

深度學習參考手冊

Afshine AMIDI 和 Shervine AMIDI

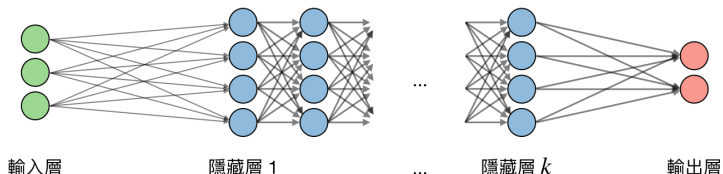
December 15, 2019

翻譯: kevingo. 審閱: TobyOoO.

神經網路

神經網路是一種透過layer來建構的模型。經常被使用的神經網路模型包括了卷積神經網路(CNN)和遞迴式神經網路(RNN)。

□ **架構** – 神經網路架構所需要用到的詞彙描述如下：



我們使用 i 來代表網路的第 i 層、 j 來代表某一層中第 j 個隱藏神經元的話，我們可以得到下面得等式：

$$z_j^{[i]} = w_j^{[i]T} x + b_j^{[i]}$$

其中，我們分別使用 w 來代表權重、 b 代表偏差項、 z 代表輸出的結果。

□ **激活函數** – 激活函數是為了在每一層尾端的神經元帶入非線性轉換而設計的。底下是一些常見激活函數：

Sigmoid	Tanh	ReLU	Leaky ReLU
$g(z) = \frac{1}{1 + e^{-z}}$	$g(z) = \frac{e^z - e^{-z}}{e^z + e^{-z}}$	$g(z) = \max(0, z)$	$g(z) = \max(\epsilon z, z)$ 與 $\epsilon \ll 1$

□ **交叉熵損失函式** – 在神經網路中，交叉熵損失 $L(z, y)$ 通常如下定義：

$$L(z, y) = - \left[y \log(z) + (1 - y) \log(1 - z) \right]$$

□ **學習速率** – 學習速率通常用 α 或 η 來表示，目的是用來控制權重更新的速度。學習速度可以是一個固定值，或是隨著訓練的過程改變。現在最熱門的最佳化方法叫作Adam，是一種隨著訓練過程改變學習速率的最佳化方法。

□ **反向傳播演算法** – 反向傳播演算法是一種在神經網路中用來更新權重的方法，更新的基準是根據神經網路的實際輸出值和期望輸出值之間的關係。權重的導數是根據連鎖律(chain rule)來計算，通常會表示成下面的形式：

$$\frac{\partial L(z, y)}{\partial w} = \frac{\partial L(z, y)}{\partial a} \times \frac{\partial a}{\partial z} \times \frac{\partial z}{\partial w}$$

因此，權重會透過以下的方式來更新：

$$w \leftarrow w - \eta \frac{\partial L(z, y)}{\partial w}$$

□ **更新權重** – 在神經網路中，權重的更新會透過以下步驟進行：

- **步驟一**：取出一個批次(batch)的資料
- **步驟二**：執行前向傳播演算法(forward propagation)來得到對應的損失值
- **步驟三**：將損失值透過反向傳播演算法來得到梯度
- **步驟四**：使用梯度來更新網路的權重

□ **丟棄法** – 丟棄法是一種透過丟棄一些神經元，來避免過擬和的技巧。在實務上，神經元會透過機率值的設定來決定要丟棄或保留

卷積神經網路

□ **卷積層的需求** – 我們使用 W 來表示輸入資料的維度大小、 F 代表卷積層的filter尺寸、 P 代表對資料墊零(zero padding)使資料長度齊一後的長度、 S 代表卷積後取出的特徵stride數量，則輸出的維度大小可以透過以下的公式表示：

$$N = \frac{W - F + 2P}{S} + 1$$

□ **批次正規化(Batch normalization)** – 它是一個藉由 γ, β 兩個超參數來正規化每個批次 $\{x_i\}$ 的過程。每一次正規化的過程，我們使用 μ_B, σ_B^2 分別代表平均數和變異數。請參考以下公式：

$$x_i \leftarrow \gamma \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \epsilon}} + \beta$$

批次正規化的動作通常在全連接層/卷積層之後、在非線性層之前進行。目的在於接納更高的學習速率，並且減少該批次學習初期對取樣資料特徵的依賴性。

遞歸神經網路(RNN)

□ **閘的種類** – 在傳統的遞歸神經網路中，你會遇到幾種閘：

輸入閘	遺忘閘	輸出閘	閘
要不要將資料寫入到記憶區塊中？	要不要將存在在記憶區塊中的資料清除？	要不要將資料從記憶區塊中取出？	要寫多少資料到記憶區塊？

□ **長短期記憶模型** – 長短期記憶模型是一種遞歸神經網路，藉由導入遺忘閘的設計來避免梯度消失的問題

強化學習及控制

強化學習的目標就是為了讓代理(agent) 能夠學習在環境中進化

□ **馬可夫決策過程** – 一個馬可夫決策過程(MDP) 包含了五個元素($\mathcal{S}, \mathcal{A}, \{P_{sa}\}, \gamma, R$)：

- \mathcal{S} 是一組狀態的集合
- \mathcal{A} 是一組行為的集合
- $\{P_{sa}\}$ 指的是，當 $s \in \mathcal{S}$ 、 $a \in \mathcal{A}$ 時，狀態轉移的機率
- $\gamma \in [0, 1]$ 是衰減係數
- $R: \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ 或 $R: \mathcal{S} \rightarrow \mathbb{R}$ 指的是獎勵函數，也就是演算法想要去最大化的目標函數

□ **策略** – 一個策略 π 指的是一個函數 $\pi: \mathcal{S} \rightarrow \mathcal{A}$ ，這個函數會將狀態映射到行為

注意：我們會說，我們給定一個策略 π ，當我們給定一個狀態 s 我們會採取一個行動 $a = \pi(s)$

□ **價值函數** – 給定一個策略 π 和狀態 s ，我們定義價值函數 V^π 為：

$$V^\pi(s) = E \left[R(s_0) + \gamma R(s_1) + \gamma^2 R(s_2) + \dots | s_0 = s, \pi \right]$$

□ **貝爾曼方程** – 最佳的貝爾曼方程是將價值函數 V^{π^*} 和策略 π^* 表示為：

$$V^{\pi^*}(s) = R(s) + \max_{a \in \mathcal{A}} \gamma \sum_{s' \in \mathcal{S}} P_{sa}(s') V^{\pi^*}(s')$$

注意：對於給定一個狀態 s ，最佳的策略 π^* 是：

$$\pi^*(s) = \operatorname{argmax}_{a \in \mathcal{A}} \sum_{s' \in \mathcal{S}} P_{sa}(s') V^*(s')$$

□ **價值迭代演算法** – 價值迭代演算法包含兩個步驟：

- 1) 針對價值初始化：

$$V_0(s) = 0$$

- 根據之前的值，迭代此價值的值：

$$V_{i+1}(s) = R(s) + \max_{a \in \mathcal{A}} \left[\sum_{s' \in \mathcal{S}} \gamma P_{sa}(s') V_i(s') \right]$$

□ **最大似估計** – 針對狀態轉移機率的最大似估計為：

$$P_{sa}(s') = \frac{\# \text{從狀態 } s \text{ 到 } s' \text{ 所採取行為的次數}}{\# \text{從狀態 } s \text{ 所採取行為的次數}}$$

□ **Q學習演算法** – Q學習演算法是針對 Q 的一個model-free 的估計，如下：

$$Q(s, a) \leftarrow Q(s, a) + \alpha \left[R(s, a, s') + \gamma \max_{a'} Q(s', a') - Q(s, a) \right]$$