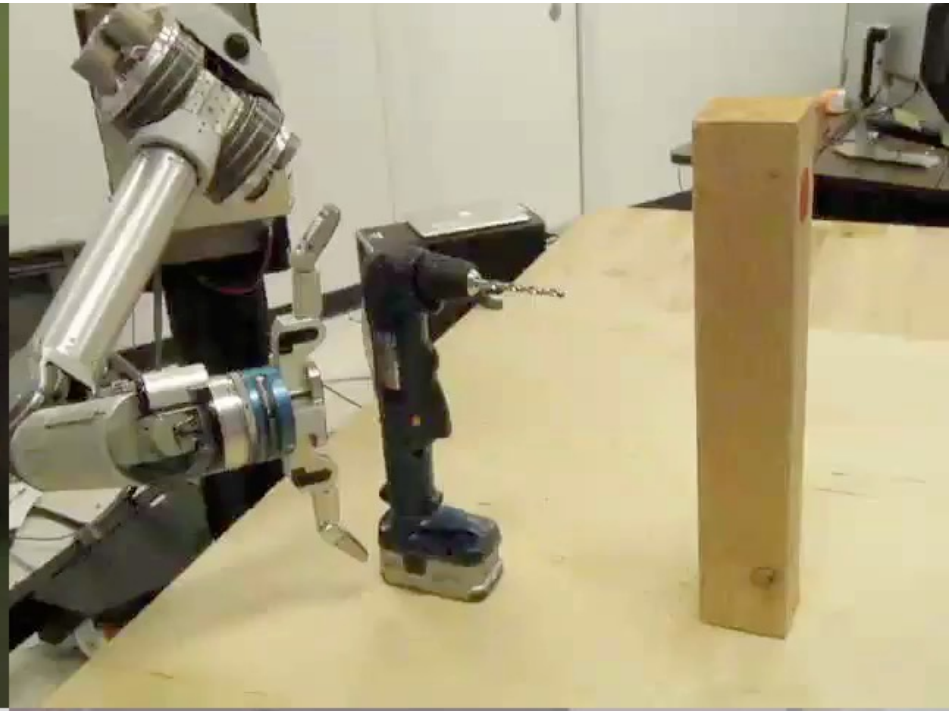
What Can AI Do?

Quiz: Which of the following can be done at present?

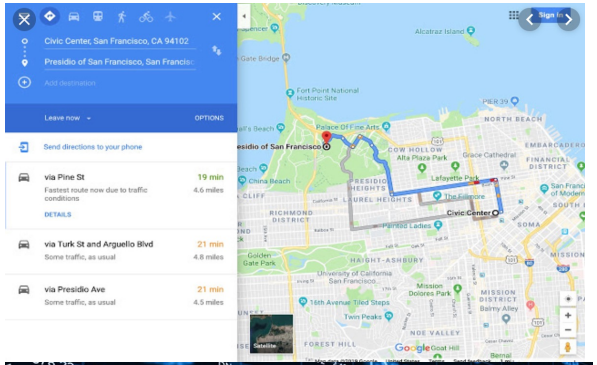
- ✓ Play a decent game of Bingo?
- ✓ Win against any human at chess?
- ✓ Win against the best humans at Go?
- ✓ Play a decent game of table tennis?
- ✓ Grab a particular cup and put it on a shelf?
- ✗ Unload any dishwasher in any home?
- ? Drive safely along the highway?
- ✗ Drive safely in Taipei?
- ✓ Buy a week's worth of groceries on the web?
- ✗ Perform a surgical operation?
- ? Discover and prove a new mathematical theorem?
- ✓ Translate spoken Chinese into spoken English in real time?

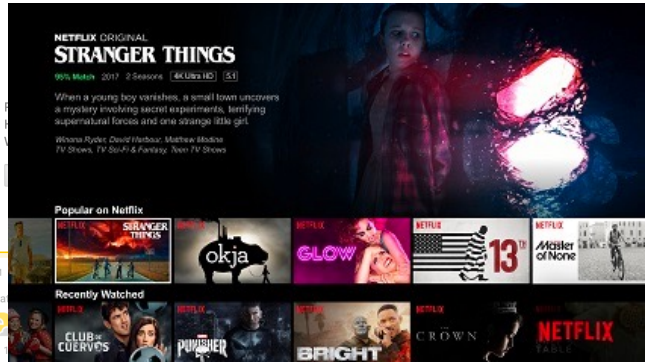


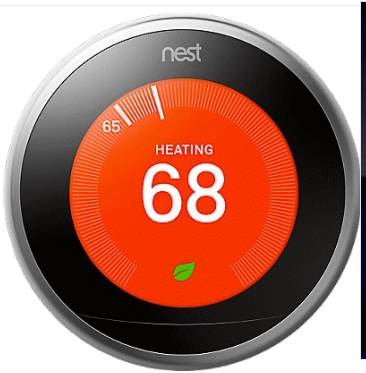
Robots



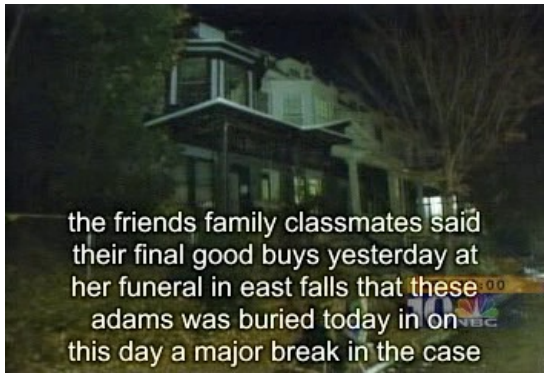
Tools for Predictions & Decisions







Natural Language



自動圖片描述
Auto-description


"Il est impossible aux journalistes de rentrer dans les régions tibétaines"

Bruno Philip, correspondant du "Monde" en Chine, estime que les journalistes de l'AFP qui ont été expulsés de la province tibétaine du Qinghai "n'étaient pas dans l'illégalité".

Les faits Le dalaï-lama dénonce l'"enfer" imposé au Tibet depuis sa fuite, en 1959
Vidéo Anniversaire de la rébellion


"It is impossible for journalists to enter Tibetan areas"

Philip Bruno, correspondent for "World" in China, said that journalists of the AFP who have been deported from the Tibetan province of Qinghai "were not illegal."

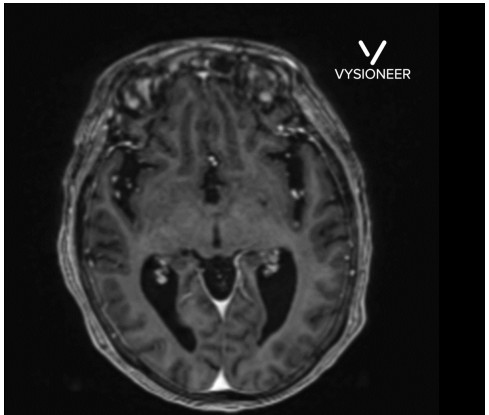
Facts The Dalai Lama denounces the "hell" imposed since he fled Tibet in 1959
Video Anniversary of the Tibetan rebellion: China on guard

機器翻譯
Machine Translation

Recent Applications of AI

一些新興的應用:

- 自動駕駛汽車
  - 運用機器學習算法進行路徑規劃和即時決策
- 醫療診斷與監控
  - AI協助醫生分析X光、MRI等醫學影像，提升診斷準確率
- 金融服務
  - AI分析大量數據，預測金融風險

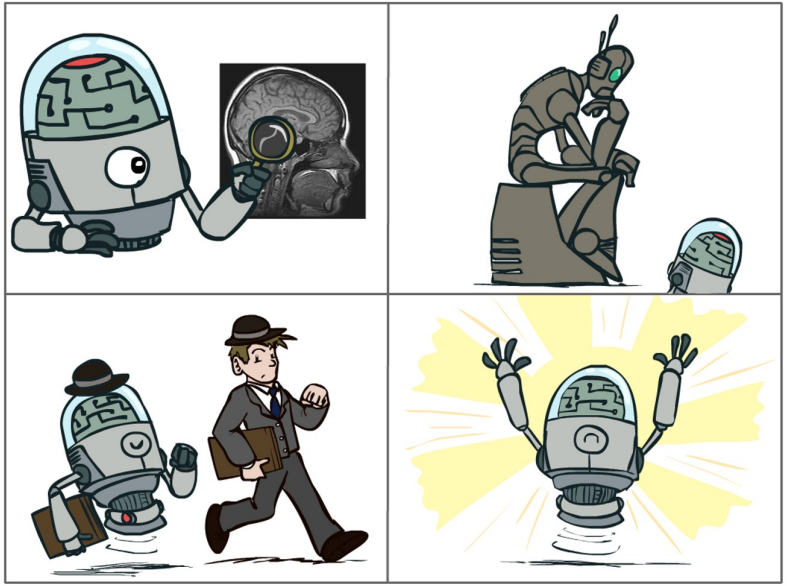


So, what is AI?

The science of making machines that:

Think like people

（像人一樣思考）



Think rationally

（理性思考）

Act like people

（像人一樣動作）

Act rationally

（理性動作）

Rational (理性的) Decisions

當這堂課講到「理性」這個詞時，定義如下：

- 理性：「最大化」達成「預先定義的」目標
- 只關注所做出的決策
  - (不涉及背後的思考過程)
- 目標以「結果的效用 (**utility** of outcomes)」 來表達
- 理性意味著最大化你的期望效用
  - **maximizing your expected utility**

The Earliest AI: Expert System

最早的專家系統:

- 專家系統是一種模擬人類專家決策過程的人工智慧系統
- 最簡單的專家系統中，使用 if-else 條件判斷來做出決策

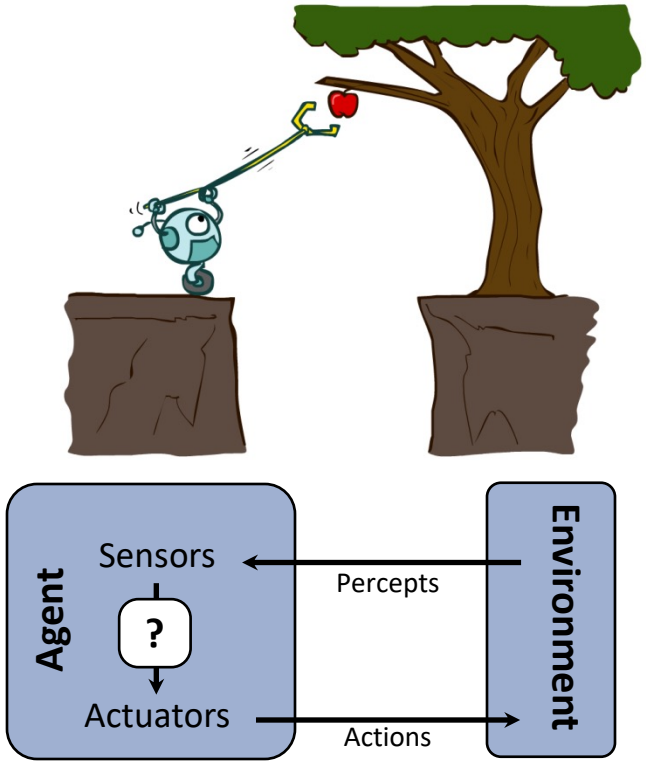
範例：診斷感冒或流感

```
如果（有高燒）且（持續時間 > 3 天）：
    結論 = 流感
否則：
    結論 = 普通感冒
```

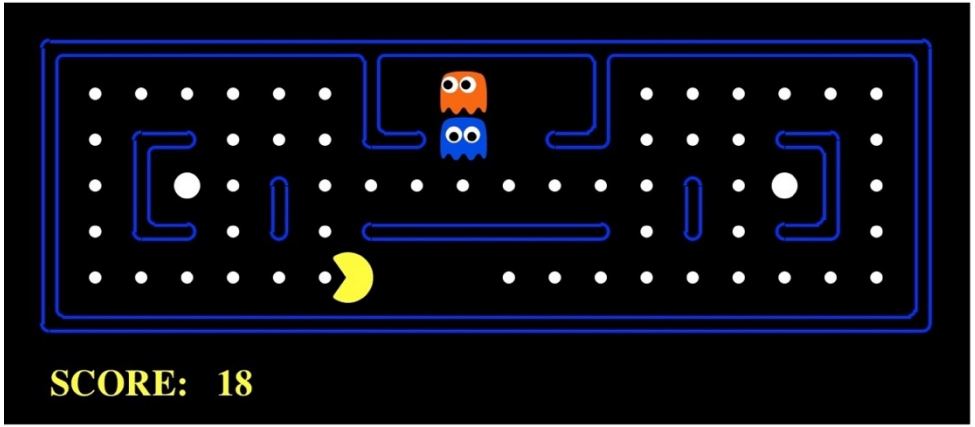
模擬醫生的決策過程!

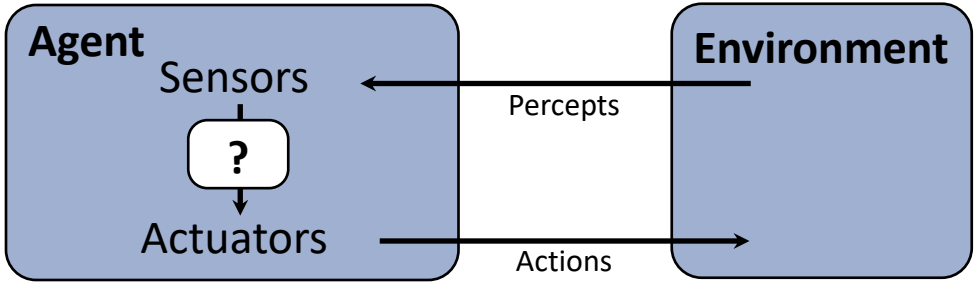
Designing Rational Agents

- An **agent** is an entity that *perceives* and *acts*.
- A **rational agent** selects actions that maximize its (expected) **utility**.
- Characteristics of the **percepts, environment,** and **action space** dictate techniques for selecting rational actions
- 一個 **agent** 是能夠感知 (*perceives*) 並執行行動 (*acts*) 的個體
- 理性 **agent** 會選擇能最大化期望效用 (expected utility) 的行動
- 感知、環境和行動空間 (**percepts, environment,** and **action space**) 的特性決定了選擇理性行動的方法

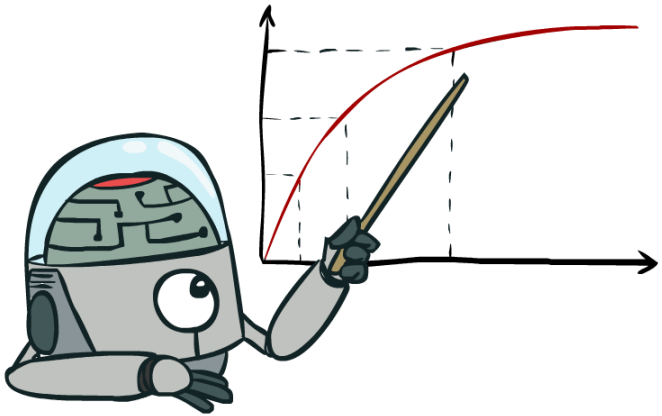


Pac-Man as an Agent



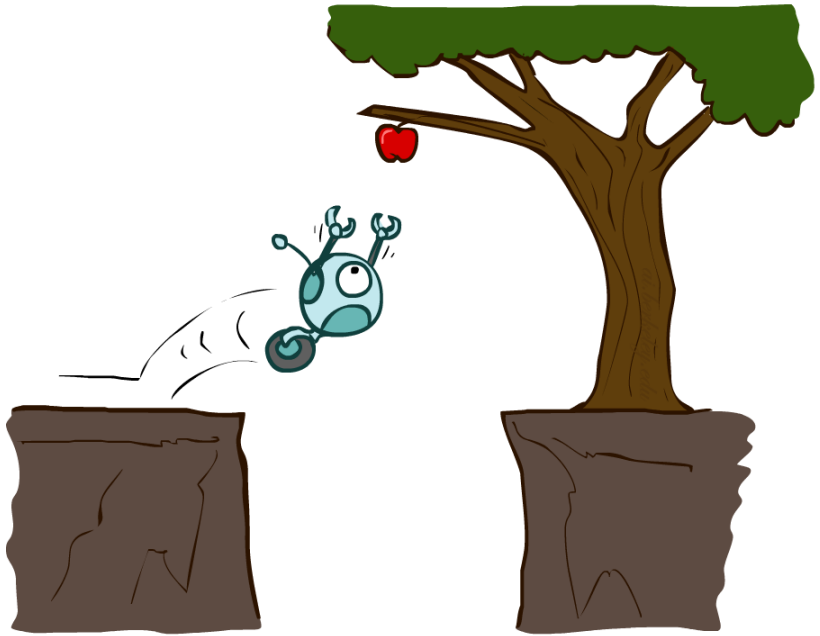


Our goal:
Design the Pacman Agent



Reflex Agents

- Reflex agents (反射型 agents):
  - 根據當前的感知 (current percept) 以及可能的記憶選擇行動
  - 會對所處的世界 (world) 進行建模
  - 不考慮目前行動的未來後果
  - Consider how the world IS
- *Can a reflex agent be rational?*
- 反射型 *agent* 可以是理性的嗎?

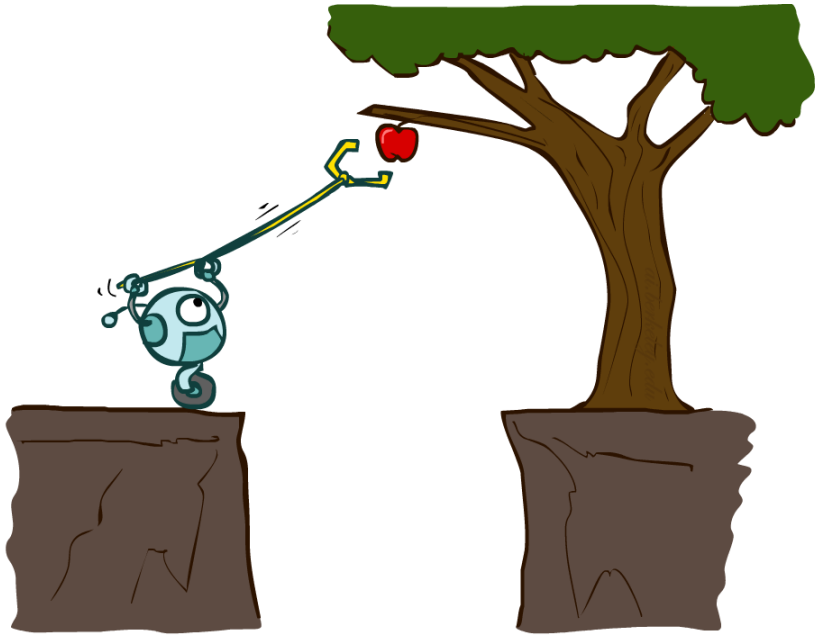


Video of Demo Reflex Optimal

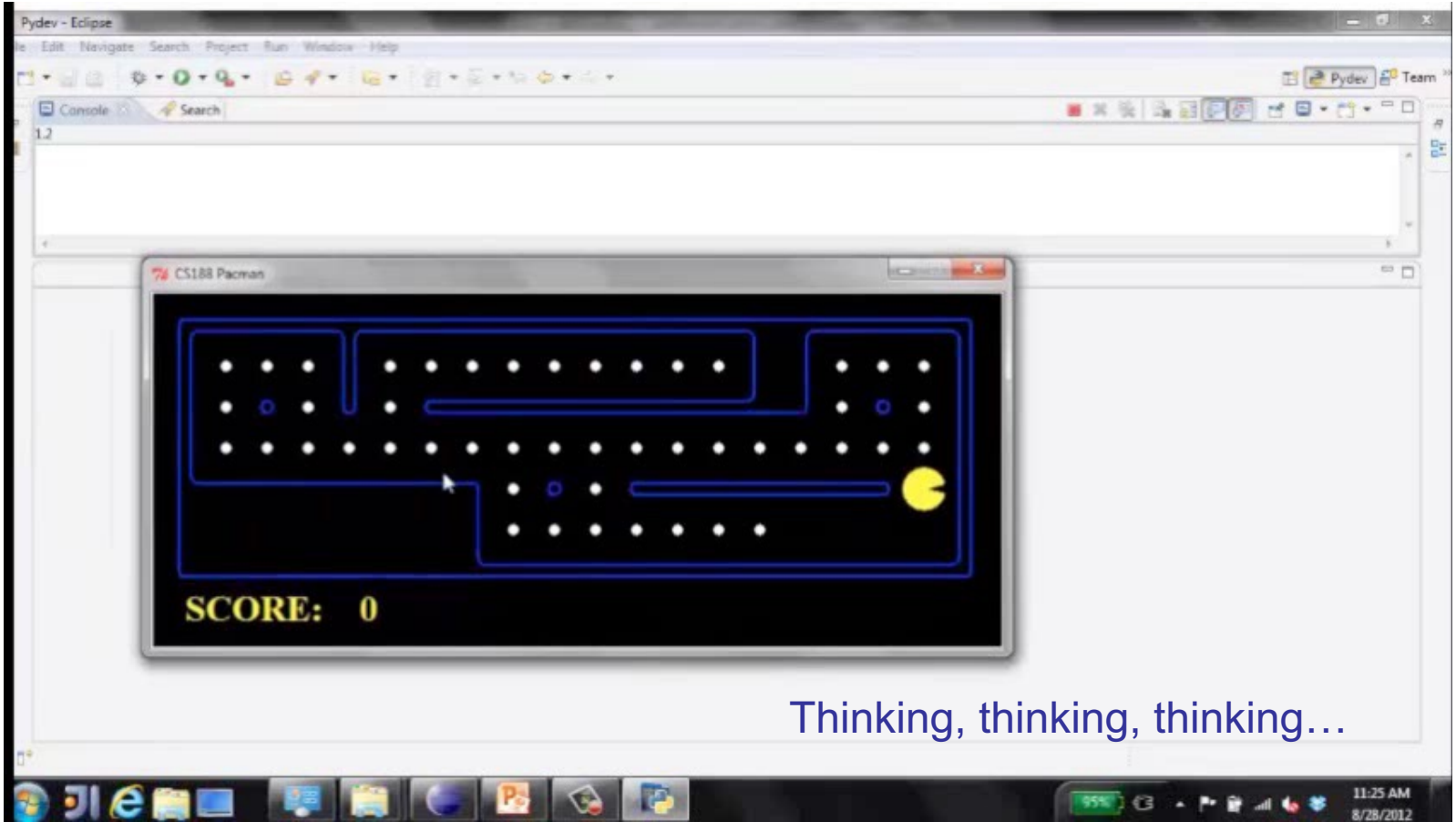
Video of Demo Reflex Odd

Planning Agents

- Planning agents (計畫型 agents) :
  - Ask “what if”
  - 根據行動 (act) 的「假設性後果」做決策
  - 必須具備一個模型 (model) 描述行動對世界的影響
  - 必須設定一個目標 (goal)，例如：是否所有的點點都被吃完?
  - 精神：Consider how the world WOULD BE



Video of Demo Planning

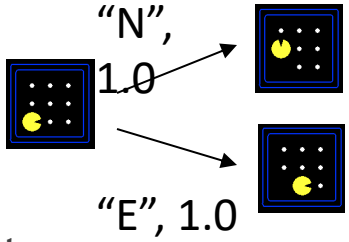


Search Problems

Search Problems

- 搜尋問題 (search problem) 有以下的元素：
  - A state space
    （狀態空間）

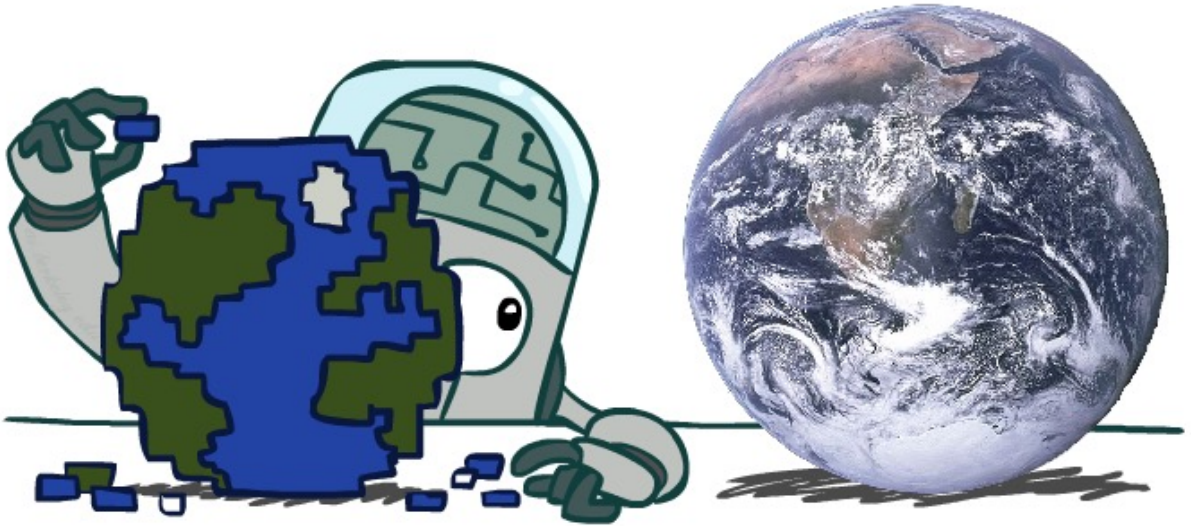
  - A successor function
    (with actions, costs)
    （繼承函數，
    包含動作與成本）



  - A start state and a goal test
    （起始狀態與目標測試）

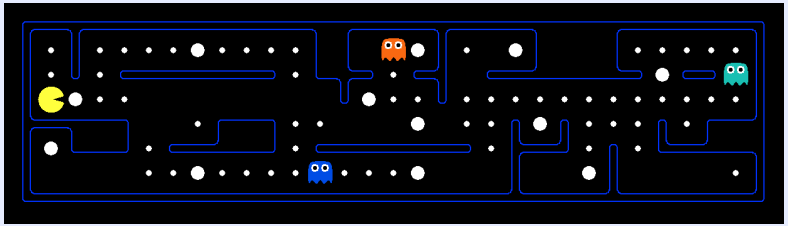
- 解答 (solution)：一系列計畫好的行動，將起始狀態轉化為目標狀態

Search Problems Are Models



What's in a State Space?

World state 包含所有環境中的細節



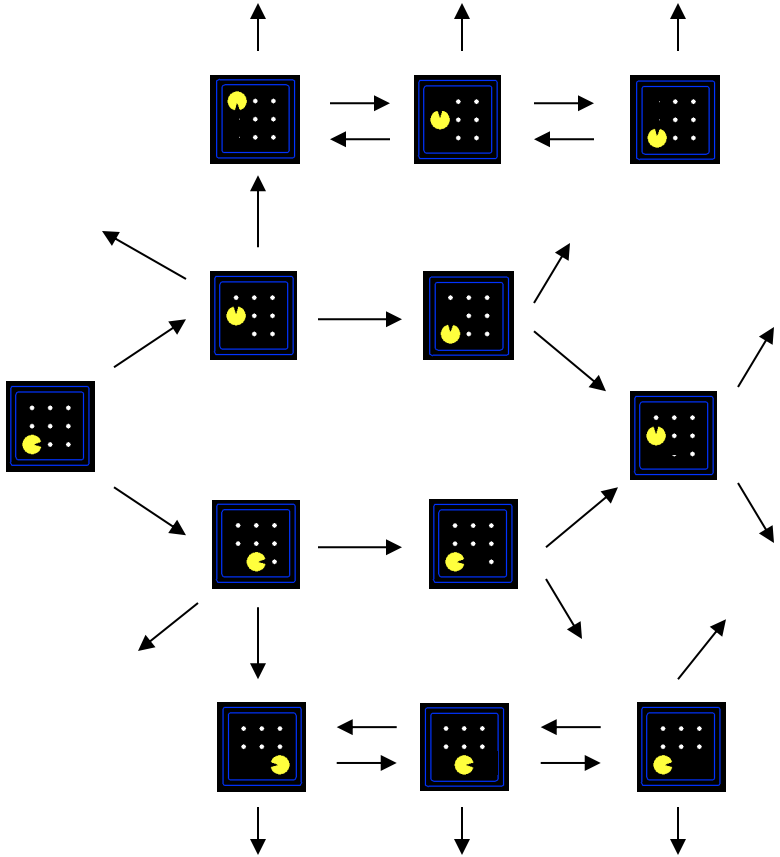
Search state 只保留 planning 所需的細節 (抽象化)

- Problem: Pathing (路徑規劃問題)
  - States: (x,y) 座標
  - Actions: NSEW
  - Successor: 只更新座標資訊
  - Goal test: is (x,y) = destination?

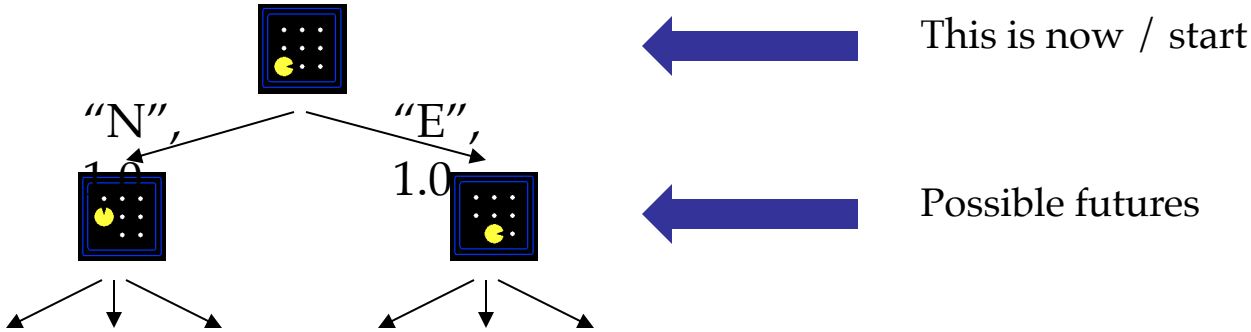
- Problem: Eat-All-Dots
  - States: {(x,y), dot booleans}
  - Actions: NSEW
  - Successor: 更新座標資訊以及 dot boolean
  - Goal test: 所有的點是否已被吃完?

State Space Graphs

- 狀態空間圖（State space graph）：一種用來表示搜尋問題的數學模型
  - Nodes (節點) are abstracted world configurations
  - Arcs (邊) represent successors (動作的結果)
  - Goal test 是一組 goal nodes 的集合
- 在狀態空間圖中，每個狀態只會出現一次!
- 我們通常無法在記憶體中構建完整的狀態空間圖（因為它太大了…）
- 但這是一個有用的概念



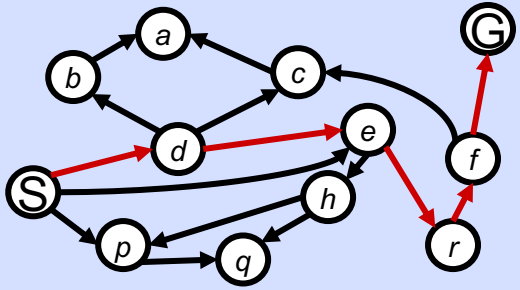
Search Trees



- 搜尋樹:
  - A "what if" tree of plans and their outcomes
  - 起始狀態 (start state) 是根節點 (root node)
  - 子節點 (children) 對應到後繼節點 (successors)
  - Nodes show states, but correspond to PLANS that achieve those states
  - 對於大多數問題來說，我們實際上無法構建完整的樹

State Space Graphs vs. Search Trees

State Space Graph

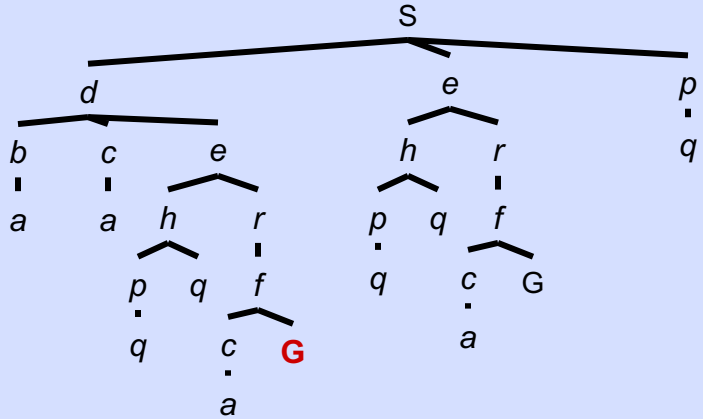


S: Start
G: Goal

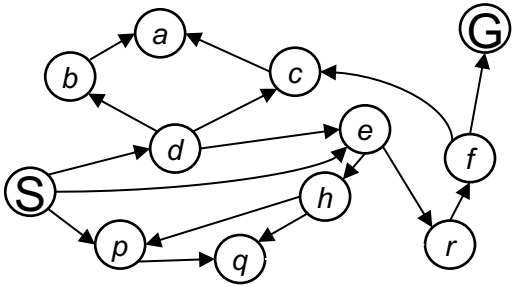
每個節點（NODE）在搜尋樹中代表狀態空間圖中的一整條路徑（PATH）

We construct both on demand – and we construct as little as possible.

Search Tree

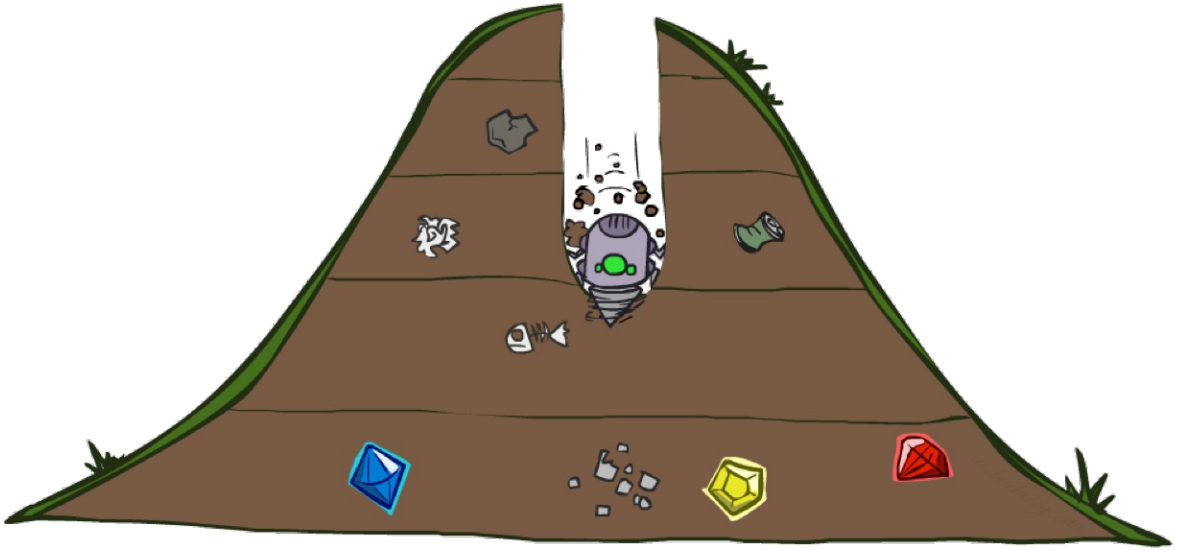


Example: Tree Search



Depth-First Search

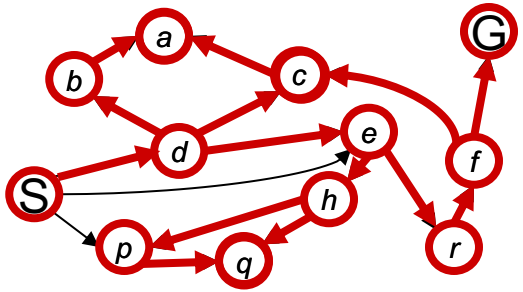
（深度優先搜尋）



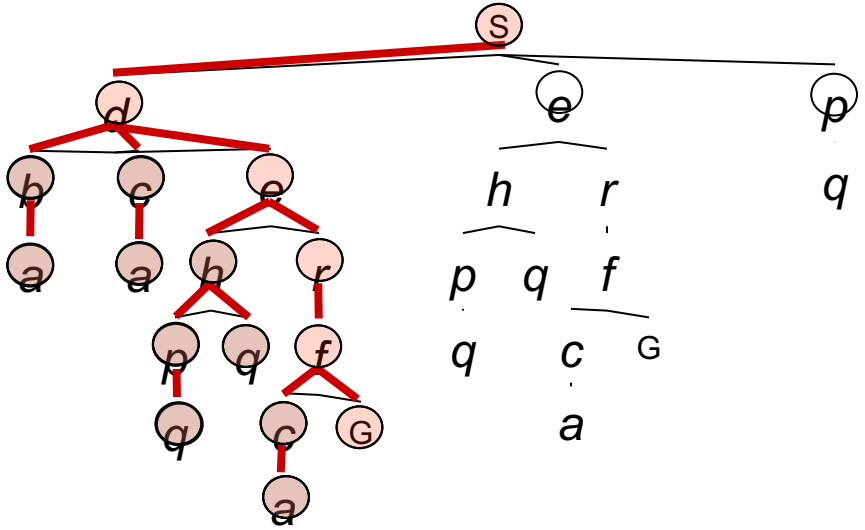
Depth-First Search

策略：優先展開最深的節點

Implementation:
LIFO stack

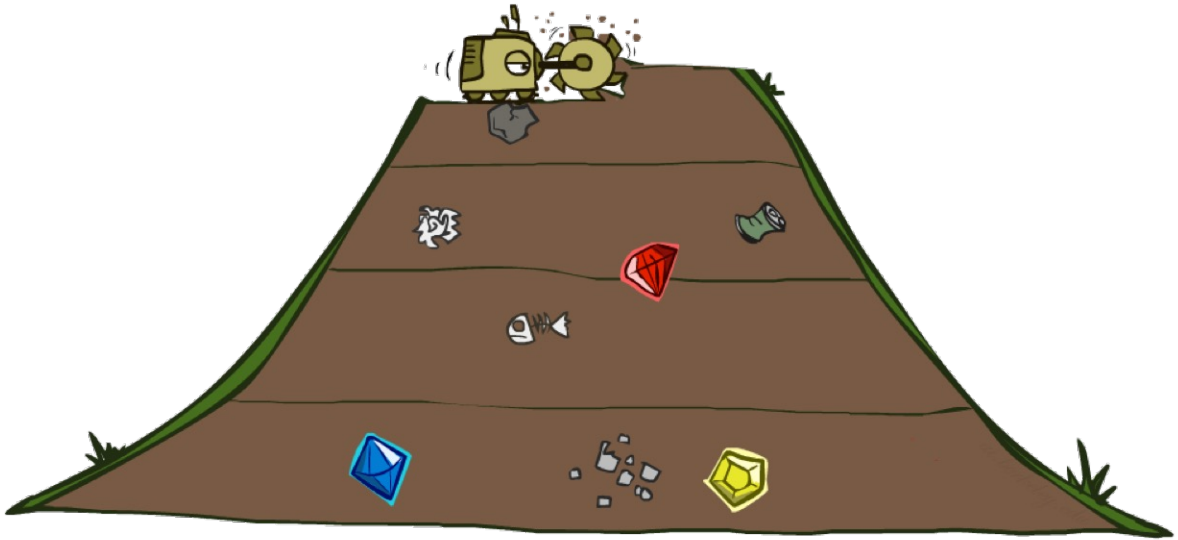


做 DFS 的 Agent 是一種 Planning Agent



Breadth-First Search

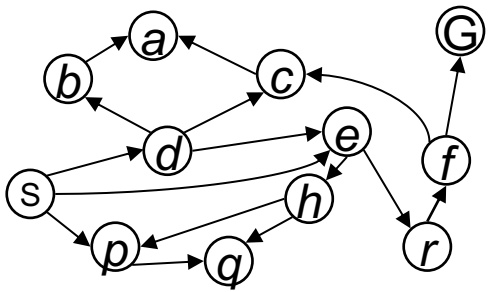
（廣度優先搜尋）

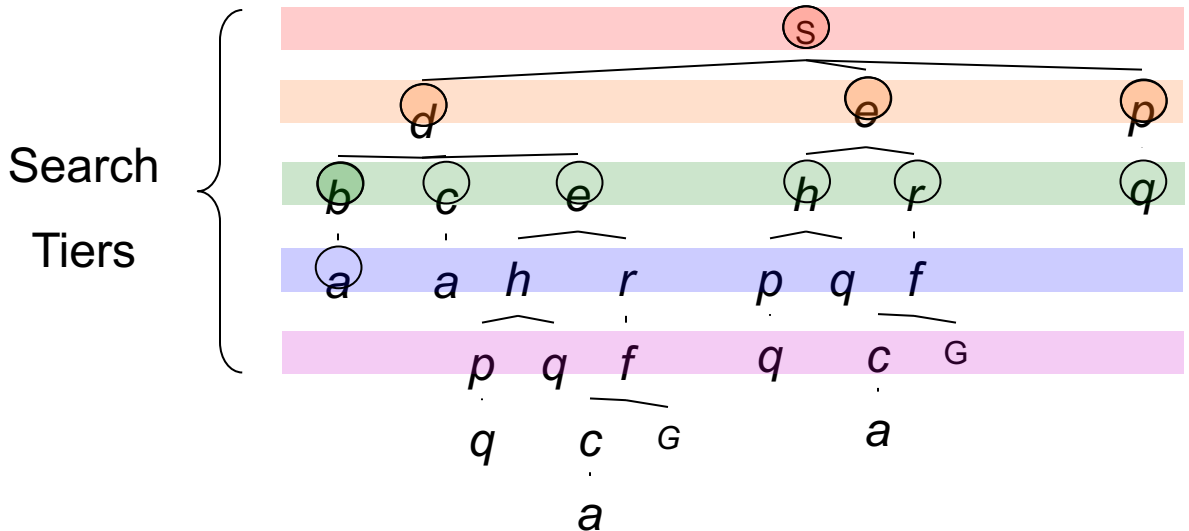


Breadth-First Search

策略：優先展開
最淺的節點

Implementation:
FIFO queue





Other Searching Methods

其他進階的搜尋演算法：

- Uniform-Cost Search
- Greedy Search
- A* Search
- **You need to understand “Heuristics” if you want to know the last two methods.**

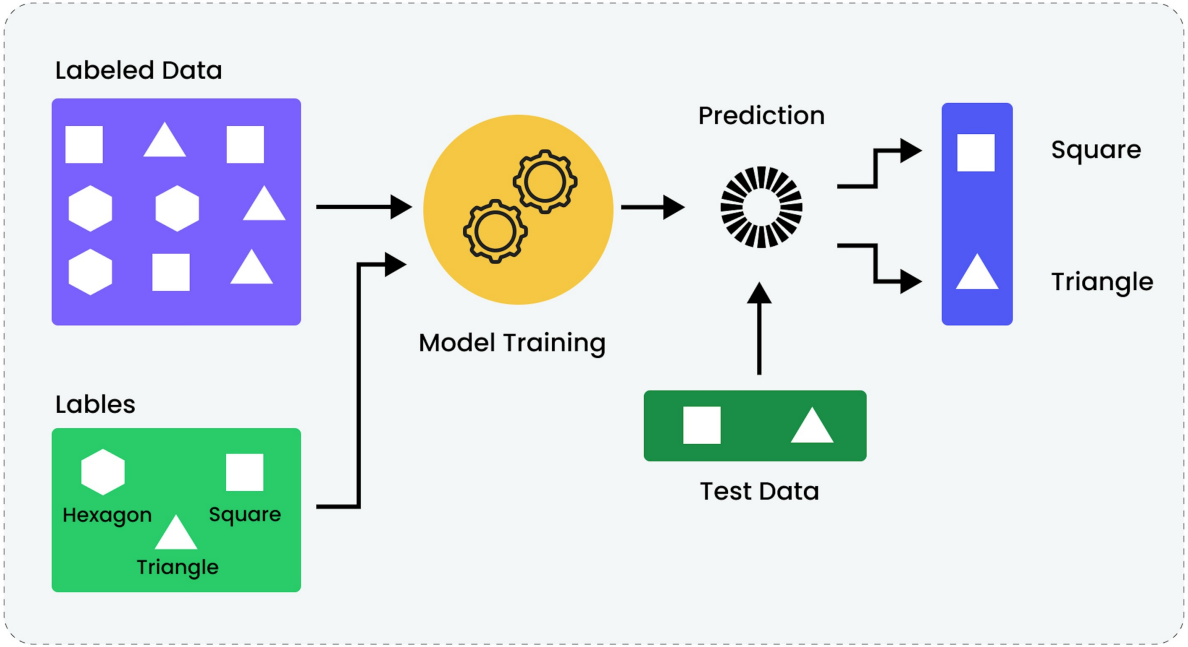
Supervised Learning

- **什麼是監督式學習?**
  - 透過已知的「輸入」和對應的「標籤」來訓練模型
  - 目標是讓模型學會從過去的數據中預測或分類新數據

1. **訓練資料**：包括已標註的輸入數據和對應的真實標籤（如圖片與其標籤）
2. **模型學習**：模型學會從輸入數據中抽取模式，並預測對應的標籤
3. **目標**：最小化預測與實際標籤之間的誤差

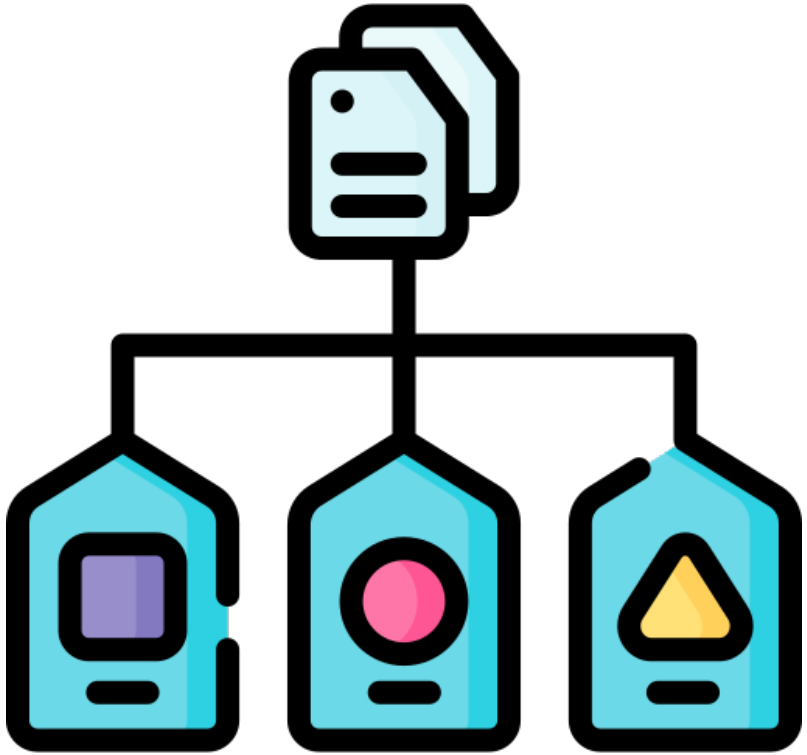
Supervised Learning

監督式學習: 需要 data，也需要對應的 label

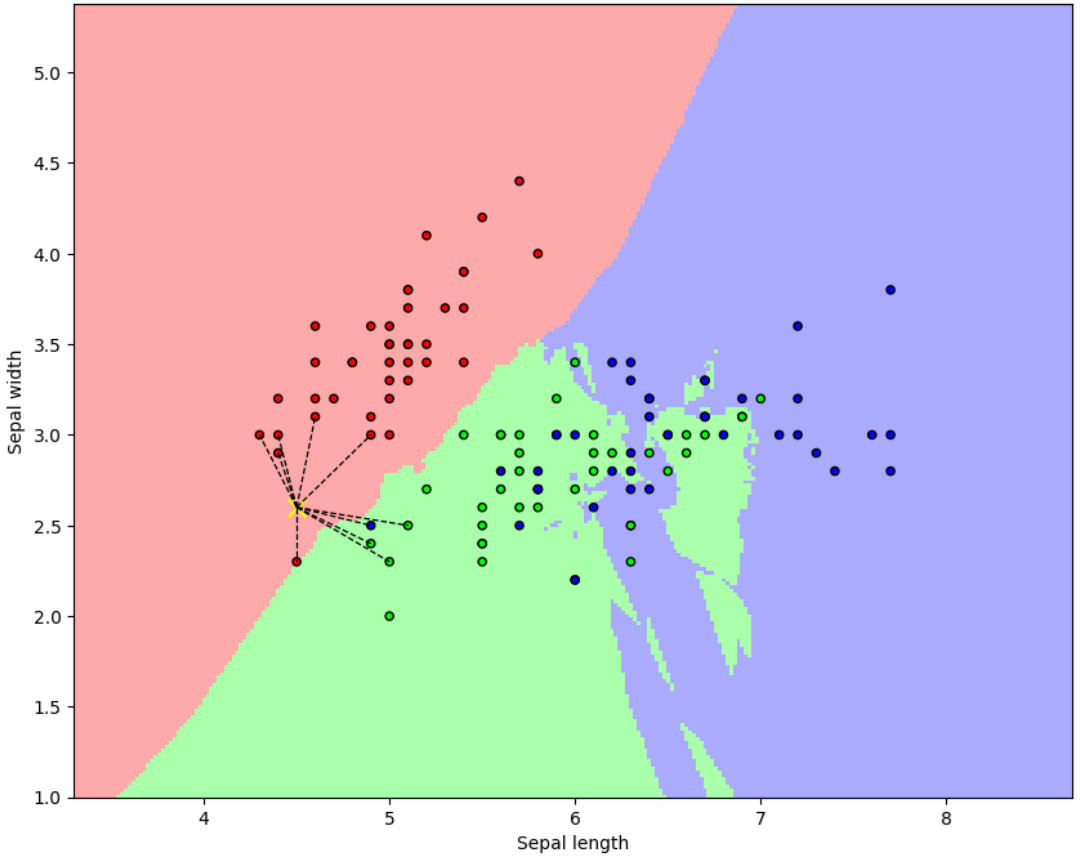


來源: "Supervised Machine Learning," *Tutorials Point*, https://www.tutorialspoint.com

Classification by kNN



k Nearest Neighbor (kNN)



k Nearest Neighbor (kNN)

Training data: $(\boldsymbol{x}_i, y_i)$

- y_i 是標籤
- $\boldsymbol{x}_i$ 是長度為 m 的特徵向量。可能包括多個特徵

標籤 y_i 的兩種情況：

- 1. 分類問題:
  - 如果 y_i 是離散值（例如 $y_i \in \{1,2,3\}$），表示這是分類任務
  - kNN 將根據鄰近的 k 個樣本的標籤進行多數投票，選出一個最可能的類別作為預測結果
- 2. 回歸問題:
  - 如果 y_i 是連續值（例如 $y_i \in R$），表示這是回歸任務
  - kNN 將計算鄰近 k 個樣本標籤的平均值，作為預測結果

How to Measure Distance?

- L2 Norm (Euclidian distance):

$$\|x_i - x_j\| = \sqrt{\sum_{q=1}^{m} (x_{i,q} - x_{j,q})^2}$$

- L1 Norm:

$$\|x_i - x_j\|_1 = \sum_{q=1}^{m} |x_{i,q} - x_{j,q}|$$

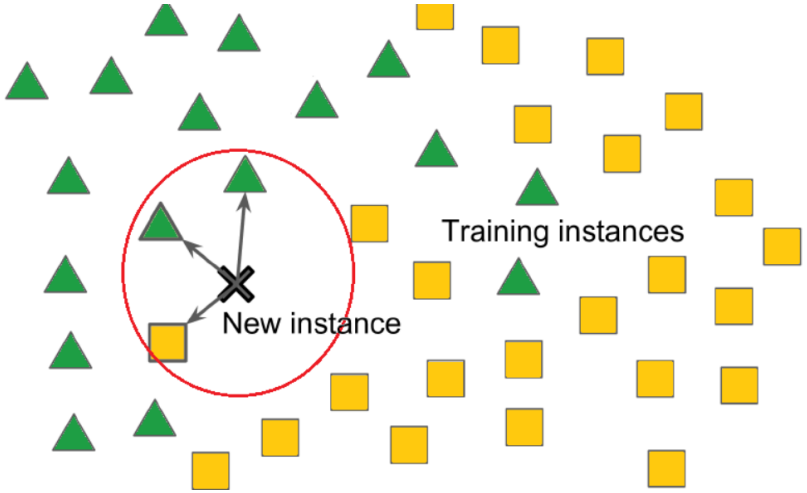
k Nearest Neighbor (kNN)

Training data: $(\boldsymbol{x}_i, y_i)$

- y_i 是標籤
- $\boldsymbol{x}_i$ 是長度為 m 的特徵向量。可能包括多個特徵

標籤 y_i 的兩種情況：

1. 分類問題
2. 回歸問題



來源: Ahmed Raafat, "K-Nearest Neighbor (KNN) Explained," *Machine Learning Mastery*, September 9, 2022. https://machinelearningmastery.com

k Nearest Neighbor 分類問題

分類問題：

- 如果 y_i 是離散值（例如 $y_i \in \{1,2,3\}$），表示這是分類任務
- kNN 將根據鄰近的 k 個樣本的標籤進行多數投票，選出一個最可能的類別作為預測結果

範例：

- $k = 3$，最近的三個樣本標籤為 [1,2,2]
- 多數投票結果：預測類別為 2

k Nearest Neighbor 回歸問題

回歸問題：

- 如果 y_i 是連續值（例如 $y_i \in R$ ），表示這是回歸任務
- kNN 將計算鄰近 k 個樣本標籤的平均值，作為預測結果

範例：

- $k = 3$，最近的三個樣本標籤為 [3.2, 4.1, 3.8]
- 平均值：$\frac{3.2+4.1+3.8}{3} = 3.7$
- 預測結果為 3.7

Measuring Prediction Performance

監督式學習訓練流程:

① 隨機排列數據，並分成**訓練集**（90%）和**測試集**（10%）

② 訓練集可再分為**子訓練集**（80%）和**調參集**（10%）

③ 若模型沒有超參數，則直接用訓練集訓練，並用測試集測試

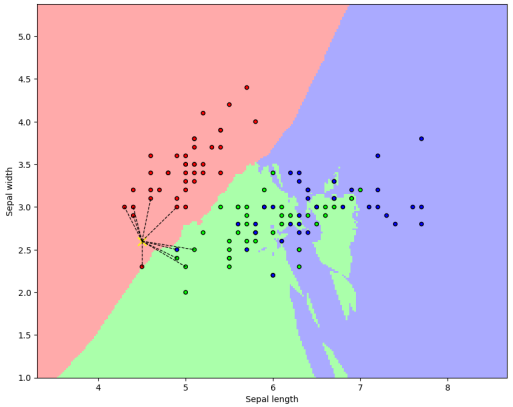
④ 若模型有超參數，使用子訓練集來訓練，調整超參數後，固定超參數並重新訓練

⑤ 最後，使用測試集測試最終模型的效果

Discussions on kNN

1.k 在這裡是個「超參數」(Hyperparameter)：

- 使用不同的 k，可能會帶來不同的模型準確率!



2. kNN 是一個「不需要訓練」的演算法

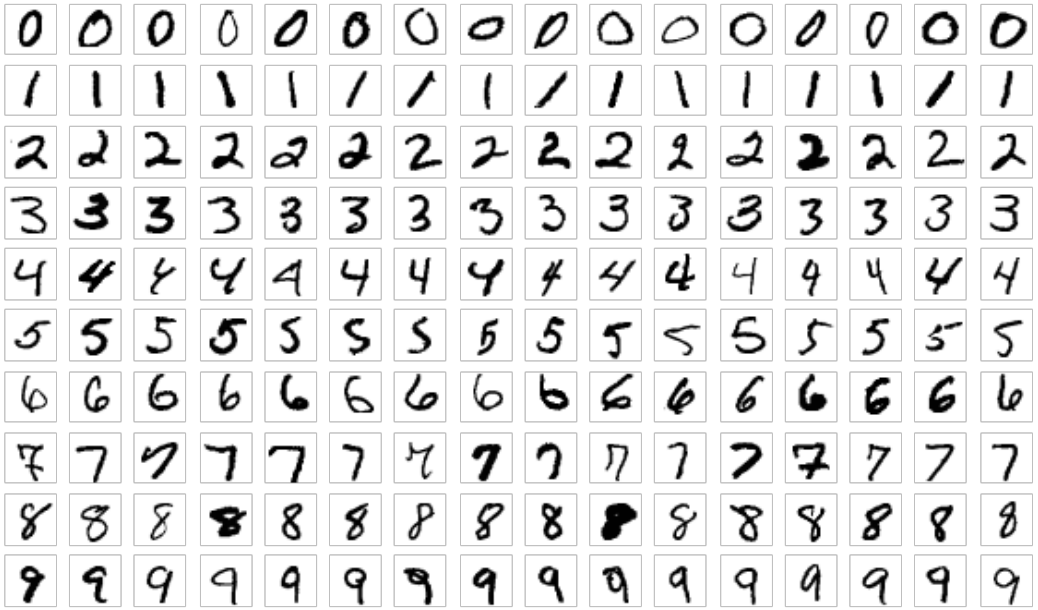
- 簡單易懂

3. 代價：kNN 的計算複雜度很高

- 找出前 k 個最靠近的鄰居，這件事情是在計算上很複雜的

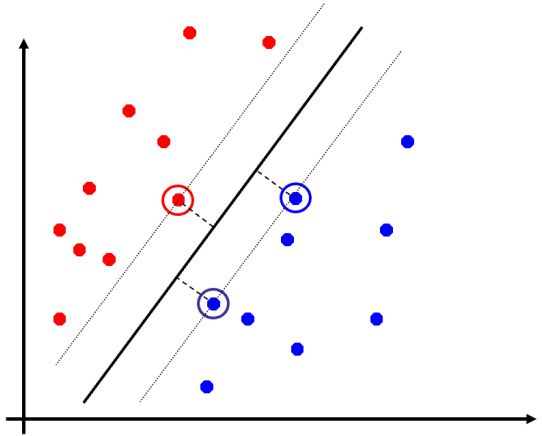
kNN 實作手寫數字分類

- Google Colab link: https://shorturl.at/AfaKK
- 請在打開檔案後，登入自己的 Google 帳號

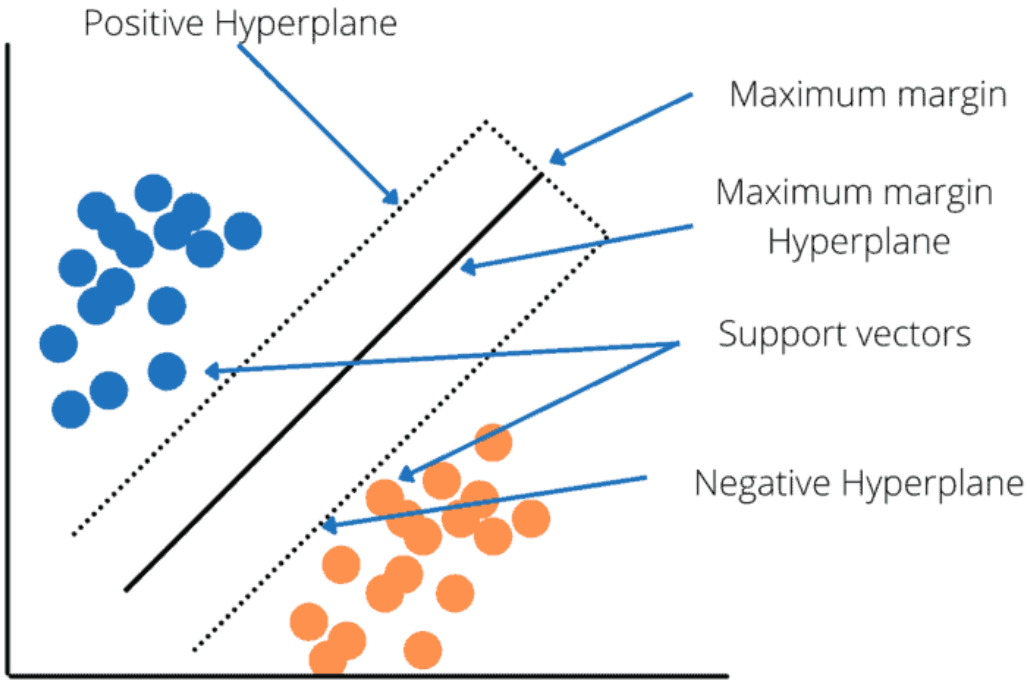


Another Tool：Support Vector Machines

- 最大化邊界（Maximizing the margin）
- 只有支持向量（Support Vectors）對於模型來說是重要的。
- 其他訓練樣本可以忽略
- 支持向量機（Support Vector Machines, SVMs）找到具有最大邊界的分隔線



Example: SVMs



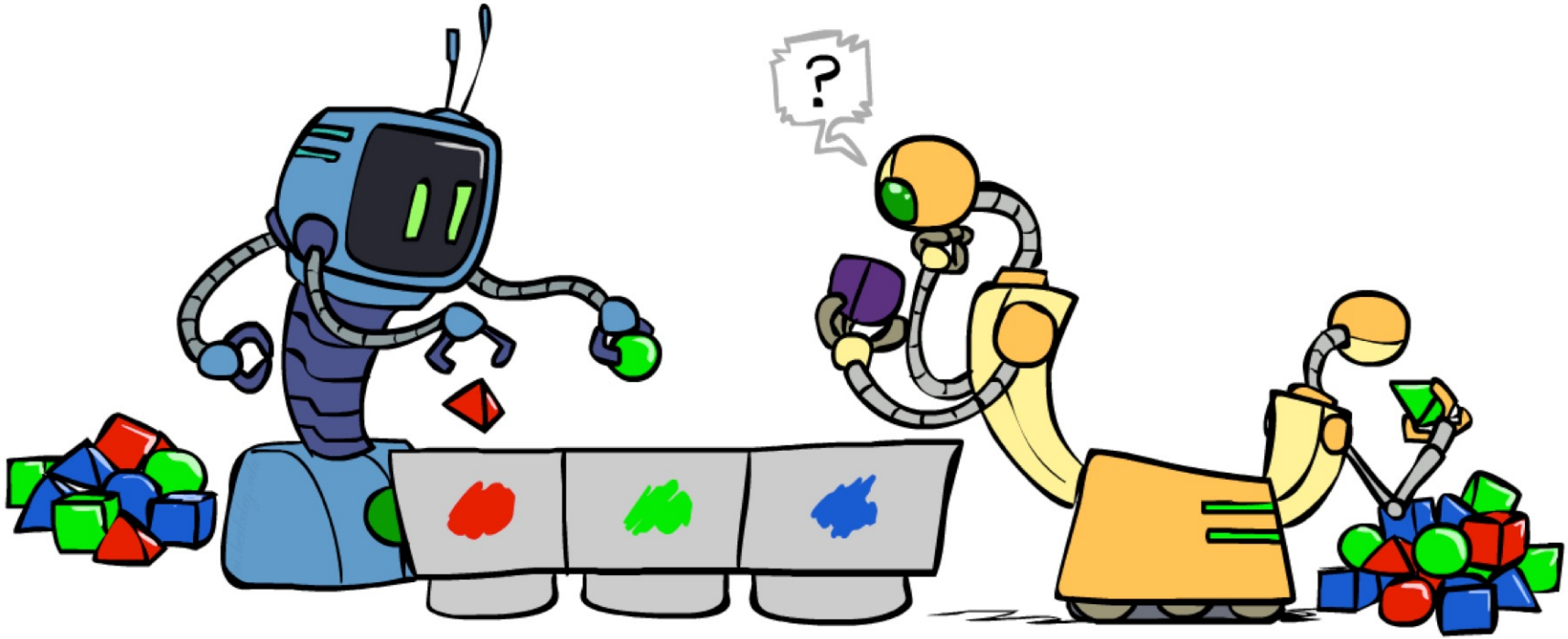
來源: Bashir Alam, "SVM Python – Easy Implementation of SVM algorithm," *Hands-On.Cloud*, 01/15/2022. https://hands-on.cloud

Clustering

分群系統:

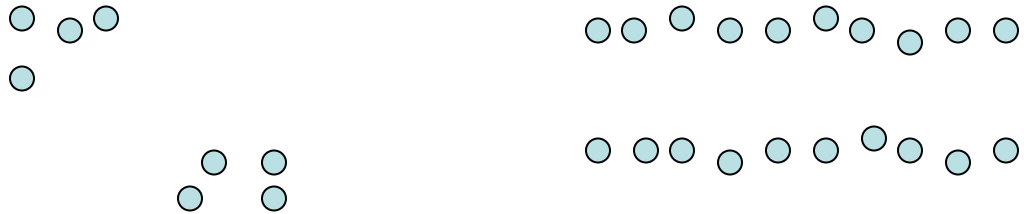
- 非監督式學習
- 在未標記的資料中尋找特徵
  - 對電子郵件或搜尋結果進行分組
  - 依照顧客的特性進行分群
  - 檢測程式的異常行為
- 適用於「不知道具體要尋找什麼」的情況
  - 需要 data，但不需要 label
- 分群的結果常常很亂

Clustering



Clustering

- 基本概念：將相似的 instance 分在一起
- 以 2D 座標點為例：



- “similar” 的意思是什麼?
  - One option: squared Euclidean distance

$$\text{dist}(x, y) = (x - y)^{\mathsf{T}}(x - y) = \sum_i (x_i - y_i)^2$$

Review: How to Measure Distance?

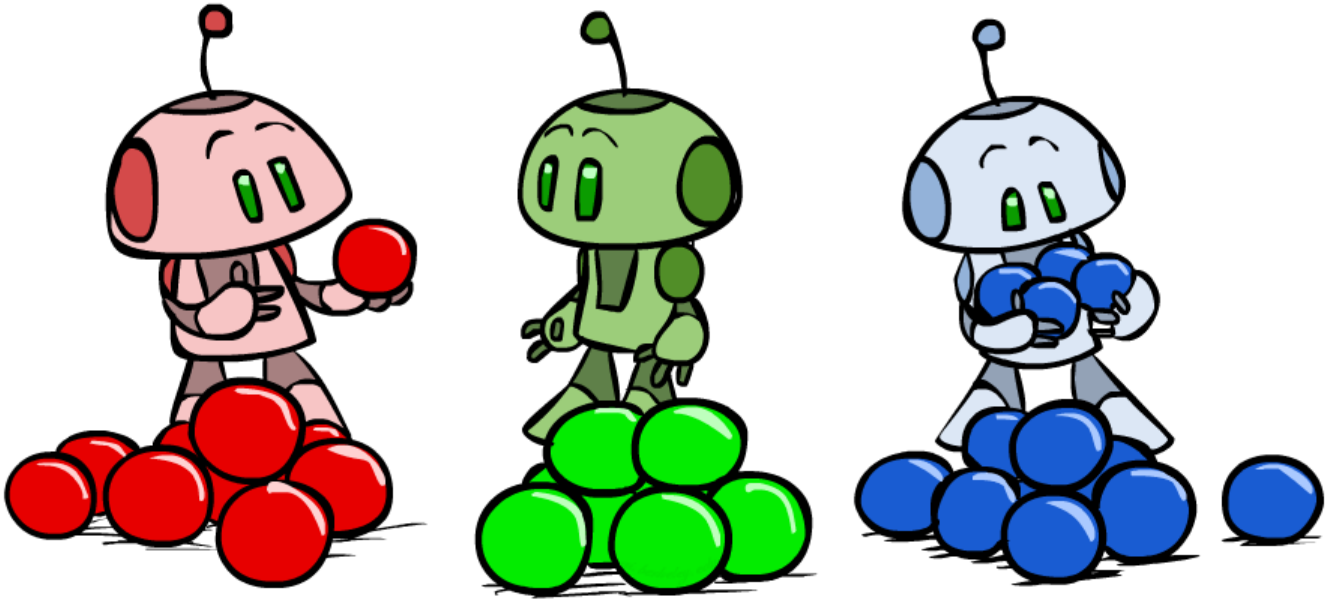
- L2 Norm (Euclidian distance):

$$\|x_i - x_j\| = \sqrt{\sum_{q=1}^{m} (x_{i,q} - x_{j,q})^2}$$

- L1 Norm:

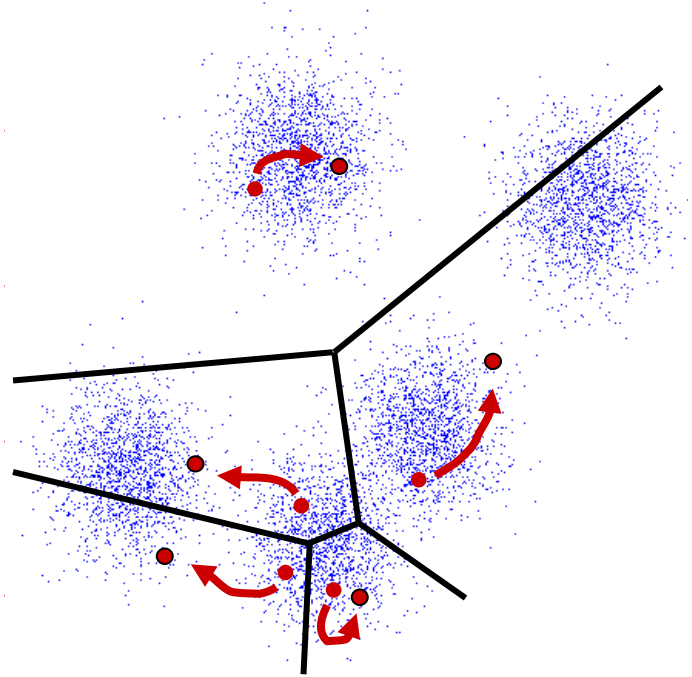
$$\|x_i - x_j\|_1 = \sum_{q=1}^{m} |x_{i,q} - x_{j,q}|$$

K-Means

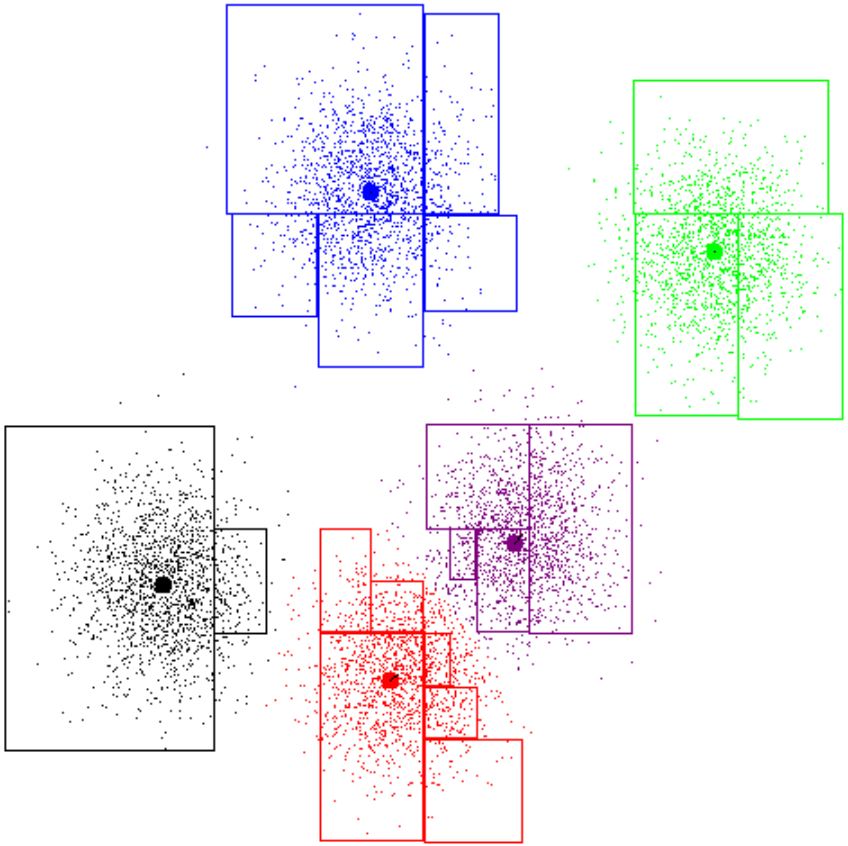


K-Means

- K-Means 是一個迭代的分群演算法：
  - 隨機選擇 K 個點作為群中心
  - 重複以下步驟:
    - 將 data instances 分配到距離最近的群中心所屬的群
    - 更新每個群的中心為該群內且偶點的平均值
  - 當所有點的分配不再改變時 -> 停止



K-Means Example



K-Means 實作手寫數字分群

- Google Colab link: https://shorturl.at/Ih9KE
- 請在打開檔案後，登入自己的 Google 帳號