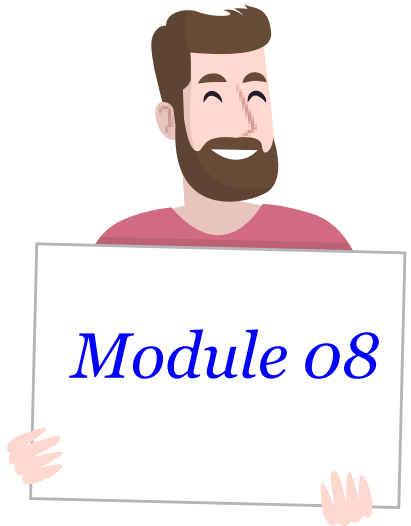
神經網路技巧及討論



Estimated time:
45 min.

學習目標

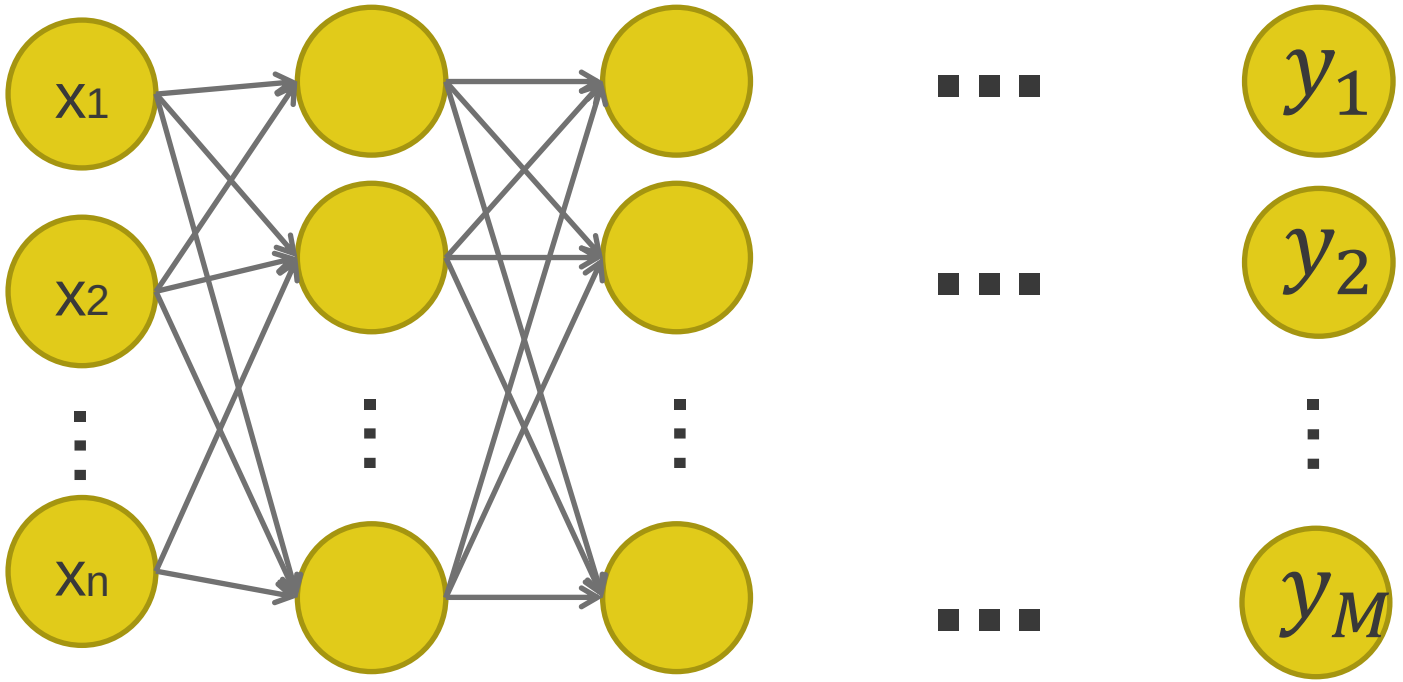
- 8-1: 激活函數討論
- 8-2: Lagrange與正則化
- 8-3: Dropout及Batch Normalization

8-1:激活函數討論

- 激活函數討論

激活函數討論

- 激活函數賦予神經網路非線性的能力
- 好的激活函數會影響網路訓練的結果

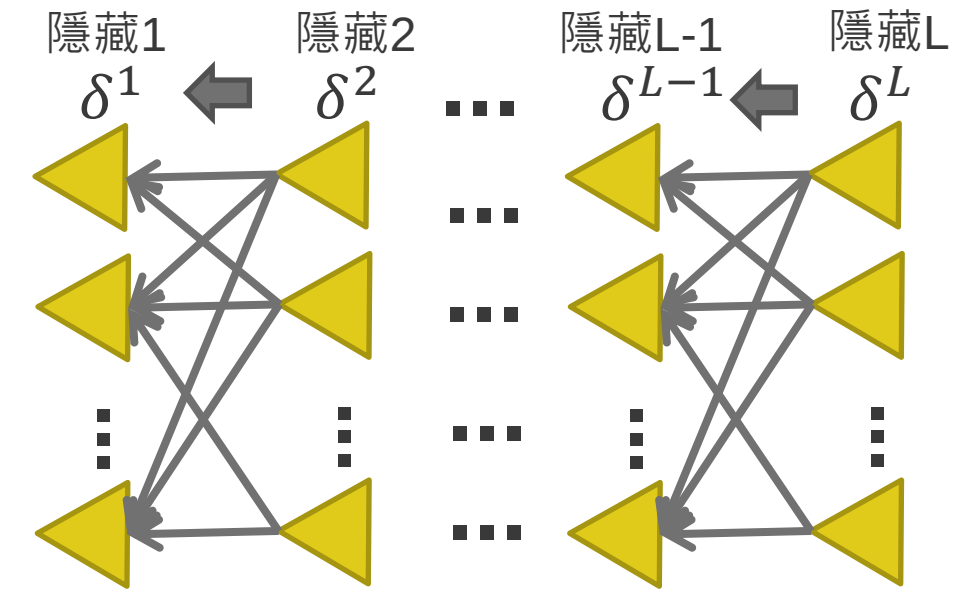


激活函數討論

- 過去神經網路的專家喜歡用sigmoid函數來當激活函數
  - 現在被證明出沒有relu函數好
- 在做Backpropagation時，每將error signal往前傳一層時，都會需要乘以一次當層激活函數的微分

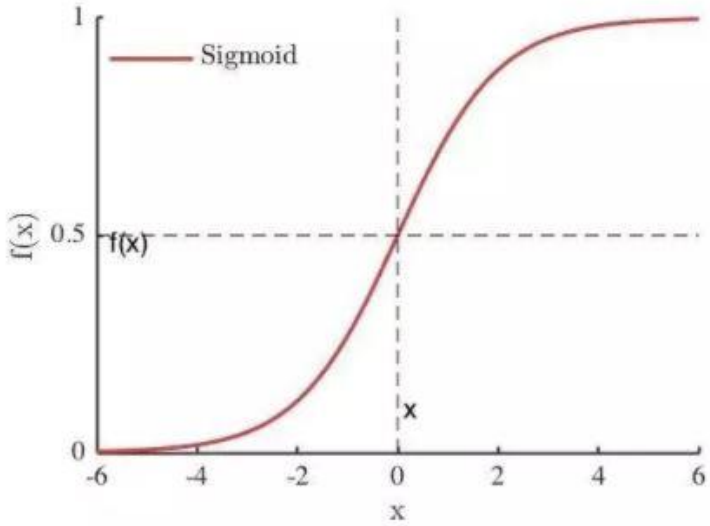
$$\begin{cases} \delta^L = \sigma'(z^L) \bullet \nabla C(y) \\ \delta^l = \sigma'(z^l) \bullet (W^{l+1})^T \delta^{l+1} \end{cases}$$

Backpropagation過程中
牽扯到大量激活函數的微分



激活函數討論

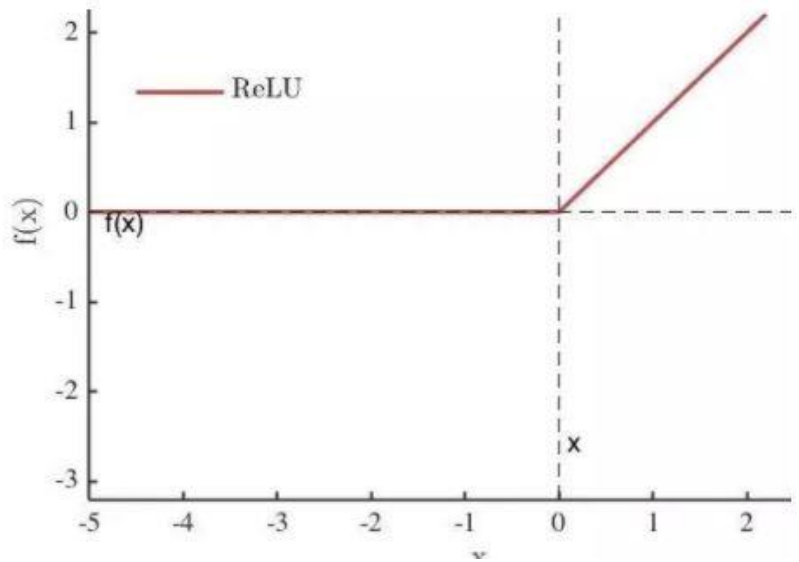
- **Sigmoid函數微分最大的地方為x=0，此時微分值為0.25**
  - **會造成當神經網路非常深的時候，其在做Backpropagation時，error signal每往前傳一層，數值越小，造成淺層網路參數幾乎無法調到參數(無法學習)**



$$f(x)=\frac{1}{1+e^{-x}}$$

激活函數討論

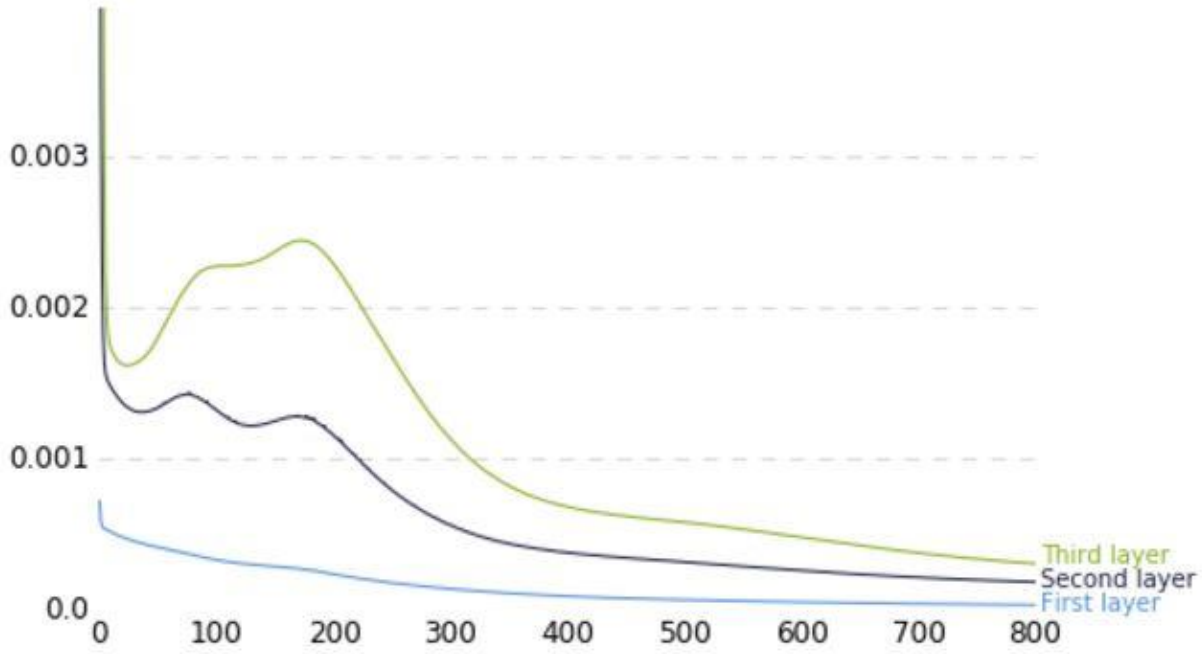
- 相較Sigmoid函數，relu函數在x>0時微分為1，x<0時微分為0
  - 比較不會有error signal逐漸變小的問題



$$y = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{if } x < 0 \end{cases}$$

激活函數討論

- 這種隨著訓練過程中，error signal會越傳越小，導致淺層網路無法學習的現象，叫做”梯度消失”問題



梯度消失問題

8-2:Lagrange與正則化

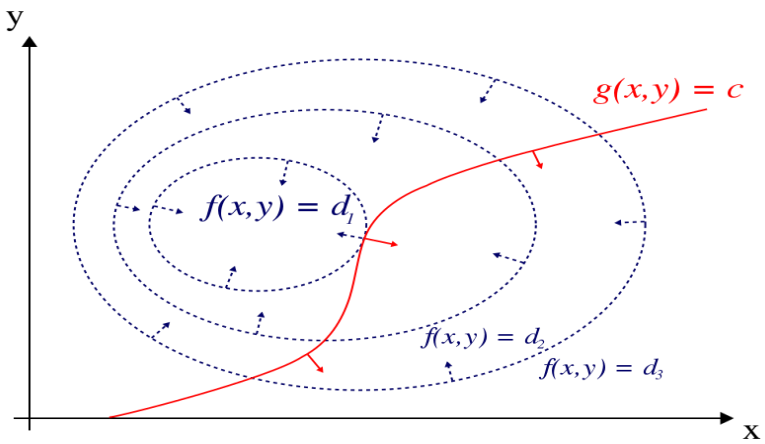
- **Lagrange複習**
- 正則化

Lagrange複習

- **Lagrange是一種解決最佳化問題的方法**
  - 通常包含目標函數以及多個限制式

maximize $f(x, y)$

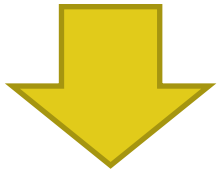
subject to $g(x, y) = 0$



Lagrange複習

- **假設我們有一最佳化問題如下，請使用Lagrange 解決此最佳化問題：**

$$\max\{4x^2 + 10y^2\}$$
$$\text{s.t. } x^2 + y^2 - 4 \leq 0$$



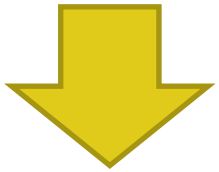
將限制式乘以lagrange multiplier
並成新的函數L

$$L(x, y, \lambda) = 4x^2 + 10y^2 + \lambda(x^2 + y^2 - 4)$$

Lagrange複習

- 當得到新的函數L後，針對不同變數作微分
  - 除了本來的變數x,y為變數外， lagrange multiplier(λ)也須被視為變數

$$L(x, y, \lambda) = 4x^2 + 10y^2 + \lambda(x^2 + y^2 - 4)$$



$$\nabla_x L(x, y, \lambda) = 8x + 2x\lambda = 0$$

$$\nabla_y L(x, y, \lambda) = 20y + 2y\lambda = 0$$

$$\nabla_\lambda L(x, y, \lambda) = x^2 + y^2 - 4 = 0$$

Lagrange複習

- 將微分過後的等式做聯立方成解
  - 會得到多組解，一一帶入原始式子即可以找到最佳解

$$\nabla_x L(x, y, \lambda) = 8x + 2x\lambda = 0$$

$$\nabla_y L(x, y, \lambda) = 20y + 2y\lambda = 0$$

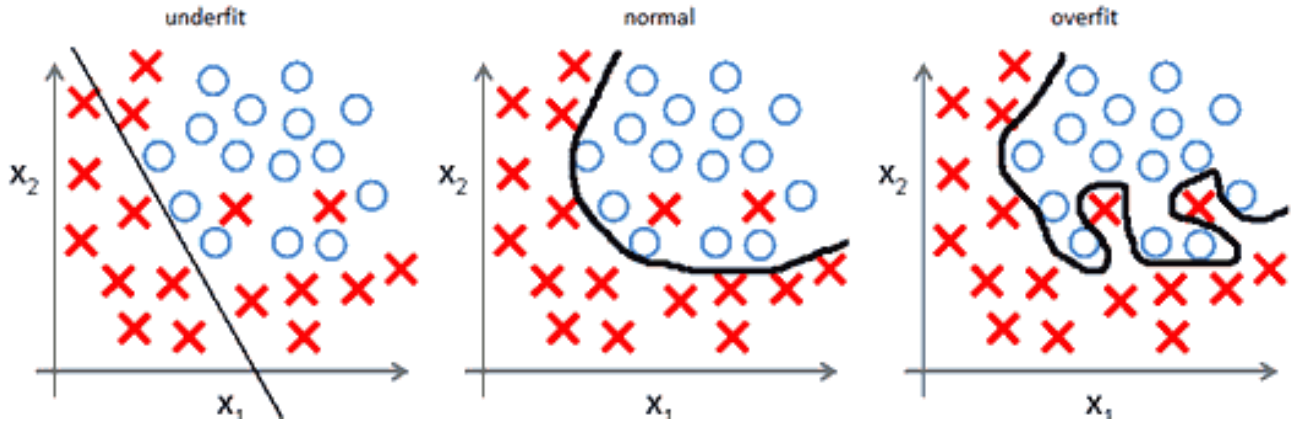
$$\nabla_\lambda L(x, y, \lambda) = x^2 + y^2 - 4 = 0$$

λ =4
(x,y) = (0, 2), (0, -2), (2, 0), (-2, 0)

f(0, 2) 的值最大

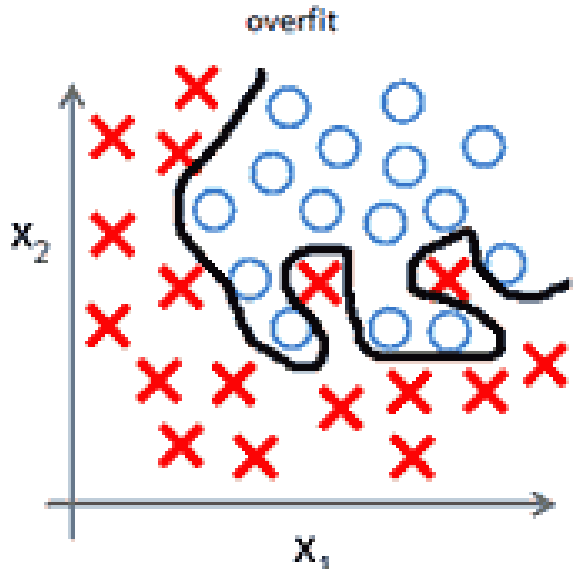
正則化

- 當模型如左下圖時，模型在怎麼調整也無法將此分類問題分好
  - 我們稱模型為underfitting
- 當模型如中間下圖時，模型表現剛剛好，雖然有時還是會分錯
- 當模型如右下圖時，模型100%在訓練資料集完美分類
  - 雖然看似完美，但往往此模型在部屬時實際表現不佳
  - 我們稱模型為overfitting



正則化

- 當模型overfitting時，其模型的參數之間的標準差會非常大
  - 因為唯有當參數之間的標準差非常大才有可能產生哪麼極端且陡峭的曲線

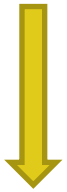


模型參數之間標準差大

正則化

- 在神經網路裡，使用正則化技巧減緩overfitting發生
  - 增加正則向到原有的損失函數裡

$$L_{new}(\theta) = L_{old}(\theta) + \text{正則向}$$



MSE, Cross-Entropy

正則化

- **線性代數複習L1 norm以及L2 norm**

$$\left\| \begin{bmatrix} 0.1 & 0.5 & -0.3 \\ -0.2 & 0.4 & 0.2 \\ -0.1 & 0.3 & 0.3 \end{bmatrix} \right\|_1 = |0.1| + |0.5| + |-0.3| + |-0.2| + |0.4| + |0.2|$$
$$+|-0.1| + |0.3| + |0.3| = 1.2$$

L1 norm

$$\left\| \begin{bmatrix} 0.1 & 0.5 & -0.3 \\ -0.2 & 0.4 & 0.2 \\ -0.1 & 0.3 & 0.3 \end{bmatrix} \right\|_2 = (0.1)^2+(0.5)^2+(-0.3)^2+(-0.2)^2+(0.4)^2+(0.2)^2$$
$$+(-0.1)^2+(0.3)^2+(0.3)^2$$

L2 norm

正則化

- 將正則化的技巧加入神經網路中，只要將所有W權重之L1/L2 norm加到現有損失函數後面即可
  - 其意義為當極小化損時函數時，同時確保W所學到的權重越接近0越好
  - W權重皆接近0的情況下，網路產生高維度的曲線會比較平滑，使得overfitting比較不會發生

$$\sigma(W^L \ldots \sigma(W^2\sigma(W^1x + b^1) + b^2) \ldots + b^L)$$

$$L_{new}(\theta) = L_{old}(\theta) + \lambda\|\theta\|_1$$

$$\{W^1, W^2, \ldots\ldots, W^L\}$$

L1 正則化

$$L_{new}(\theta) = L_{old}(\theta) + \frac{\lambda}{2}\|\theta\|_2$$

$$\{W^1, W^2, \ldots\ldots, W^L\}$$

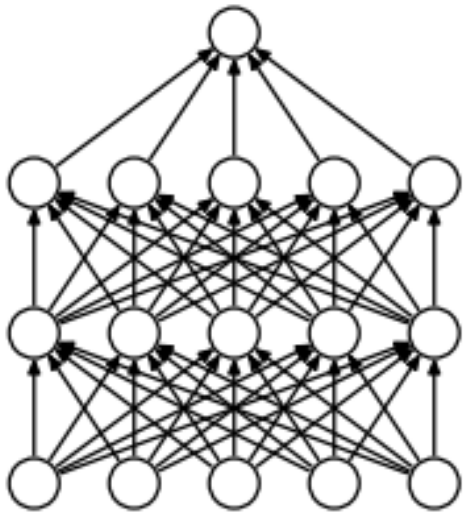
L2 正則化

8-3:Dropout及Batch Normalization

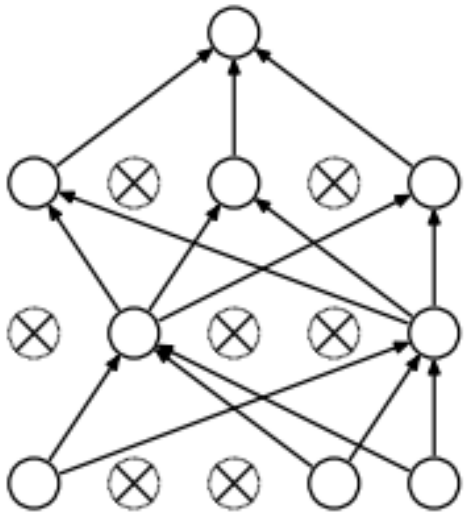
- Dropout介紹
- Batch Normalization介紹

Dropout介紹

- **Dropout是一種可以增加網路準確度的技巧**
  - **其主做法是，當網路在訓練並輸入批次資料時，隨機將P%之神經元消失**



(a) Standard Neural Net

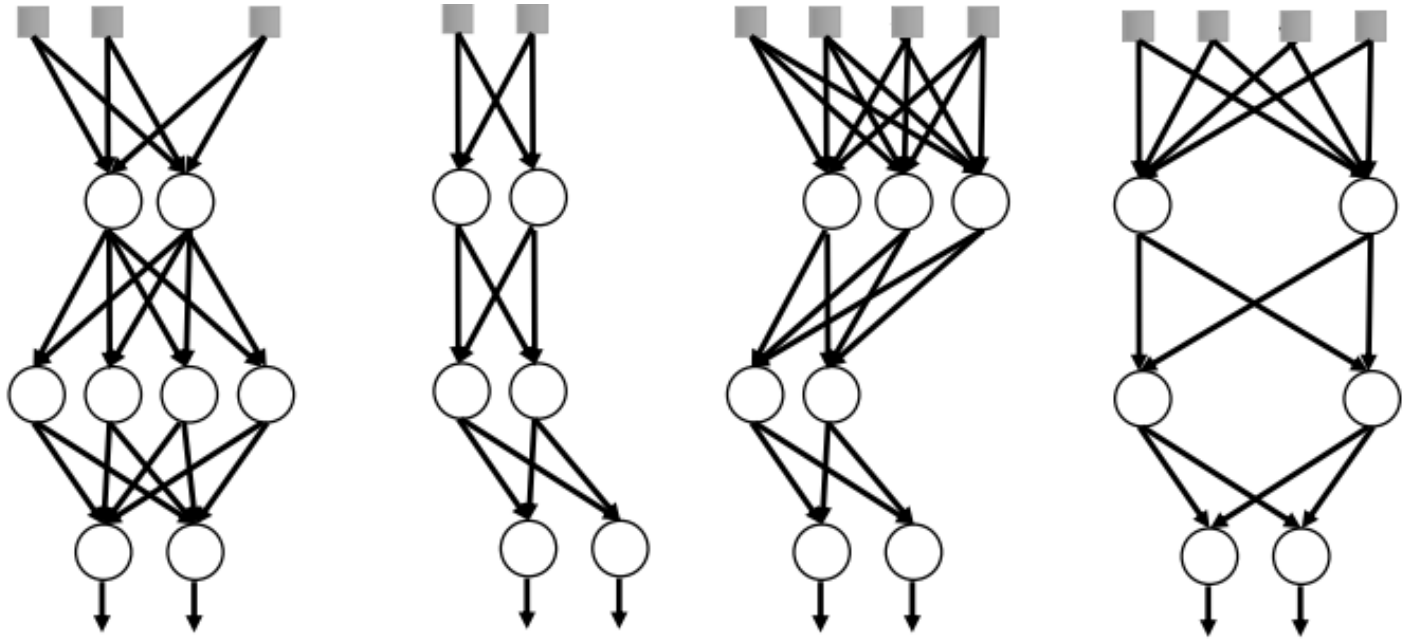


(b) After applying dropout.

Dropout介紹

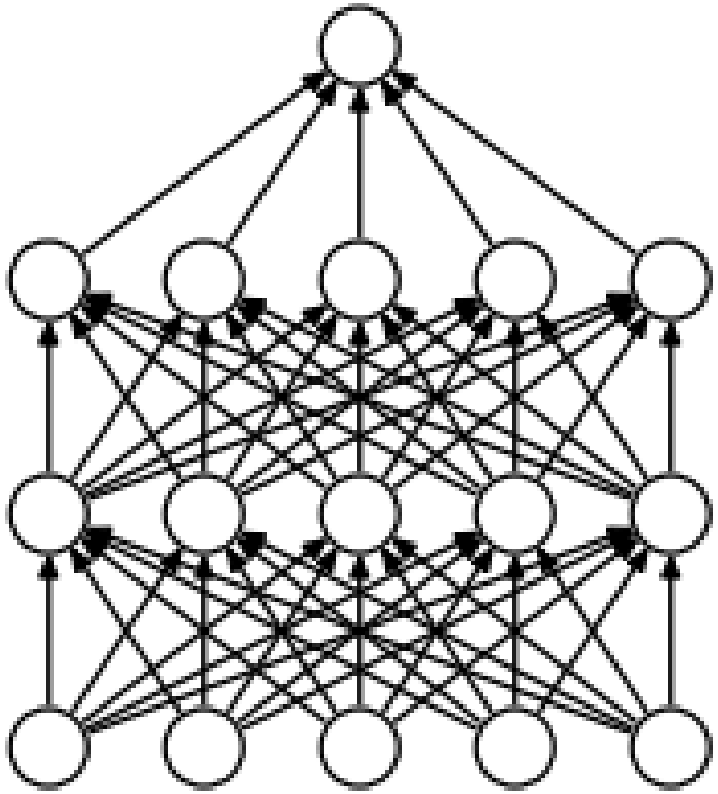
- 在使用Dropout訓練網路時，輸入每個批次時所消失的神經元是不一樣的
  - 可以讓不同神經元學習到不同面向的東西

第一批次　第二批次　第三批次　第四批次



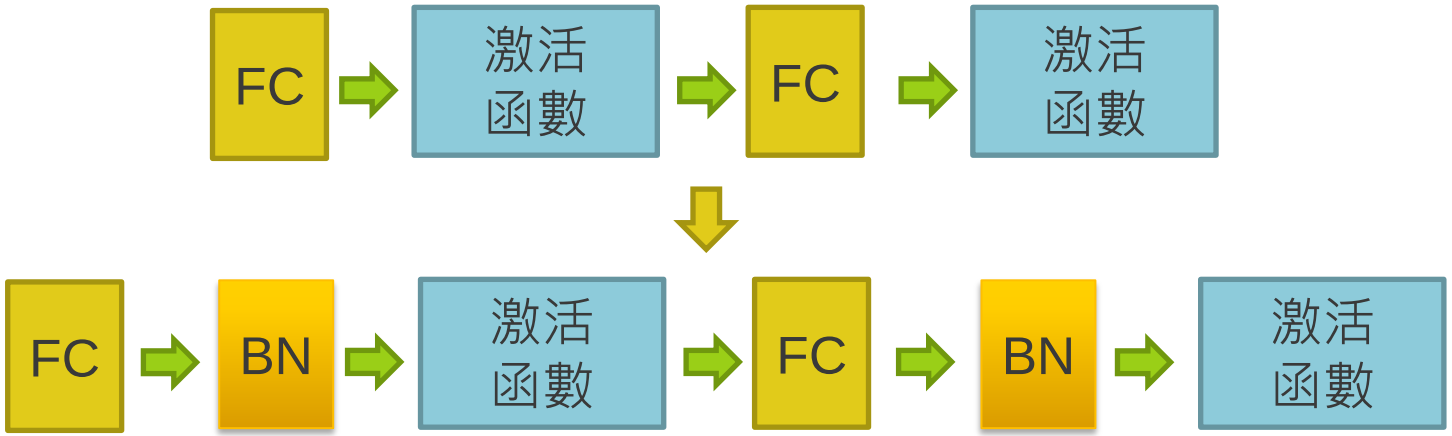
Dropout介紹

- 在使用Dropout測試網路時，再將所有神經元連結會來做預測



Batch Normalization介紹

- **Batch Normalization是一種增加網路訓練速度的技巧**
  - **其作法將資料出入每層激活函數前，做批次資料的標準化**
  - **或者可以看成在每個激活函數前加上了一個Batch Normalization轉換成**



$$\sigma(W^L \dots \sigma(W^2\sigma(W^1x + b^1) + b^2) \dots + b^L)$$

$$\sigma(BN(W^L \dots \sigma(BN(W^2\sigma(BN(W^1x + b^1))) + b^2) \dots + b^L))$$

Batch Normalization介紹

- **BN層的做法是將每一批次資料做標準化**
  - **(每筆資料 - 批次平均)/批次標準差再輸入激活函數**

$$\sigma(BN(W^L \ldots \sigma(BN(W^2\sigma(BN(W^1x + b^1))) + b^2) \ldots + b^L))$$

$$BN(x_i) = \gamma\left(\frac{x_i - \mu_B}{\sqrt{{\sigma_B}^2 + \varepsilon}}\right) + \beta$$

“1” (usually)

“0.001” (usually)

“0” (usually)

Demo 8-3

- 開啟Demo_8-3.ipynb
- 建構更精簡寫法的DNN神經網路
- 在網路加上正則向
- 在網路加上Dropout

線上Corelab

- 題目1：激活函數ReLU與Leaky_ReLU
  - TensorFlow中有提供許多種激活函數，其中ReLU是我們常常使用的函數，Leaky_ReLU是ReLU的改進版，請用以下的矩陣帶入這兩種函數中觀察輸出的不同
- 題目2：資料的正規化
  - 請試著在以下的訓練程式碼中加上正則項觀察準度的變化
- 題目3：Dropout的應用
  - 請試著依照以下指定的參數建立一個5層的DNN網路，並使用drop out觀察是否對準度有何影響
  - 優化器請使用Adam、Epoch = 15、Batch_size = 100、Learning_rate = 0.001、正則項Beta = 0.02、Dropout = 0.5

本章重點精華回顧

- 激活函數與Backpropagation的關係
- 正則化的技巧
- Dropout技巧
- Batch Normalization技巧

Lab:神經網路技巧

- **Lab01: 更精簡寫法的DNN神經網路**
- **Lab02: 在網路加上正則向**
- **Lab03: 在網路加上Dropout**

Estimated time:
20 minutes


THANK YOU