機器與深度學習基礎知識初探
Weight initialization和 Batch Normalization

黃志勝 Chih-Sheng (Tommy) Huang

義隆電子人工智慧研發部

國立陽明交通大學AI學院合聘助理教授

Outline

- 神經網路結構設定完成後，最重要的是

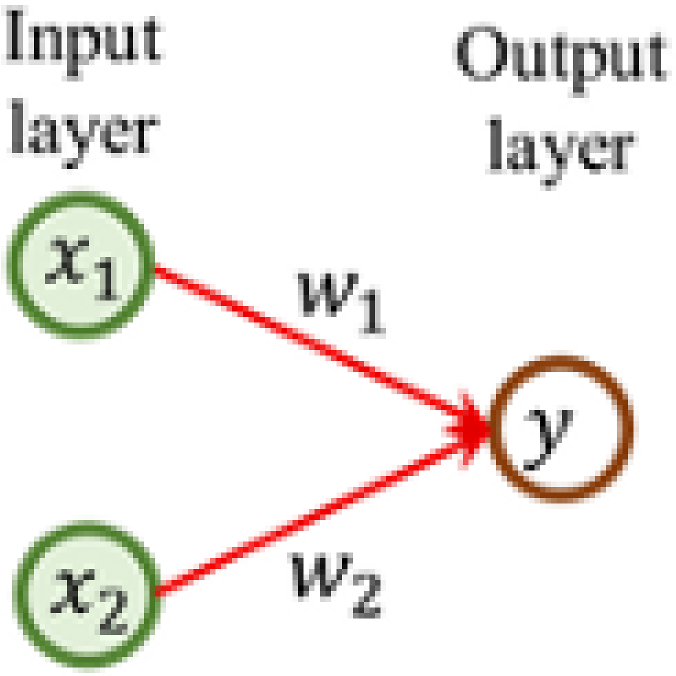
1. 如何更新權重 (梯度下降法)
2. 權重如何初始設定

- Weight initialization和Batch Normalization

Outline

- 1. weight初始值是0
- 2. Random initialization
- 3. Xavier initialization
- 4. He initialization
- 5. Batch Normalization

weight初始值是0



$$y = w_1 x_1 + w_2 x_2$$

Backward update:

$$w_i = w_i - \eta \Delta w_i$$

Forward:

$$\hat{y} = w_1 x_1 + w_2 x_2 = 0$$

weight的gradient

$$\Delta w_i = \frac{\partial E}{\partial w_i} = \frac{\frac{1}{2} y^2}{\partial w_i} = 0$$

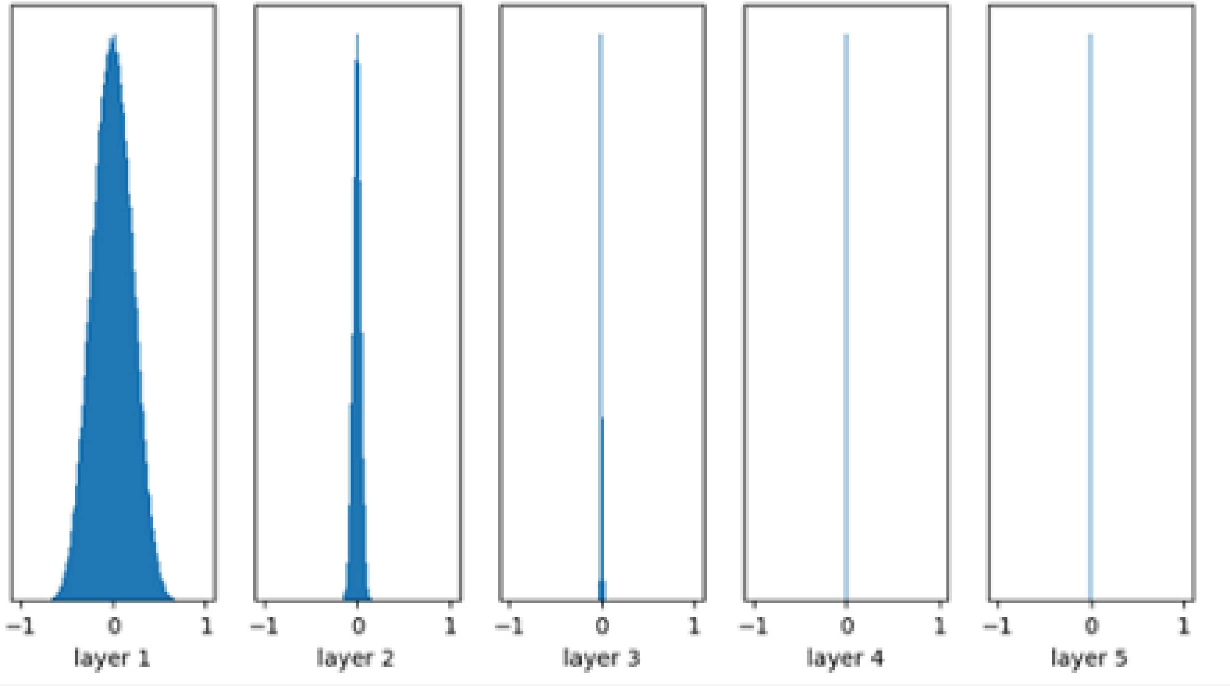
$$E = \frac{1}{2}(y - \hat{y})^2 = \frac{1}{2} y^2$$

Random initialization

所以第二個最簡單的想法，初始權重用隨機方式建立。

我們建立一個6層的MLP，每一層輸出的activation用tanh。

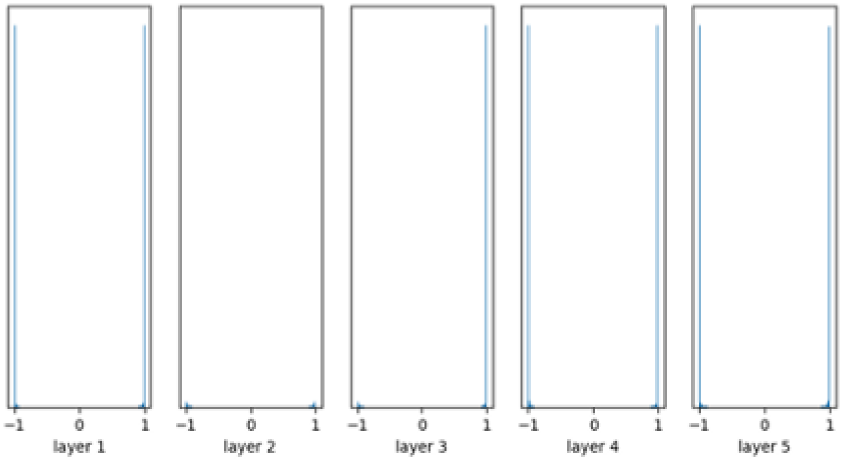
Weight從常態分佈(平均數為0，標準差為0.01)生成



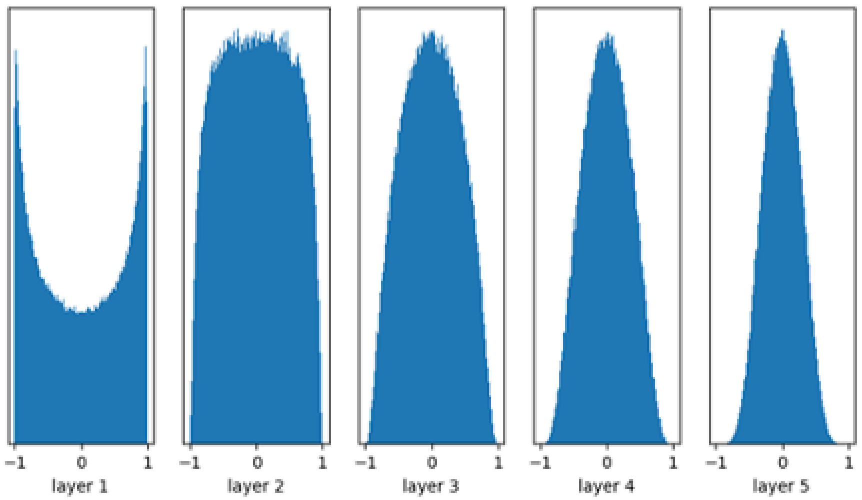
input mean 0.00065 and std 0.99949
layer 1 mean 0.00025 and std 0.21350
layer 2 mean -0.00002 and std 0.04516
layer 3 mean -0.00001 and std 0.00899
layer 4 mean -0.00000 and std 0.00168
layer 5 mean -0.00000 and std 0.00029

Random initialization

實驗2. 常態分佈(平均數為0，標準差為1)

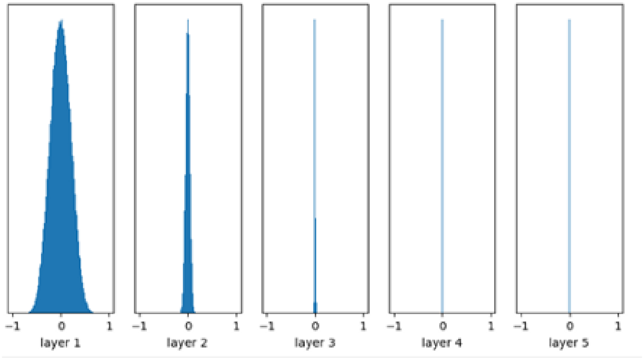


實驗3. 常態分佈(平均數為0，標準差為0.05)

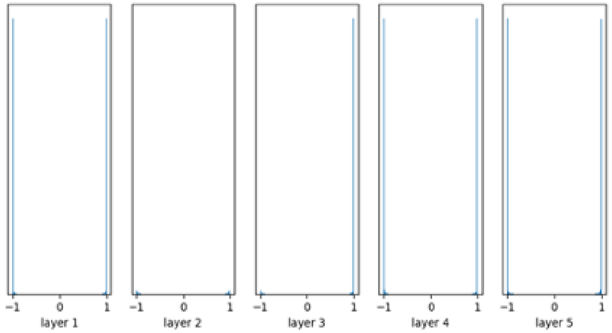


Random initialization

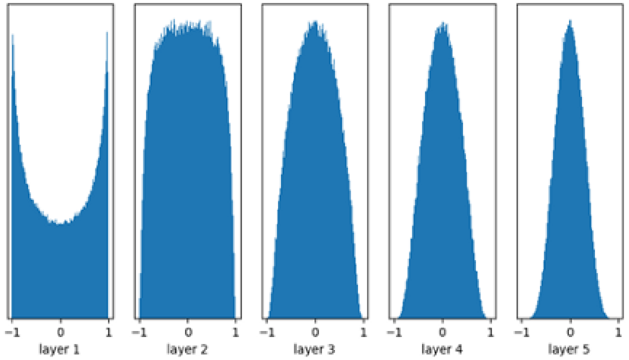
常態分佈
(平均數為0，標準差為0.02)



常態分佈
(平均數為0，標準差為1)



常態分佈
(平均數為0，標準差為0.05)



weight初始值從常態分布(也可以用均勻分布)的標準差變化會影響結果。

Xavier initialization

- 由Random initialization得知weight權重的生成會影響神經元的輸出太集中或是過於飽和。

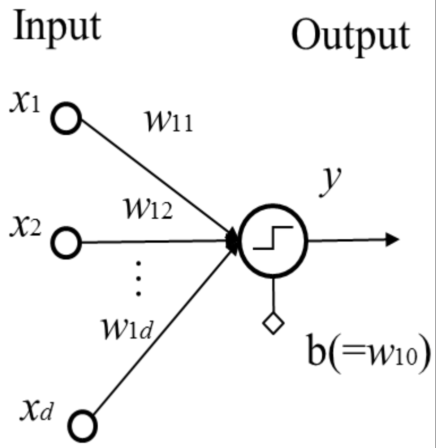
- Xavier的論文(Understanding the difficulty of training deep feedforward neural networks)的想法是希望神經元輸入(*xi*, *i=1,2,…,d*)和輸出(*y*)的變異數(variance，標準差的平方)保持一致。

Suppose$x_i, w_i \overset{iid}{\sim} Distribution(mean = 0)$

$$y = w_1x_1 + w_2x_2 + \cdots + w_dx_d \text{ (bias先不考慮)}$$

$$Var(y) = Var(w_1x_1 + w_2x_2 + \cdots + w_dx_d) = \sum_{i}^{d} Var(w_ix_i)$$

$$Var(w_ix_i) = \mathrm{E}(x_i)^2Var(w_i) + \mathrm{E}(w_i)^2Var(x_i) + Var(w_i)Var(x_i)$$



Xavier initialization

$$Var(y) = Var(w_1 x_1 + w_2 x_2 + \cdots + w_d x_d) = \sum_i^d Var(w_i x_i)$$

$$Var(w_i x_i) = Var(w_i) Var(x_i)$$

$$Var(y) = \sum_i^d Var(w_i x_i) = \sum_i^d Var(w_i) Var(x_i) = d \times Var(w_i) Var(x_i)$$

- Xavier initialization的想法是希望$Var(y)$=1

$$Var(y) = d \times Var(w_i) Var(x_i) = 1 \Rightarrow Var(w_i) = \frac{1}{d \times Var(x_i)}$$

因為前一層的輸出我們也希望其variance=1 (所以資料前處理很常做z-score處理)

$$Var(w_i) = \frac{1}{d} = \frac{1}{n_{inputnode}}$$

Xavier initialization

同樣得程序在back-propagation也推一遍就可以得到

$$Var(w_i) = \frac{1}{n_{outputnode}}$$

為了希望forward和backward的variance可以一致，除非 $n_{inputnode} = n_{outputnode}$，文章就直接取

$$Var(w_i) = \frac{2}{n_{inputnode} + n_{outputnode}}$$

$$Var(y) = d \times Var(w_i) = = \frac{2n_{inputnode}}{n_{inputnode} + n_{outputnode}}$$

Xavier initialization

如果採用相加除以2:

$$\frac{\left(\frac{1}{n_{inputnode}}+\frac{1}{n_{outputnode}}\right)}{2}=\left(\frac{n_{inputnode}+n_{outputnode}}{2n_{outputnode}n_{inputnode}}\right)$$

Suppose

Layer 1 node=32, Layer 2 node=64, Layer 3 node=128

$$Var(w_i)=\frac{2}{n_{inputnode}+n_{outputnode}}$$

Layer 1 = 2/(32+64)=2/96=0.021, Layer 2= 2/(64+128)=2/192=0.01

Layer 1 = (32+64)/(2*32*64)=0.023, Layer 2= (32+64)/(2*64*128)=0.00586

Note how both constraints are satisfied when all layers have the same width.

constraints: $Var(w_i)=\frac{1}{n_{inputnode}}$, $Var(w_i)=\frac{1}{n_{outputnode}}$

Xavier initialization

- 文章提到假設每一層的node數一樣

$$Var(w_i) = \frac{2}{n_{inputnode} + n_{outputnode}}$$

$$\forall i, Var\left[\frac{\partial Cost}{\partial s^i}\right] = \boxed{\left[nVar[W]\right]^{d-i}} Var[x] \qquad (13)$$

$$\forall i, Var\left[\frac{\partial Cost}{\partial w^i}\right] = \boxed{\left[nVar[W]\right]^{d}} Var[x]Var\left[\frac{\partial Cost}{\partial s^d}\right] \qquad (14)$$

所以這個方法只能盡量緩和gradient vanish/explode，不能避免。

Xavier initialization

$$w_i \overset{iid}{\sim} U(a, b)$$

$$var(w_i) = \frac{(b-a)^2}{12}$$

一般用均匀分布

$$w_i \overset{iid}{\sim} U(-\frac{1}{\sqrt{n}}, \frac{1}{\sqrt{n}})$$

$$var(w_i) = \frac{\left(\frac{2}{\sqrt{n}}\right)^2}{12} = \frac{1}{3n} = n\ var(w_i) = \frac{1}{3}$$

論文建議用他們提出來的normalized initialization

$$w \sim U\left(-\sqrt{\frac{6}{n_{inputnode} + n_{outputnode}}}, \sqrt{\frac{6}{n_{inputnode} + n_{outputnode}}}\right)$$

Xavier initialization

$$w_i \underset{\sim}{\overset{iid}{}} U(-b, b\ )$$

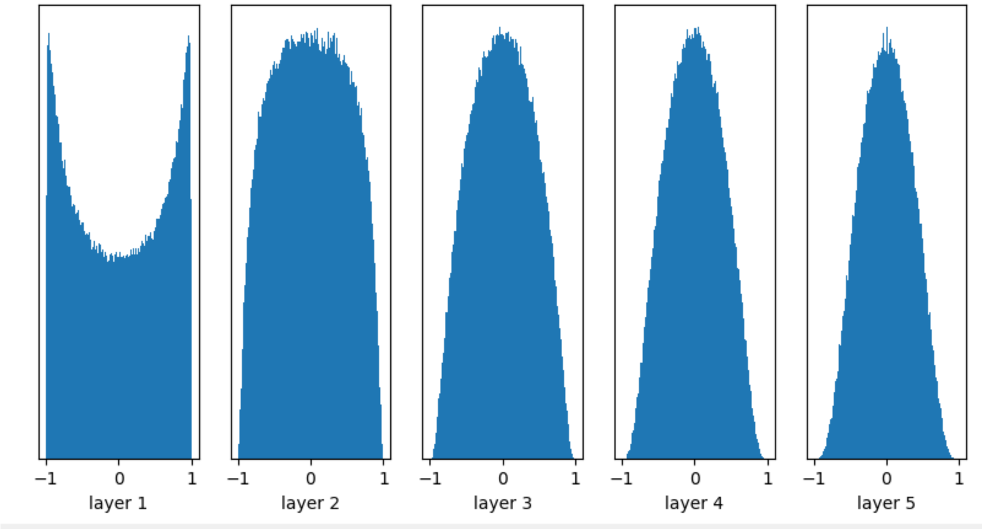
$$var(w_i) = \frac{(b-a)^2}{12} = \frac{b^2}{3} = \frac{2}{n_{inputnode} + n_{outputnode}}$$

$$\Longrightarrow b^2 = \frac{6}{n_{inputnode} + n_{outputnode}}$$

$$\Longrightarrow b = \sqrt{\frac{6}{n_{inputnode} + n_{outputnode}}}$$

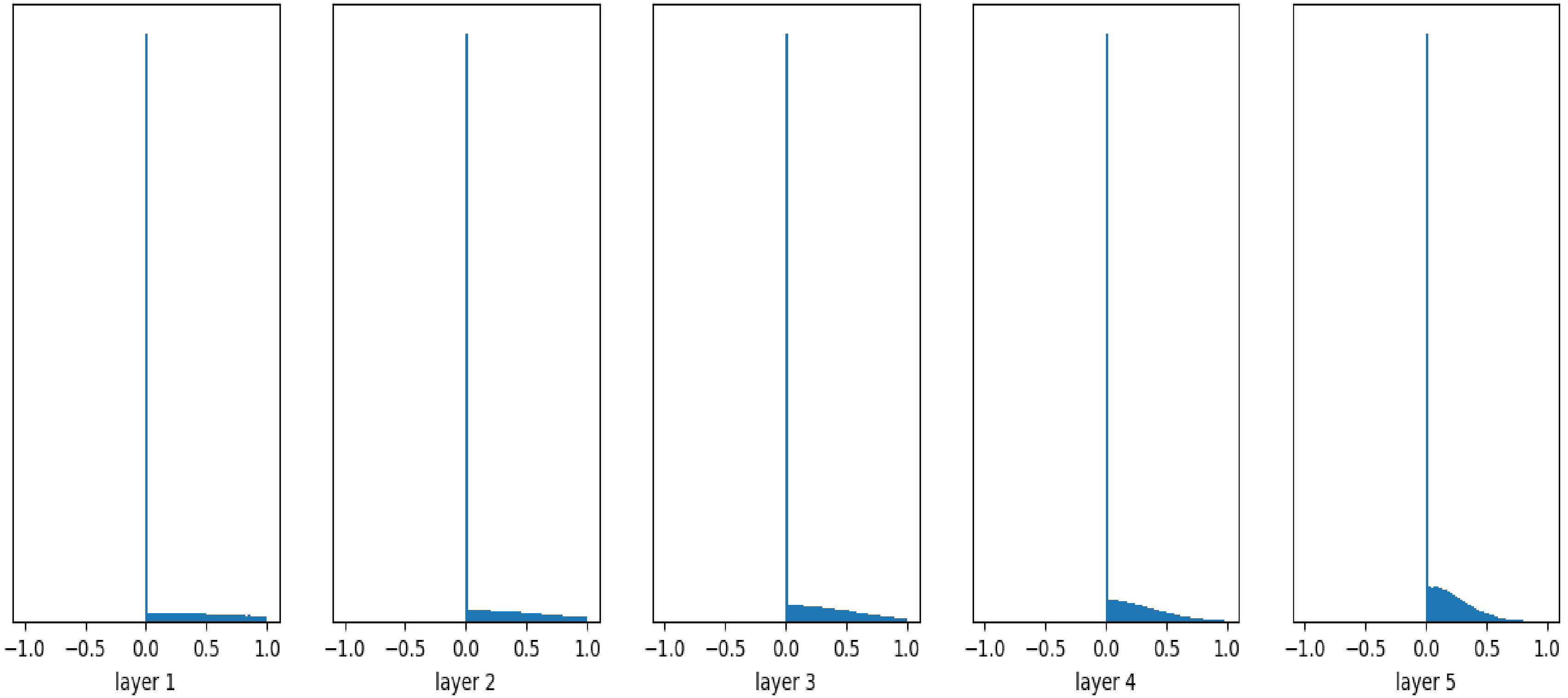
Xavier initialization

實驗一: 用Xavier initialization，神經網路activation function採用tanh輸出。



Xavier initialization

實驗二: 用Xavier initialization，神經網路activation function採用ReLU輸出。後深層值會越接近0。



He initialization

He initialization為何證明的文章Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification

$$Var(y) = d \times Var(w_i)Var(x_i)$$

因為He initialization的推導稍微複雜一點，需要引入前層和當層的關係，所以我將公式修改成

$$Var(y_l) = n_l \times Var(w_l)Var(x_l)$$

- y_l:第l層的輸出
- n_l: 第l層輸入神經元數量
- w_l: 第l層的權重
- x_l: 第l層的輸入，且$x_l = f(y_{l-1}), f$: ReLU

He initialization

w_{l-1}是對稱於0的分佈，所以y_{l-1}的結果也是對稱於0的分佈(平均數等於0)

$$Var(x_l) = Var\big(f(y_{l-1})\big) = \frac{1}{2}Var(y_{l-1})$$

$$Var(y_l) = \frac{n_l}{2} \times Var(w_l)Var(y_{l-1})$$

$$Var(y_L) = \frac{n_L}{2} \times Var(w_L)Var(y_{L-1})$$

$$= \frac{n_L}{2} \times Var(w_L) \times \frac{n_{L-1}}{2}Var(w_{L-1})Var(y_{L-2}) = \cdots$$

$$= Var(y_1)\left(\prod_{l=2}^{L}\frac{n_l}{2}Var(w_l)\right)$$

只要這個variance大於1或是小於1都會因為層數增加造成vanish或是explosion

He initialization

$$\frac{n_l}{2} Var(w_l) = 1 \Rightarrow Var(w_l) = \frac{2}{n_l}, \forall l$$

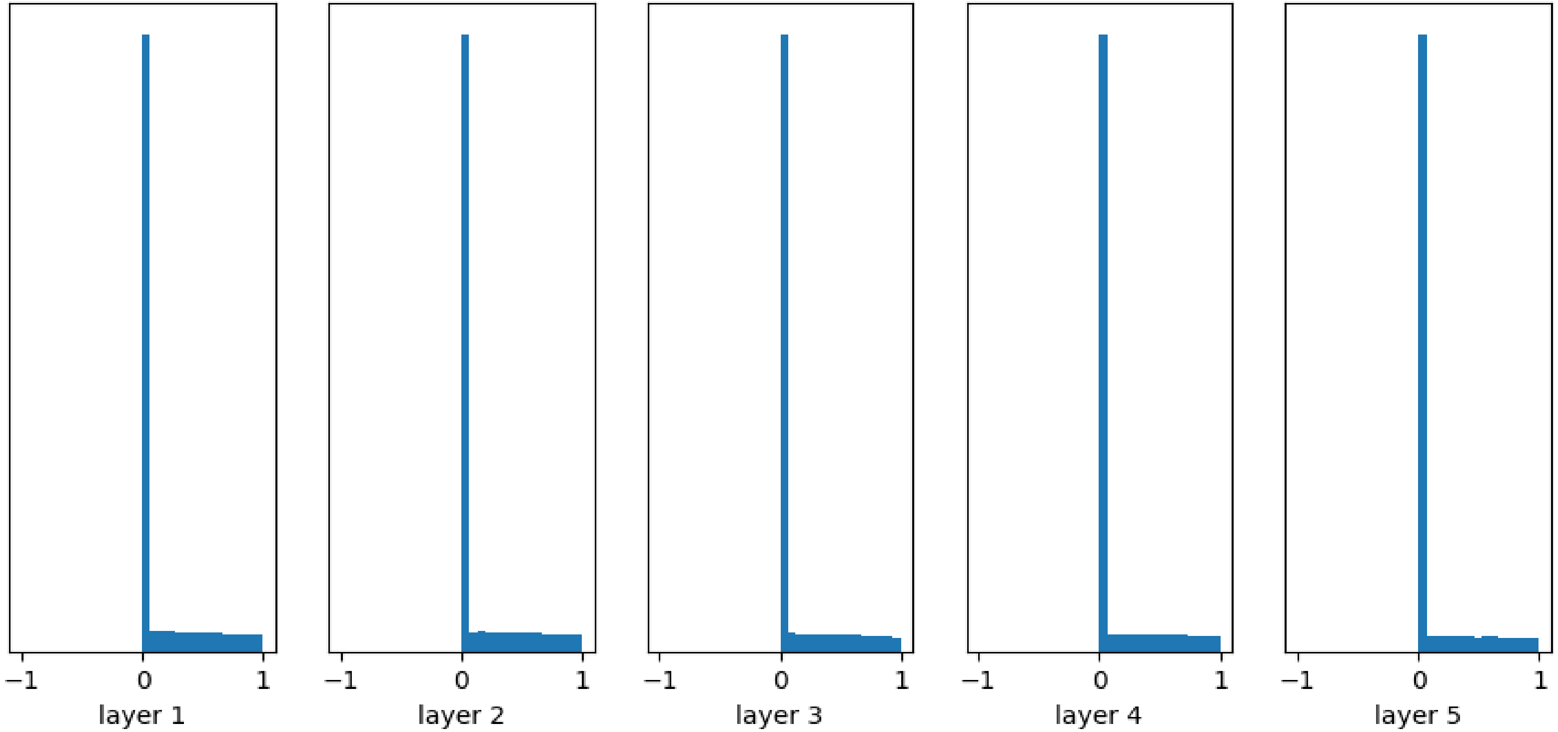
$$w_i \overset{iid}{\sim} U(-b, b\ )$$

$$var(w_i) = \frac{(b-(-b))^2}{12} = \frac{2}{n_l} \Rightarrow b^2 = \frac{6}{n_l} \Rightarrow b = \sqrt{\frac{6}{n_l}}$$

$$w \sim U\left(-\sqrt{\frac{6}{n_l}}, \sqrt{\frac{6}{n_l}}\right)$$

He initialization

實驗: 剛剛 Xavier initialization發生問題的例子，改用He initialization，神經網路activation function採用ReLU輸出。



Batch Normalization

- Weight Initialization前面說了這麼多，不外乎是要怎麼有效決定weight生成時分佈的參數。
- 有沒有不管參數生成方法都可以避免發生問題的方法: Batch Normalization(BN)。(Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift)
- 我們剛剛前面都希望輸出的變異數等於1 ($Var(y) = 1$)

最簡單的方式統計學的z-score

假設資料是$x \sim N(\mu, \sigma)$

$$\frac{x-\mu}{\sigma} \sim N(0,1)$$

Batch Normalization

Input: Values of x over a mini-batch: $\mathcal{B} = \{x_{1...m}\}$;
Parameters to be learned: γ, β

Output: $\{y_i = \mathrm{BN}_{\gamma,\beta}(x_i)\}$

$$\mu_{\mathcal{B}} \leftarrow \frac{1}{m}\sum_{i=1}^{m} x_i \qquad \text{// mini-batch mean}$$

$$\sigma_{\mathcal{B}}^2 \leftarrow \frac{1}{m}\sum_{i=1}^{m} (x_i - \mu_{\mathcal{B}})^2 \qquad \text{// mini-batch variance}$$

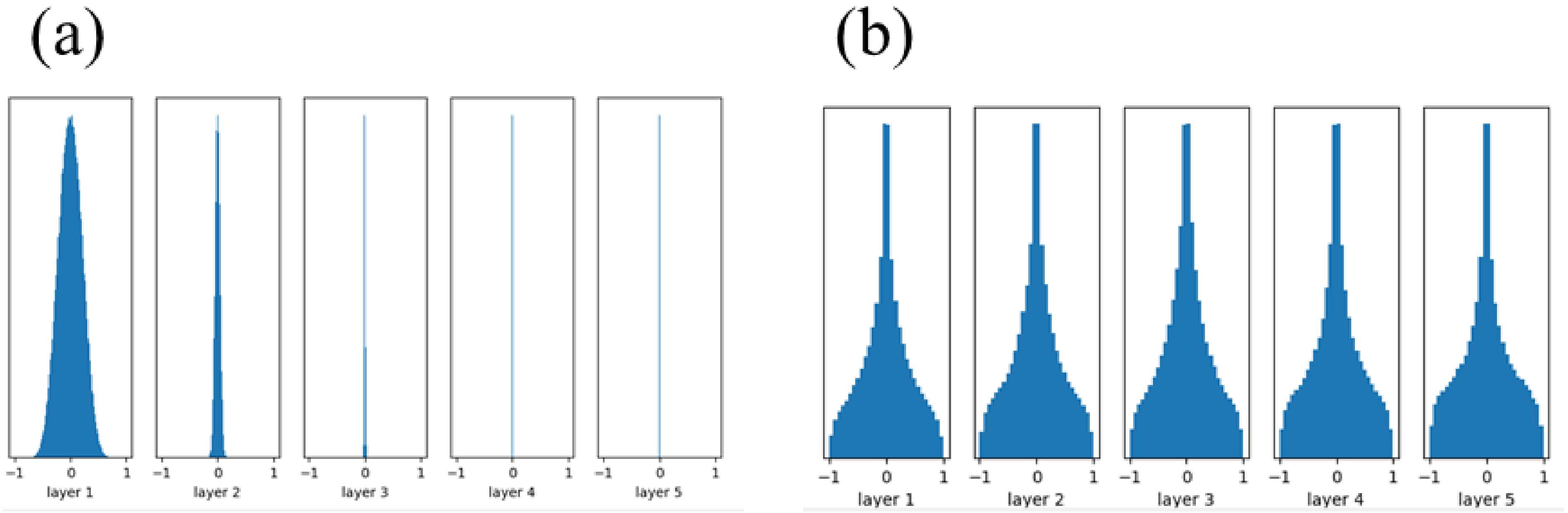
$$\widehat{x}_i \leftarrow \frac{x_i - \mu_{\mathcal{B}}}{\sqrt{\sigma_{\mathcal{B}}^2 + \epsilon}} \qquad \text{// normalize}$$

$$y_i \leftarrow \gamma \widehat{x}_i + \beta \equiv \mathrm{BN}_{\gamma,\beta}(x_i) \qquad \text{// scale and shift}$$

強制把值拉到標準常態:$\frac{x-\mu}{\sigma} \sim N(0,1)$

Batch Normalization

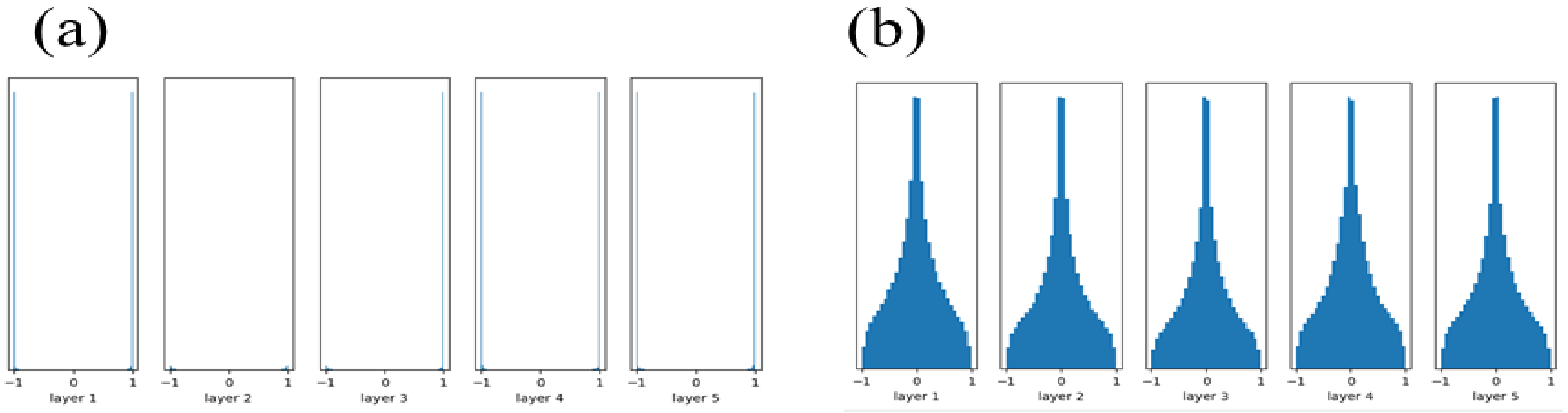
實驗1. Weight是由常態分佈隨機生成(平均數為0，標準差為0.01)。



(a) 神經網路沒有加Batch normalization，(b)神經網路加入Batch normalization。

Batch Normalization

實驗2. Weight是由常態分佈隨機生成(平均數為0，標準差為1)。



(a) 神經網路沒有加Batch normalization，(b)神經網路加入Batch normalization。