類神經控制期末專題

比較逆傳學習法與強化學習法

系所：生機四
學號：B06611032
姓名：武敬祥

OUTLINE

Backpopagation learning inroduce

- Gradient descent

梯度下降法是一種最佳化的演算法，也是目前最常用來最佳化類神經網路的演算法，一般形式如下:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla L(\theta_t)$$

$\theta_{(N\times1)}$：參數向量(N 個參數)

α：learning rate

L：Loss function

t：迭代次數

令參數數量 N = 1，也就是說 Loss function(L) 會退化成二元一次方程式，如此一來就可以以一個二維平面來圖像化表示 gradient descent 在經過多次迭代後，cost 值是如何下降的(Figure 1)。

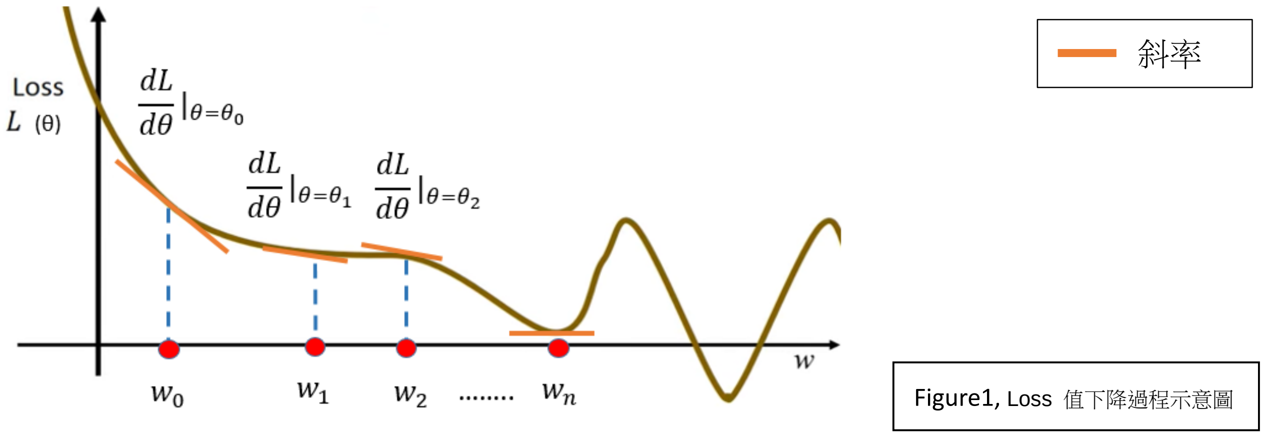

Figure1, Loss 值下降過程示意圖

$$\theta_1 = \theta_0 - \alpha \cdot \frac{dL}{d\theta}|_{\theta=\theta_0}$$

$$\theta_2 = \theta_1 - \alpha \cdot \frac{dL}{d\theta}|_{\theta=\theta_1}$$

$$\theta_n = \theta_{n-1} - \alpha \cdot \frac{dL}{d\theta}|_{\theta=\theta_{n-1}}$$

$$\vdots$$

$$\theta_{n+1} = \theta_n - \alpha \cdot \frac{dL}{d\theta}|_{\theta=\theta_n} \qquad \theta_{n+1} = \theta_n$$

其中迭代到第 n 次(t = n) 的時候 Loss function 的值 $L(\theta_n)$ 已經達到 local minimum 了，同時斜率也會達到 0，這時候θ就不會有任何更動了。

- Efficient compute the gradient in neural network

在類神經網路(NN)中，因為參數數量很多，且層與層之間參數互相影響，要有效率的使用 gradient descent 來達到最佳化的目的，因此就有了反向傳播法(backpropogation)這個演算法。

- Updating weight

假設使用的神經網路及神經網路中的各個參數如下圖所示:

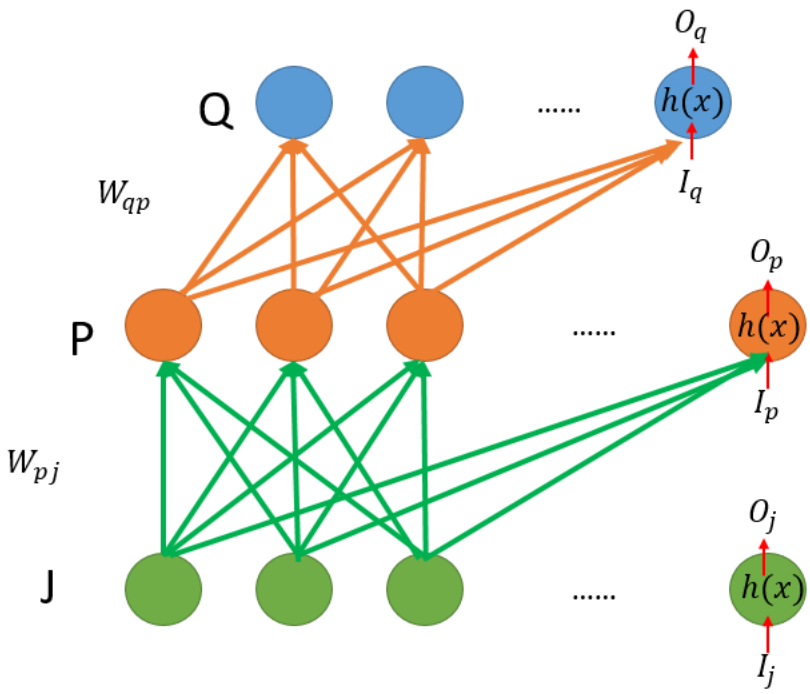

Figure2,NN structure

h(x) : activation function

W_{pj} :P -layer 和 J-layer 之間的 weight

W_{qp} :Q-layer 和 P-layer 之間的 weight

O_k : k-layer 的 output

I_k : k-layer 的 input

更新 W 的過程可以用下面的算式表示

由δ_p及δ_q的關係可以看出 backpropagation 是由最外層的δ一層一層往前推，weight 也同時一層層的往前更新

Update the W_{qp} :

$$W_{qp}^{k+1} = W_{qp}^{k} - \alpha \frac{\partial E}{\partial W_{qp}}$$

$$E = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2 \quad O_q = h(I_q) \quad I_q = \sum_{p=1}^{N_P} W_{qp}O_p$$

$$\frac{\partial E}{\partial W_{qp}} = \frac{\partial E}{\partial O_q}\frac{\partial O_q}{\partial I_q}\frac{\partial I_q}{\partial W_{qp}} = -(r_q - O_q)h'(I_q)O_p$$

令$(r_q - O_q)h'(I_q) = \delta_q$

$$W_{qp}^{k+1} = W_{qp}^{k} - \alpha \frac{\partial E}{\partial W_{qp}} = W_{qp}^{k} + \alpha\delta_q O_p$$

Update the W_{pj} :

$$W_{pj}^{k+1} = W_{pj}^{k} - \alpha \frac{\partial E}{\partial W_{pj}}$$

$$E = \frac{1}{2}\sum_{q=1}^{N_Q}(r_q - O_q)^2 \quad O_p = h(I_p) \quad I_p = \sum_{j=1}^{N_J} W_{pj}O_j$$

$$\frac{\partial E}{\partial W_{pj}} = \frac{\partial E}{\partial O_p}\frac{\partial O_p}{\partial I_p}\frac{\partial I_p}{\partial W_{pj}} = \frac{\partial E}{\partial O_p}h'(I_p)O_j$$

$$\frac{\partial E}{\partial O_p} = \frac{\partial E}{\partial O_q}\frac{\partial O_q}{\partial I_q}\frac{\partial I_p}{\partial O_p} = -\underbrace{(r_q - O_q)h'(I_q)}_{\delta_q}W_{qp}$$

令$\sum_{q=1}^{N_q}\delta_q W_{qp}\, h'(I_p) = \delta_p$

$$W_{pj}^{k+1} = W_{pj}^{k} - \alpha \frac{\partial E}{\partial W_{pj}} = W_{pj}^{k} + \alpha\delta_p O_j$$

由δ_p及δ_q的關係可以看出 backpropagation 是由最外層的δ一層一層往前推，weight 也同時一層層的往前更新

Reinforcement learning introduce

- Structure

強化學習法(reinforcement learing)是一種交互式的學習方法(Figure3)，主要是由 agent 下達動作指令(action)，environment 接收指令後會發出經由這個指令後產生的回饋(reward)給 agent ，同時將這個時間點觀察到的(observation)回傳給 agent。此外還有一個用來描述整個環境的狀態(state)，state 包含了整個環境所有時間點的 reward、action 及 observation，可以寫成以下形式:

$$S_t = f(o_1, r_1, a_1, \ldots., a_{t-1}, o_t)$$

Agent 再依據這個S_t去決定下一個 action 是甚麼。RL 最終的目標是要讓得到的 reward 最大化。

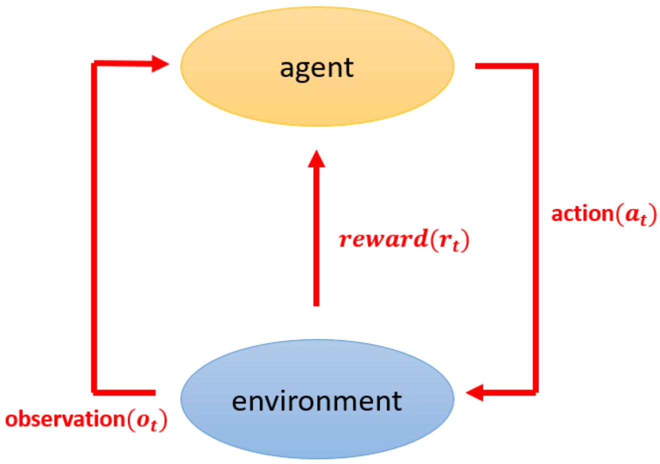

Figure3,RL structure

Goal : maximize the reward(r_t)

- markov decision process(MDP)

S_t的資訊一般來說越完整，越能夠準確地預測下一個 action 所帶來的 reward 但隨著時間的推移，資訊量過大，模型也就相對變得很大，MDP 就告訴我們，下一個時間的 state(S_{t+1})只由現在這個時間的 state(S_t)來決定，因此原本的式子

$$S_t = f(o_1, r_1, a_1, \ldots., a_{t-1}, o_t)$$

就可以簡化成:

$$P(s_{t+1}|S_t) = P(s_{t+1}|S_{1 \ldots\ldots} S_t)$$

- Action-value function

Value function 表示的是這個 state 在未來的潛在價值，也可以表示成這個 state 造成的未來所有 reward 的期望值

$$\mathrm{v(s)} = E[G_t|S_t = S]$$

其中G_t代表的是 future reward，代表所有未來 reward 的總和

$$G_t = r_{t+1} + \gamma r_{t+2} + \cdots$$

γ= discounting factor

而再將 value function 用 bellman function 做展開，就會產生以下的形式

$$\mathrm{v(s)} = E[r_{t+1} + \gamma v(S_{t+1})|S_t = S]$$

這代表著 value function 的值可以由下一個狀態的 reward 及 value function 來迭代求解。最後我們想知道的是我們施加一個 action 在這個 state 上對未來產生的 reward 總和，最後就產生了 action-value function

$$Q^{\pi}(s,a) = E[r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} \ldots |s,a]$$

在經過 bellman function 展開之後，就會產生下式

$$Q^{\pi}(s,a) = E_{s'}[r_{t+1} + \gamma Q^{\pi}(s',a')|s,a]$$

π : policy 利用這個 policy 可以決定接下來的 action。

- Q-learning

如果使$Q^{\pi}(s,a)$最大化，則代表著在這個策略下所決定的 action 使得預期的 reward 最大化

$$Q^*(s,a) = E_{s'}[r_{t+1} + \gamma \max_{a'} Q^*(s',a')\,|s,a]$$

在這個情況下，我們可以使用 value iteration 的方式，迭代取得最好的 Q 值

$$Q^*(s,a) = Q(s,a) + \alpha[r_{t+1} + \gamma \max_{a'} Q^*(s',a') - Q(s,a)]$$

既然$Q^*(s,a)$是一個 function，就代表著它也可以被一個神經網路來表示

$$\mathrm{Q(s,a,w)} \approx Q^*(s,a)$$

$$r_{t+1} + \gamma \max_{a'} Q(s',a',w) \approx r_{t+1} + \gamma \max_{a'} Q^*(s',a')$$

而我們的目標是要讓$Q^*(s,a)$ *與* $r_{t+1} + \gamma \max_{a'} Q^*(s',a')$ 越接近越好，所以可以定義 Loss function，並使用 stochastic gradient descent 得到最小的值

$$\mathrm{L(w)} = E[(r_{t+1} + \gamma \max_{a'} Q(s',a',w) - \mathrm{Q(s,a,w)})^2]$$

- Temporal difference

在做 Q-learning 的時候所使用的優化演算法是 temporal difference，下列算式是在做 Q-learning 的 pseudo code

Input: the policy π to be evaluated
Initialize $V(s)$ arbitrarily (e.g., $V(s)=0, \forall s \in \mathcal{S}^+$)
Repeat (for each episode):
  Initialize S
  Repeat (for each step of episode):
    $A \leftarrow$ action given by π for S
    Take action A, observe R, S'
    $V(S) \leftarrow V(S) + \alpha\left[R + \gamma V(S') - V(S)\right]$
    $S \leftarrow S'$
  until S is terminal

其中$\mathrm{V(S)} \leftarrow \mathrm{V(S)} + \alpha[R+\gamma V(S') - V(S)]$就是在做 temporal difference ，希望$R+\gamma V(S')$與 V(S)在經過多次迭代之後能夠逐漸一致，如果把 V(S)以 Q(s,a)取代可得到下式:

$$Q^{k+1}(s,a) = Q^k(s,a) + \alpha[r_{t+1} + \gamma\, Q(s',a') - Q^k(s,a)]$$

Compare

為了方便比較，我使用δ 取代$r_{t+1} + \gamma\, Q(s',a') - Q^k(s,a)$ 得到:

$$Q^{k+1} = Q^k + \alpha\delta$$

就可以將 reinforcement learning 與 backpropagation learning 做以下比較:

	reinforcement	backpropagation
Learning algorithm	Temporal difference	Gradient descent
Update function	$Q^{k+1} = Q^k + \alpha\delta$	$W^{k+1} = W^k + \alpha\delta o$

Reference:

- NTU ADLXMLDS
- Mnih, Volodymyr & Kavukcuoglu, Koray & Silver, David & Graves, Alex & Antonoglou, Ioannis & Wierstra, Daan & Riedmiller, Martin. (2013). Playing Atari with Deep Reinforcement Learning.
- NTU csie machine learning foundation/technique