AI無線通訊系統實

Module 1 Report

賴昱凱 111511141

1. Description and final results of mini project

一張含有 文字, 螢幕擷取畫面, 行, 圖表 的圖片

AI 產生的內容可能不正確。

本實驗主要利用(15, 11) hamming code將訊號encode、BPSK進行modulation，在加上高斯雜訊後，利用deep learning的方式將其decode回原訊號。

1. 訊號產生

一張含有 螢幕擷取畫面, 文字, 字型 的圖片

AI 產生的內容可能不正確。

產生隨機於[0, 1]的整數，即二進位，並指定輸出形狀為(dataset\_size, 11)，代表共有dataset\_size筆資料，每筆資料有11個位元。

1. (15, 11) hamming code

一張含有 文字, 螢幕擷取畫面, 軟體, 陳列 的圖片

AI 產生的內容可能不正確。

(n, k)-linear Block Codes是一種encode方式，他可以皆由額外的位元去驗證接收到的訊號是否正確，若有少數位元遺失，可以將其回復，具體方式如下：

我們有k-bit message 𝒎, n-bit codeword **c**，兩者將符合以下關係：

G為一個的矩陣，稱為generator matrix。

並存在一個的H矩陣，符合以下關係：

Rows of ***H*** span the nullspace of ***G***，我們稱***H***為parity check matrix。而其中generator matrix以及parity check matrix又符合：

一張含有 文字, 字型, 行 的圖片

AI 產生的內容可能不正確。

可以更容易的做出generator matrix。

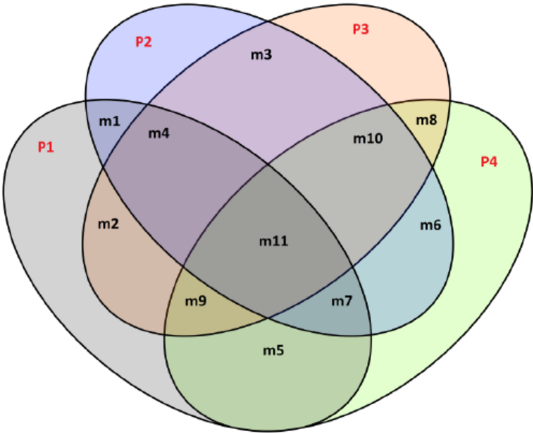
因此，若收到的訊號用parity check matrix驗證後等於零，就可以知道這個訊號沒有錯誤。而不同的linear Block Codes有不同的糾正方式，以比較簡單的(7, 4) hamming code為例，他利用下圖進行糾正：

一張含有 文字, 圖表, 圓形, 字型 的圖片

AI 產生的內容可能不正確。

簡單來說，它是由三個圓組成，若在圓中的m為1的總數為奇數，那該圓的c就是1，反之總數為偶數就是0，組成7位元的codeword。當有錯誤出現時，若只有一個bit出錯也可以很輕易地找到錯誤並修正他。

而本實驗使用的(15, 11) hamming code就比較複雜了，圖片如下。



1. BPSK modulation

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

因為SNR是以分貝為單位，因此要先將其轉換為線性比例才是訊號的功率：

BPSK modulation則是將0, 1映射到 -1, +1，利用就可以達成這個目的，而最後再乘上剛剛算出來的，讓輸出訊號能量與SNR匹配。

至於為什麼要開根號，這是因為訊號功率為振幅的平方，因此訊號本身應該要乘以根號P才會讓其功率為P倍。

1. AWGN channel

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

這個步驟就是讓傳出去的訊號加上高斯雜訊以模擬現實中訊號被影響的問題。其中參數分別為(平均值μ, 標準差σ, 形狀)。

1. Deep Learning

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

利用一個input dimension=15、output dimension=的模型進行訓練。輸入即為加上雜訊的codeword，因此為15位元，而輸出結果為一個機率分布，代表每一種code可能的機率，因為有11個位元，且每個位元都有可能是0或1兩種選項，因次共有種可能。而預測結果即為機率最高的那個選項。訓練結果如下：

一張含有 文字, 行, 繪圖, 圖表 的圖片

AI 產生的內容可能不正確。一張含有 文字, 行, 圖表, 繪圖 的圖片

AI 產生的內容可能不正確。

1. Description and discussion of all decoding method in Module 1
2. Hard decision

這是最簡單的一種解碼方式，簡單來說就是經過BPSK modulation並加上雜訊的訊號若大於0，就將其判斷為1，反之若小於0，就判斷為0。這個方式計算量小、解碼速度快，不過性能較差。

1. Syndrome decoding

這個方式主要用於linear block codes，例如前面提到的hamming code。利用Parity-Check Matrix來檢查是否出現錯誤，，若s = 0就代表沒有錯誤，不等於0就代表有錯誤，並可以根據s來判斷錯誤的地方，本實驗直接利用查表的方式將每一種錯誤都有對應的錯誤位元。這個方式可以處理錯誤，但可處理的數量不多。

1. Maximum Likelihood (ML) decoding

這個方式是將所有可能的組合都通過generator matrix後得到他們的codewords，並經過BPSK modulation，計算所有經過調變的可能結果與收到的訊號y的距離，最短的那個即為預測的結果。這個方式雖然會需要極大量的計算資源以及計算時間，但他也是最佳的解碼方式，並適用於任何encode方式。

1. Auto-encoder

這個方式並不是像其他的方式是利用已知的encode方式再想辦法去解碼，而是讓模型自行尋找最適合的encode以及decode方式，也可以算是一種un-supervised learning，我們只需事先告知模型他的input dimension、encoder和decoder之間的channel dimension，並讓decoder將加上雜訊的channel資料想辦法解碼回input即可。整個過程都是利用模型自行尋找相關特徵，因此我們不必手動設計編碼及解碼方式，不過這個方式需要大量數據進行訓練，同時訓練過程也需要大量的計算資源，硬體需求高。

1. Deep Learning decoding

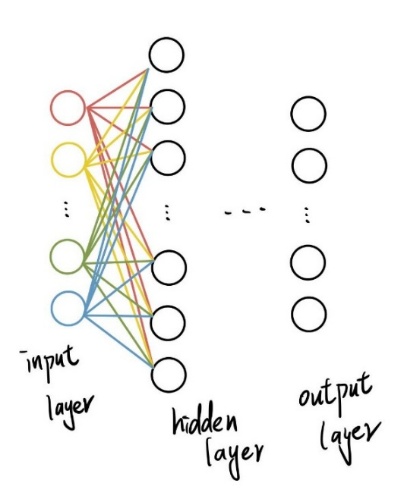
這個方式與上一個auto-encoder有異曲同工之妙，不過他是利用已知的encode方式，僅將Decoder的部分應用深度學習進行解碼。深度學習有很多種的架構，包括CNN、RNN/LSTM、transformer等，而本實驗是選擇使用最基礎的DNN，以下簡單介紹DNN model。

我使用的模型架構如下：

一張含有 文字, 螢幕擷取畫面, 字型 的圖片

AI 產生的內容可能不正確。

本模型是由2層hidden layer以及input、output layer，共4層所組成的DNN模型，DNN模型的全稱為Deep Neural Network，顧名思義代表所有的節點都與前後層的所有節點相連，利用權重做相乘往下一層前進，因此整個每層共有個參數，以本模型為例：



全部相加即為本模型的總參數：18128

除了linear以外，還有一個部分稱為activation function，他是非線性的layer，因為若整個model都是linear的layer，會很難處理現實中的問題，因為現實中的問題基本上都不會是linear的，因此需要在model中加上activation function來讓模型更加靈活。

本模型皆使用ReLU這個最常見的activation function，他會將正數原封不動的送給下一層，而遇到負數就將其設為0，這是一種最不耗計算資源的non-linear方式。下圖還有更多常見的activation function。

一張含有 文字, 圖表, 字型, 數字 的圖片

AI 產生的內容可能不正確。

本模型的輸出層為有16個節點的神經層，最後一層利用softmax作為actiavtion function，利用指數化確保輸出皆為正數，再讓所有輸出的值都介於0~1且相加為1，代表各種類的可能性，最高的那個就代表此次模型預測的結果，這是幾乎所有分類問題處理輸出最基本的方式。

Softmax:

Simulation results (BLER of all decoding method) is necessary!

一張含有 文字, 行, 圖表, 繪圖 的圖片

AI 產生的內容可能不正確。BLER\_hard\_decision = [0.07629, 0.05586, 0.03645, 0.02239, 0.01302, 0.0059, 0.00239, 0.00095, 0.00017]

BLER\_SymdromeDecoding = [0.26725, 0.187, 0.114, 0.068, 0.036375, 0.01578, 0.005475, 0.00147, 0.000282]

BLER\_ML = [8.05625e-02, 5.10625e-02, 2.65625e-02, 1.53125e-02, 4.96250e-03, 1.50875e-03, 3.70000e-04, 5.62500e-05]

BLER\_autoencoder = [0.20328333, 0.13721667, 0.08078333, 0.04095, 0.01881667, 0.00628333, 0.00201667]

BLER\_DL = [0.1975, 0.13392857, 0.07, 0.0375, 0.02189286, 0.00413571, 0.00114286, 0.00022857]