內容

[FNN（Feedforward Neural Network） 1](#_Toc166418068)

[RNN（Recurrent Neural Network） 2](#_Toc166418069)

[CNN（Convolutional Neural Network） 2](#_Toc166418070)

[FNN RNN CNN三者之間的差別 3](#_Toc166418071)

[激活函數（Activation Function） 4](#_Toc166418072)

[Entropy 與 Cross Entropy 的差別 5](#_Toc166418073)

[Entropy（熵） 5](#_Toc166418074)

[Cross Entropy（交叉熵） 6](#_Toc166418075)

[DFS 7](#_Toc166418076)

[BFS 8](#_Toc166418077)

[DFS VS. BFS 8](#_Toc166418078)

[🕰️ 最小最大演算法的時間複雜度 9](#_Toc166418079)

[⚖️ 最小最大演算法的局限性 9](#_Toc166418080)

[✂️ α-β剪枝法 10](#_Toc166418081)

[🔄 最佳移動策略 10](#_Toc166418082)

[➕ 以极大和极小的策略进行移動 11](#_Toc166418083)

[🌳 遊戲樹的分析 12](#_Toc166418084)

## FNN（Feedforward Neural Network）

- 中文描述：FNN（前饋神經網路）是一種**最基本的神經網路模型**，其中**訊息只會向前流動，不會形成循環或回饋。**它由多個層組成，每一層包含多個神經元（節點），**相鄰層之間的神經元之間存在權重**。每個神經元的輸出是前一層神經元的加權和，**經過激活函數處理後得到**。

- 英文描述：A Feedforward Neural Network (FNN) is a basic neural network model where information flows only in one direction, without cycles or feedback loops. It consists of multiple layers, each containing multiple neurons (nodes), with weighted connections between adjacent layers. The output of each neuron is the weighted sum of inputs from the previous layer, followed by an activation function.

## RNN（Recurrent Neural Network）

- 中文描述：RNN（循環神經網路）是一種**具有循環結構的神經網路**，能夠處理序列數據或時間序列數據。**RNN的特點是在每個時間步都會將當前的輸入與上一個時間步的隱藏狀態進行計算**，**產生當前時間步的輸出**。RNN**具有記憶功能**，可以**捕捉序列數據中的時間依賴性**。

- 英文描述：A Recurrent Neural Network (RNN) is a type of neural network with a loop structure that allows it to handle sequential or time-series data. In RNN, the current input at each time step is processed along with the hidden state from the previous time step to produce the output at the current time step. RNNs have the ability to capture temporal dependencies in sequential data.

## CNN（Convolutional Neural Network）

- 中文描述：CNN（卷積神經網路）是一種**專門用於處理圖像數據的神經網路模型**。它主要**包含卷積層（Convolutional Layer）、池化層（Pooling Layer）和全連接層（Fully Connected Layer）**。CNN通過使用卷積操作來提取圖像的特徵，並通過池化操作進行特徵降維，最終通過全連接層進行分類或回歸。

- 英文描述：A Convolutional Neural Network (CNN) is a type of neural network designed specifically for processing image data. It typically consists of convolutional layers, pooling layers, and fully connected layers. CNNs use convolution operations to extract features from images and pooling operations to reduce the dimensionality of the features, ultimately performing classification or regression tasks using fully connected layers.

## FNN RNN CNN三者之間的差別

**結構差異：**

FNN：訊息只向前傳播，沒有循環結構。

RNN：具有循環結構，能夠處理序列或時間序列數據。

CNN：主要用於處理圖像數據，包含卷積、池化和全連接層。

**Structural Differences:**

**Structural Differences:**

* **FNN (Feedforward Neural Network):** Information flows only forward without any cyclic structure.
* **RNN (Recurrent Neural Network):** Contains cyclic structure and is capable of handling sequential or time-series data.
* **CNN (Convolutional Neural Network):** Primarily used for processing image data and includes convolutional layers, pooling layers, and fully connected layers.

**應用領域差異：**

FNN：適用於一般的分類或回歸任務。

RNN：適用於序列建模，如自然語言處理（NLP）、時間序列預測等。

CNN：主要用於計算機視覺任務，如圖像分類、物體檢測等。

**Application Domain Differences:**

* **FNN:** Suitable for general classification or regression tasks.
* **RNN:** Suitable for sequence modeling tasks such as natural language processing (NLP) and time-series prediction.
* **CNN:** Mainly used for computer vision tasks like image classification and object detection.

**特點差異：**

FNN：較簡單，無法處理序列數據的時間依賴性。

RNN：具有記憶能力，能夠捕捉序列數據中的時間依賴性。

CNN：專門用於處理圖像數據，具有平移不變性（Translation Invariance）。

**Characteristic Differences:**

* **FNN:** Relatively simple and unable to handle time dependencies in sequential data.
* **RNN:** Possesses memory capabilities and can capture temporal dependencies in sequential data.
* **CNN:** Specifically designed for processing image data with translation invariance properties.

## 激活函數（Activation Function）

神經網路中的一種重要組件，它通常**被應用於神經元的輸出，用於引入非線性特性**，並決定神經元是否應該被激活（fired）。

在神經網路中，**每個神經元接收來自前一層的加權輸入（包括偏差項），然後通過激活函數計算出神經元的輸出**。激活函數**將線性組合的輸入轉換成非線性輸出**，這對於神經網路的學習和表示能力至關重要。

An activation function is a critical component in neural networks applied to determine the output of a neuron based on its input. It introduces non-linearity into the network, allowing it to model complex relationships and make decisions about whether a neuron should be activated (fired).

In neural networks, each neuron receives weighted inputs from the previous layer, including a bias term, and computes its output using an activation function. The activation function transforms the linear combination of inputs into a non-linear output, which is essential for the learning and representational capabilities of neural networks.

一張含有 文字, 螢幕擷取畫面, 字型, 數字 的圖片

自動產生的描述**Sigmoid Function** (Logistic Function): It compresses the input into the range of (0,1)(0,1) and is often used in the final layer of binary classification tasks.

**Tanh Function** (Hyperbolic Tangent Function): It compresses the input into the range of (−1,1)(−1,1) with a centering property.

**ReLU Function** (Rectified Linear Unit Function): It outputs the input if it's positive, otherwise outputs zero, commonly used in hidden layers.

**Softmax Function**:It converts a vector into a probability distribution, often used in the final layer of multi-class classification tasks

## Entropy 與 Cross Entropy 的差別

中文描述：

### Entropy（熵）

是訊息理論中衡量**隨機變量不確定度或混亂程度的度量**。在機器學習中，熵用於**衡量一個概率分佈的不確定性**，計算方式為將每個事件的概率乘以其自然對數，然後**求和取負值。熵越高，概率分佈的不確定性越大**。

1. Entropy:

- Definition: Entropy is a measure of uncertainty or disorder of a random variable in information theory. In machine learning, entropy is used to quantify the uncertainty of a probability distribution. It is calculated by multiplying each event's probability by its natural logarithm, then summing the results and taking the negative value. Higher entropy indicates greater uncertainty in the probability distribution.

### Cross Entropy（交叉熵）

是一種**衡量兩個概率分佈之間的差異或不匹配程度的指標**。在機器學習中，交叉熵**常用於衡量模型預測的概率分佈與真實概率分佈之間的距離**。交叉熵的計算方式是**將真實概率分佈的每個事件的概率乘以其自然對數，然後與模型預測的概率分佈相乘，最後求和取負值**。

英文描述：

2. Cross Entropy:

- Definition: Cross Entropy is a metric that measures the difference or mismatch between two probability distributions. In machine learning, cross entropy is commonly used to quantify the distance between the predicted probability distribution of a model and the true probability distribution. Cross entropy is calculated by multiplying each event's probability from the true distribution by its natural logarithm, then multiplying it with the corresponding probability from the model's predicted distribution, and finally summing the results and taking the negative value.

總之，熵衡量機率分佈內的不確定性，而交叉熵衡量兩個機率分佈之間的差異。交叉熵通常用作機器學習任務中的損失函數，例如分類，以根據預測分佈和真實分佈之間的差異來優化模型參數

In summary, entropy measures uncertainty within a probability distribution, while cross entropy measures the divergence between two probability distributions. Cross entropy is often used as a loss function in machine learning tasks such as classification to optimize model parameters based on the difference between predicted and true distributions.

Formula

## DFS

一張含有 文字, 螢幕擷取畫面, 圖表, 字型 的圖片

自動產生的描述

It follows each path to its greatest depth before

moving on to the next path

 When leaf node is reached, trace it back 到達葉子節點時回溯

A stack is maintained for controlling the search 控制堆疊來控制搜尋

 The first entry is popped 第一個入的已經彈出

 Spanned nodes are inserted at head of the list 新加入的節點家在list的前面

## BFS

一張含有 文字, 螢幕擷取畫面, 圖表, 行 的圖片

自動產生的描述

Examine all nodes one level down from the root node

More memory usage more than depth-first

 Depth increases gradually

 Goals in lower depth will be found first

A queue佇列is maintained for controlling the search, Newly spanned nodes新跨入的節點 are inserted at back of the queue插入到後面的佇列

## DFS VS. BFS

Ways of managing open list

DFS: inserted at front (as stack)

BFS: inserted at back (as queue)  
 It is often not easy to consider optimality, time complexity and space complexity simultaneously

DFS: lower memory requirement

BFS: minimum-length path  
 Which is better?

It depends on the problems

What is the shape of state space

Where are the goal states located

DFS VS. BFS

 DFS is more often used: Less memory usage

 Search tree may have very deep path, and goal node is in shallower part

Breadth-first good, but depth-first poor

8-Puzzle: very deep path, DFS not good

 All paths are of similar length

DFS good (do not try lower level)

 Search tree has high branching factor

DFS good (BFS consumes too many memories)

## CHARACTERISTICS OF SEARCH ALGORITHMS搜尋演算法的特性

Completeness完整性：如果存在目標狀態，則如果保證找到目標狀態，則搜尋方法是完整的。例如拼圖問題的 DFS 不完整。

A search method is complete if it is guaranteed to find a goal state provided there exists goal state.

e.g. DFS for puzzle problem is not complete.

Optimality 最適性 如果保證找到現有的最佳解，則搜尋方法是最優的。「最好」與「善良」的定義相關

A search method is optimal if it is guaranteed to find the best solution that exists“Best” is relevant to the definition of “Goodness” .

Irrevocability (no back tracing) 不使用回溯的方法是不可撤銷的。不可撤銷的方法，例如爬山，往往會被局部最優所愚弄

Methods that do not use backtracking are irrevocable. Irrevocable methods, such as hill-climbing, tend to be fooled by local optima.

Admissibility可接受性（Admissibility） 如果搜尋演算法能夠保證在任何時候存在一條最優路徑時都能找到解決方案，那麼該搜尋演算法就是可接受的。  任何演算法 A\* 都是可接受的。

A search algorithm is admissible if it is guaranteed to find a optimal path to a solution whenever such a path exists.

Any algorithm A\* is admissible.

## Minimax

一張含有 文字, 螢幕擷取畫面, 設計 的圖片

自動產生的描述

一張含有 行, 圖表, 設計, 三腳架 的圖片

自動產生的描述

## 🕰️ 最小最大演算法的時間複雜度

The time complexity of the minmax algorithm depends on the path factor and depth of the game tree. The degree is the depth power of the channel factor, which is O(B^D). The complexity increases with the branching factor and depth.

最小最大演算法的時間複雜度取決於遊戲樹的分支因子和深度。分支因子表示每個遊戲狀態下可能的移動數量，而深度表示遊戲樹中的層數或回合數。最小最大演算法的時間複雜度是分支因子的深度次方，即O(B^D)。複雜度的增長隨著分支因子和深度的增加而增加。

## ⚖️ 最小最大演算法的局限性

The main limitation of the minmax algorithm is that **it is not efficient when dealing with certain games**. **When the branching factor and depth of the game are very large, the game tree will become very large, causing the game tree search to be very time-consuming and computationally resource-consuming**.

For example, in chess, the branching factor is on average 35 and the depth is on average 100, which results in a very large number of nodes in the game tree, making it difficult to perform a complete search. In order to solve this problem, we need to use other algorithms, such as α-β pruning.

最小最大演算法的**主要局限性在於它在處理某些遊戲時的效率不高**。當遊戲的**分支因子和深度非常大時，遊戲樹會變得非常龐大，導致遊戲樹搜索的時間和計算資源非常消耗**。

例如在國際象棋中，分支因子平均為35，深度平均為100，這導致遊戲樹的節點數量非常龐大，難以進行完整的搜索。為了解決這個問題，我们需要使用其他的演算法，如α-β剪枝法。

## ✂️ α-β剪枝法中

### Alpha是甚麼?

alpha is the temporary maximum during the execution.

alpha 是執行期間的暫時最大值。

alpha can increase only (could be replaced by a larger entry) during the the maximization process.

**alpha 只能在最大化過程中增加**（可以用更大的條目取代）

alpha is the maximum finally after all elements are visited

**alpha 是所有元素被存取後最終的最大值**

### beta 是甚麼?

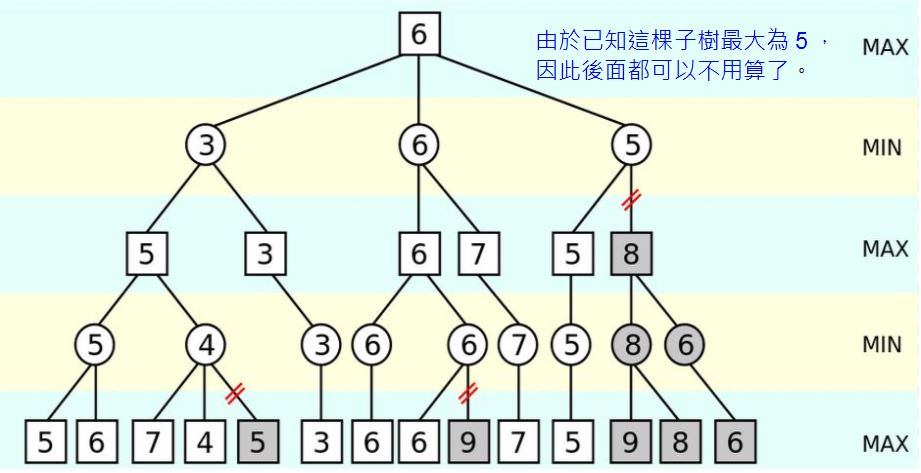
**beta is the temporary minimum during the execution.**

**beta 是執行期間的暫時最小值。**

**beta can decrease only beta 只能減少**

**beta is the maximum finally after all elements are visited** **beta 是所有元素訪問後最終的最大值**

### ALPHA-BETA PRUNING



A method that can often cut off a large part of the game tree. Based on the idea that if a move is clearly bad, there is no need to waste time to search the consequences of it completely

ALPHA-BETA 剪枝 

一種通常可以剪掉賽局樹大部分的方法。基於這樣的想法：如果一個舉動明顯是不好的，那麼就沒有必要浪費時間去徹底尋找它的後果

**The α-β pruning method is an optimized min-max algorithm used to reduce the amount of calculations in game tree search**. **It improves search efficiency by limiting the search scope and avoiding unnecessary parts of the game tree**.

**When a node is searched, the α-β pruning method will determine the nodes that do not need to be searched based on the known best movement strategy, and then prune.**

This method can greatly reduce the time and computing resources of game tree search and improve the efficiency of the game algorithm.

**α-β剪枝法是一種優化的最小最大演算法，用於减少遊戲樹搜索中的計算量。它通過限制搜索範圍，避免遊戲樹搜索不必要的部分，從而提高搜索效率。**

**當搜索到某個節點時，α-β剪枝法會根據已知的最佳移動策略來確定不必搜索的節點，從而剪枝。**

這種方法可以大大減少遊戲樹搜索的時間和計算資源，提高遊戲演算法的效率。

## 🔄 最佳移動策略

In game theory, **the optimal movement strategy refers to the strategy developed by players in the game that can not only maximize their own benefits, but also inhibit their opponents' maximum benefits.** The player's goal is **to gain maximum benefit through optimal movement strategies, while opponents work hard to prevent you from winning**. The optimal movement strategy can be formulated based on game rules and game status, and can be adjusted accordingly in each turn based on the opponent's movements.

遊戲理論中，**最佳移動策略表示玩家在遊戲中制定的既能使自己獲得最大利益，又能抑制對手獲得最大利益的策略**。玩家的目標**是通過最佳移動策略來獲得最大利益，同時對手也在努力阻止你獲勝。**最佳移動策略可以根據遊戲規則和遊戲狀態來制定，並且可以在每一個回合中根據對手的移動作出相應的調整。

## ➕ 以极大和极小的策略进行移動

**The "maximum" and "minimum" in the name of the minmax algorithm represent the two players in the game, namely the maximum player and the minimum player. The largest player tries to maximize his own interests, while the smallest player tries to minimize the largest player's interests.**

When it is the turn of the largest player to move, **he will choose the move that maximizes his own interests**; and when it is the turn of the smallest player to move, **he will choose the move that maximizes the interests of the largest player.** Such strategies are carried out alternately until the conditions for the end of the game are reached.

**最小最大演算法的名稱中的"最大"和"最小"表示的是遊戲中的兩個玩家，即最大玩家和最小玩家。最大玩家試圖最大化自己的利益，而最小玩家則試圖最小化最大玩家的利益。**

當輪到最大玩家移動時，他會**選擇使自己利益最大化的移動**；而當輪到最小玩家移動時，**他會選擇使最大玩家得到最小利益的移動**。這樣的策略交替進行，直到達到遊戲結束的條件。

## 🌳 遊戲樹的分析

The minmax algorithm uses game trees to analyze various movement strategies of the game. **A game tree is a tree-like structure consisting of all possible moves and game states of the game.**

By recursively computing each node of the game tree, **the minmax algorithm can determine the benefit generated by each move and select the next move based on the best benefit.**

The depth of the game tree depends on the complexity of the game and the length of the game.

最小最大演算法使用遊戲樹來分析遊戲的各種移動策略。**遊戲樹是由遊戲的所有可能移動和遊戲狀態構成的樹狀結構**。

通過遞歸運算遊戲樹的每個節點，**最小最大演算法可以確定每個移動所產生的利益，並根據最佳的利益來選擇下一步的移動**。

遊戲樹的深度取決於遊戲的複雜性和棋局的長度。