機器與深度學習基礎知識初探
-Gradient descent

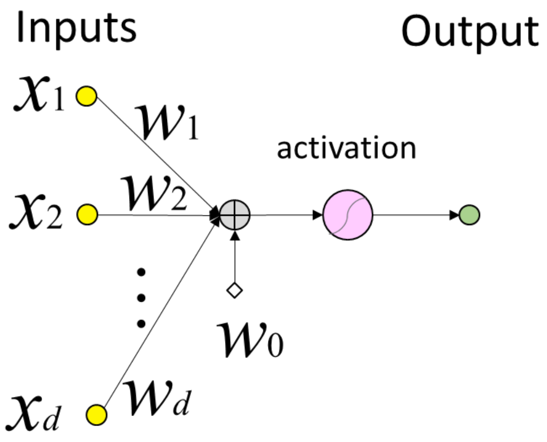
黃志勝 Chih-Sheng (Tommy) Huang

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

Introduction

神經網路求解方式為利用倒傳遞(back-propagation)方式來更新權重。



權重就是w0, w1,w2,…,wd

怎麼用Gradient descent求解?

此份投影片會利用一些簡單的解釋怎麼利用導數(Derivative)/梯度(Gradient)求最佳解。

微積分求極值

一階微分=0找解，求得的解可能為最大或最小。
二階微分判斷，一階微分找的解為最大或是最小。

$$f(x) = x^2 - 10x + 1$$

$$f'(x) = \frac{\partial f(x)}{\partial x} = 2x - 10 = 0$$

$$\Rightarrow x = 5$$

$$f''(x) = \frac{\partial f'(x)}{\partial x} = 2 > 0$$

此範例有x=5有最小值-24。

微積分求極值

上述範例為close-form可以找到解。

$$f(\boldsymbol{x}) = 0.5x_1 + 2x_2 + 4x_3 + 10$$

$f(\boldsymbol{x})$為三元一次方程式，則此函數的梯度為一個向量方程式:

$$\nabla f = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_3} \end{bmatrix} = \begin{bmatrix} 5 \\ 2 \\ 4 \end{bmatrix}$$

微積分求極值

$f(x)$為一元多次方程式，其對一元的參數作微分稱為求此參數的導數(Derivative)。

$$f'(x)$$

$f(\boldsymbol{x})$為多元多次方程式，其對多元的參數作微分稱為求此參數的梯度(Gradient)。

$$\nabla f = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \frac{\partial f(\boldsymbol{x})}{\partial x_d} \end{bmatrix}$$

Hessian Matrix

剛提到梯度順便提一下Hessian Matrix

牛頓法求解用，但相對計算量大，目前還沒被廣泛使用。

$$\mathrm{H}(x) = \nabla^2 f = \nabla \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \vdots \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_d} \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_1} & \cdots & \dfrac{\partial f(\boldsymbol{x})}{\partial x_1 \partial x_d} \\ \vdots & \ddots & \vdots \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_d \partial x_1} & \cdots & \dfrac{\partial f(\boldsymbol{x})}{\partial x_d \partial x_d} \end{bmatrix}$$

微積分求極值

上述範例為close-form可以找到解。

現實狀況

$$f(\boldsymbol{x}) = x_1^2 + x_1 - 4x_1x_2 + x_1^3x_2$$

$$\nabla f = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 + 1 - 4x_2 + 3x_1^2x_2 \\ x_1^3 - 4x_1 \end{bmatrix}$$

微積分求極值

上述範例為close-form可以找到解。

現實狀況

$$f(\boldsymbol{x}) = x_1^2 + x_1 - 4x_1x_2 + x_1^3x_2$$

$$\nabla f = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 + 1 - 4x_2 + 3x_1^2x_2 \\ -4x_1 + x_1^3 \end{bmatrix} = 0$$

$$x_1 = 0, -2, 2$$

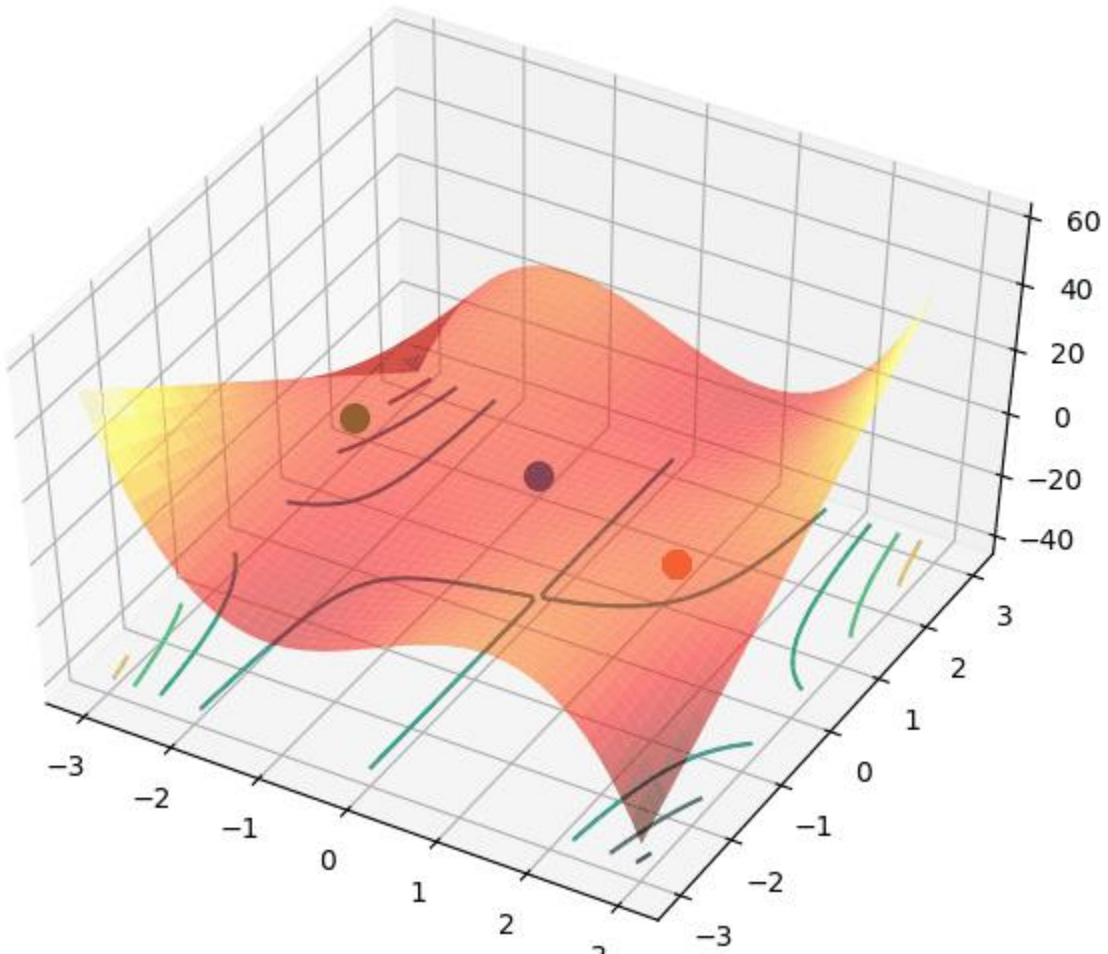
$$(x_1, x_2) = (0, 0.25), (-2, 0.125), (2, -0.625)$$

?

微積分求極值

$$f(\boldsymbol{x}) = x_1^2 + x_1 - 4x_1x_2 + x_1^3x_2$$
$$f(0, 0.25) = 0$$
$$f(-2, 0.125) = 2$$
$$f(2, -0.625) = 6$$

所以微分得到的解不一定是極值，有可能是鞍點、反曲點。



導數(Derivative)/梯度(Gradient)

- 假設有一個一元的函數為:

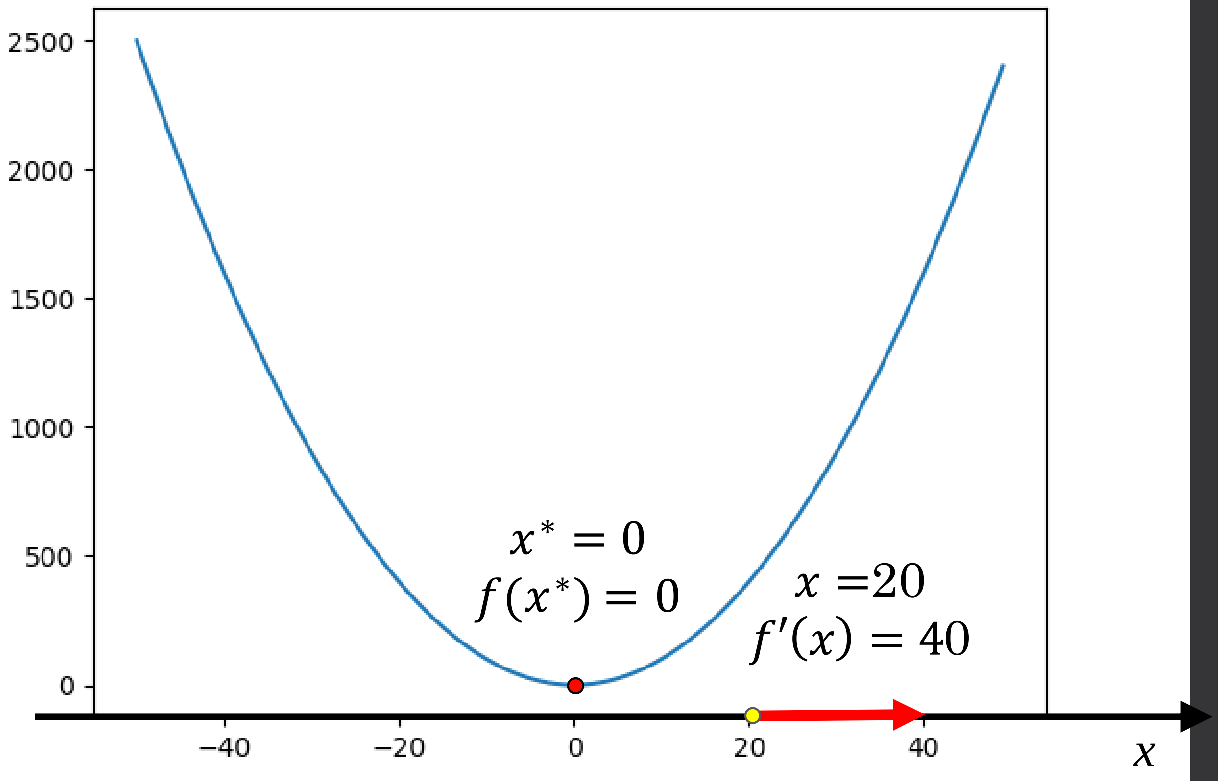
$$f(x) = x^2$$

其導數: $f'(x) = 2x$ (導數有方向性)

x>0 → 往右(正)的方向

x<0 → 往左(負)的方向

導數(Derivative)/梯度(Gradient)往極大值的方向走



導數(Derivative)/梯度(Gradient)

- 假設有一個一元的函數為:

$$f(x) = x^2$$

其導數: $f'(x) = 2x$ (導數有方向性)

$x>0 \longrightarrow$ 往右(正)的方向

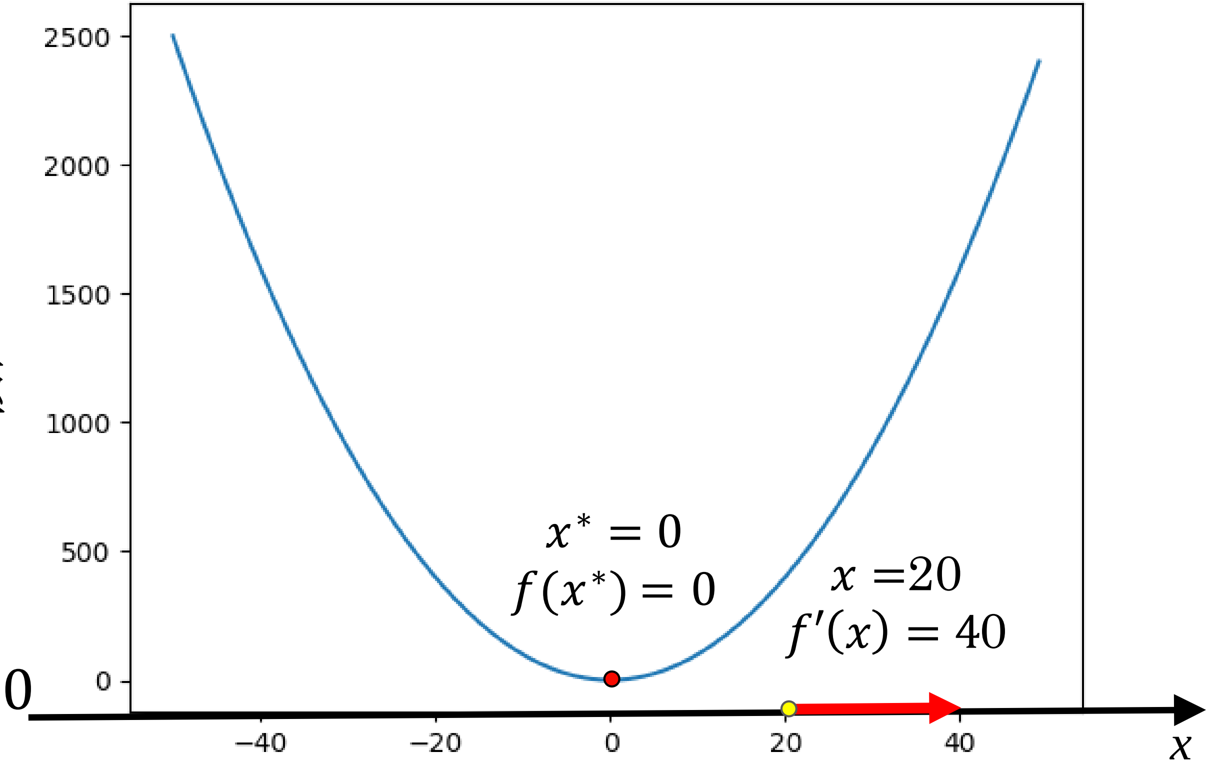
$x<0 \longrightarrow$ 往左(負)的方向

往導數(Derivative)/梯度(Gradient)往極大值的反方向走，則是往極小值

$$x^{(t+1)} = x^{(t)} - \alpha f'(x)$$

$$\alpha = 0.5$$

$$x^{(0)} = 20, f'(x) = 20, x^{(0)} = 20 - 10 = 10$$



梯度下降法(Gradient descent)

梯度(Gradient)

$$f(\boldsymbol{x}) = (x_1 - 5)^2 + (x_2)^2$$

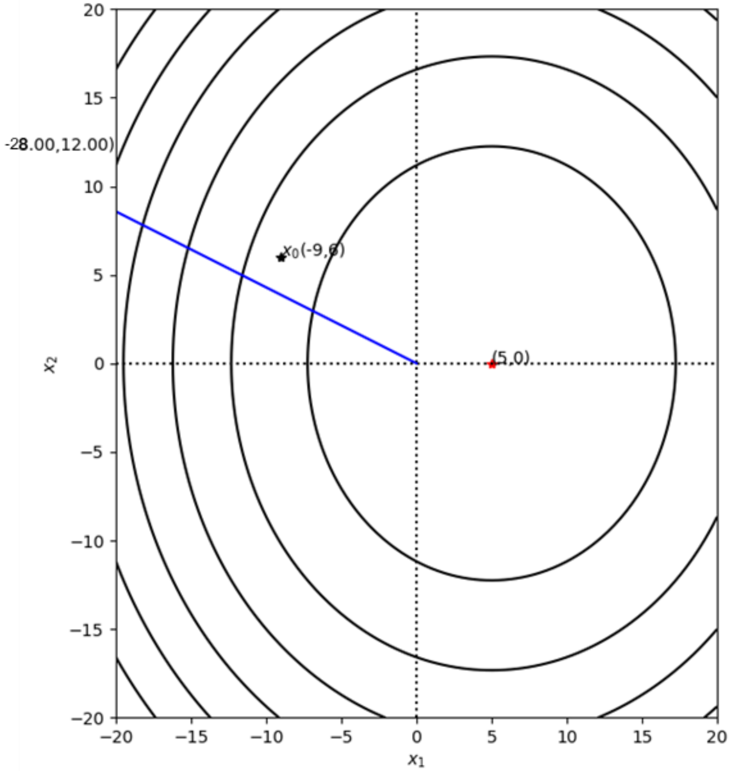
$$\nabla f = \begin{bmatrix} \dfrac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \dfrac{\partial f(\boldsymbol{x})}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 - 10 \\ 2x_2 \end{bmatrix}$$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha \nabla f\left(\boldsymbol{x}^{(\boldsymbol{t})}\right), \alpha = 0.1$$

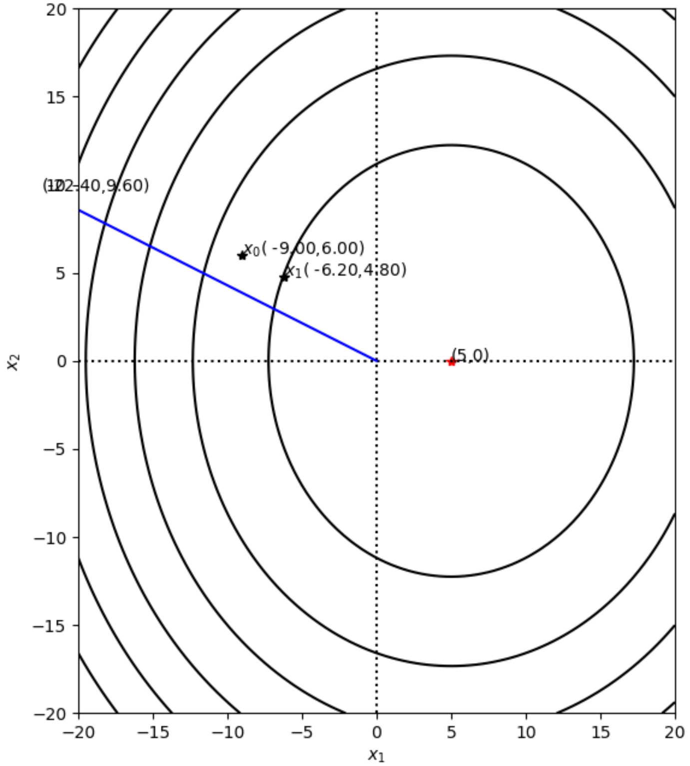
$$\boldsymbol{x}^{(\mathbf{0})} = \begin{bmatrix} -9 \\ 6 \end{bmatrix}, \nabla f\left(\boldsymbol{x}^{(\mathbf{0})}\right) = \begin{bmatrix} -28 \\ 12 \end{bmatrix}$$

$$\boldsymbol{x}^{(\mathbf{1})} = \begin{bmatrix} -6.2 \\ 4.8 \end{bmatrix}, \nabla f\left(\boldsymbol{x}^{(\mathbf{0})}\right) = \begin{bmatrix} -22.4 \\ 9.6 \end{bmatrix}$$

$$\boldsymbol{x}^{(\mathbf{0})} = \begin{bmatrix} -9 \\ 6 \end{bmatrix}$$



$$\boldsymbol{x}^{(\mathbf{1})} = \begin{bmatrix} -6.2 \\ 4.8 \end{bmatrix}$$



藍線為f的Gradient

梯度(Gradient)

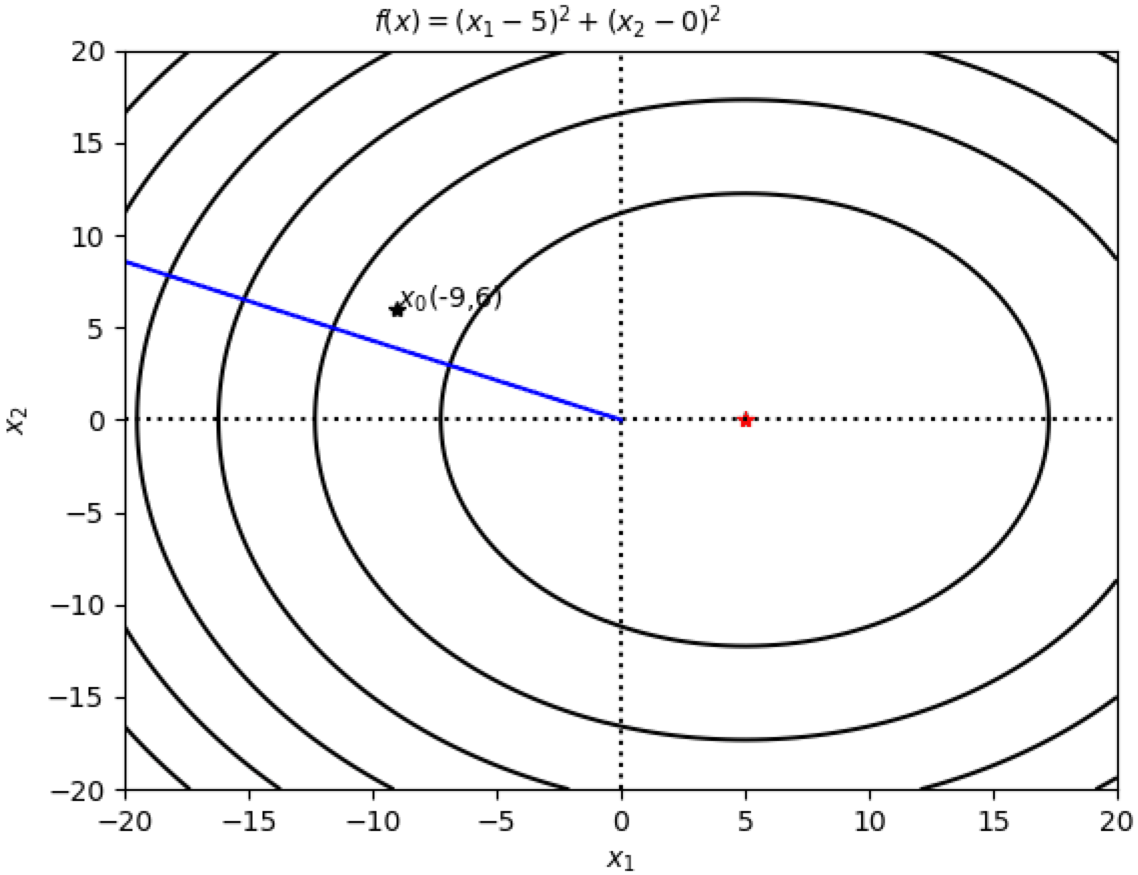
$$f(\boldsymbol{x}) = (x_1 - 5)^2 + (x_2)^2$$

$$\nabla f = \begin{bmatrix} \frac{\partial f(\boldsymbol{x})}{\partial x_1} \\ \frac{\partial f(\boldsymbol{x})}{\partial x_2} \end{bmatrix} = \begin{bmatrix} 2x_1 - 10 \\ 2x_2 \end{bmatrix}$$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \alpha \nabla f\left(\boldsymbol{x}^{(t)}\right), \alpha = 0.1$$

$$\boldsymbol{x}^{(0)} = \begin{bmatrix} -9 \\ 6 \end{bmatrix}, \nabla f\left(\boldsymbol{x}^{(0)}\right) = \begin{bmatrix} -28 \\ 12 \end{bmatrix}$$

$$\boldsymbol{x}^{(1)} = \begin{bmatrix} -6.2 \\ 4.8 \end{bmatrix}, \nabla f\left(\boldsymbol{x}^{(0)}\right) = \begin{bmatrix} -22.4 \\ 9.6 \end{bmatrix}$$



藍線為*f*的Gradient

梯度下降法(Gradient descent)

梯度下降法(Gradient descent)公式:

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \gamma \nabla f\left(\boldsymbol{x}^{(t)}\right)$$

t: 第t次迭代

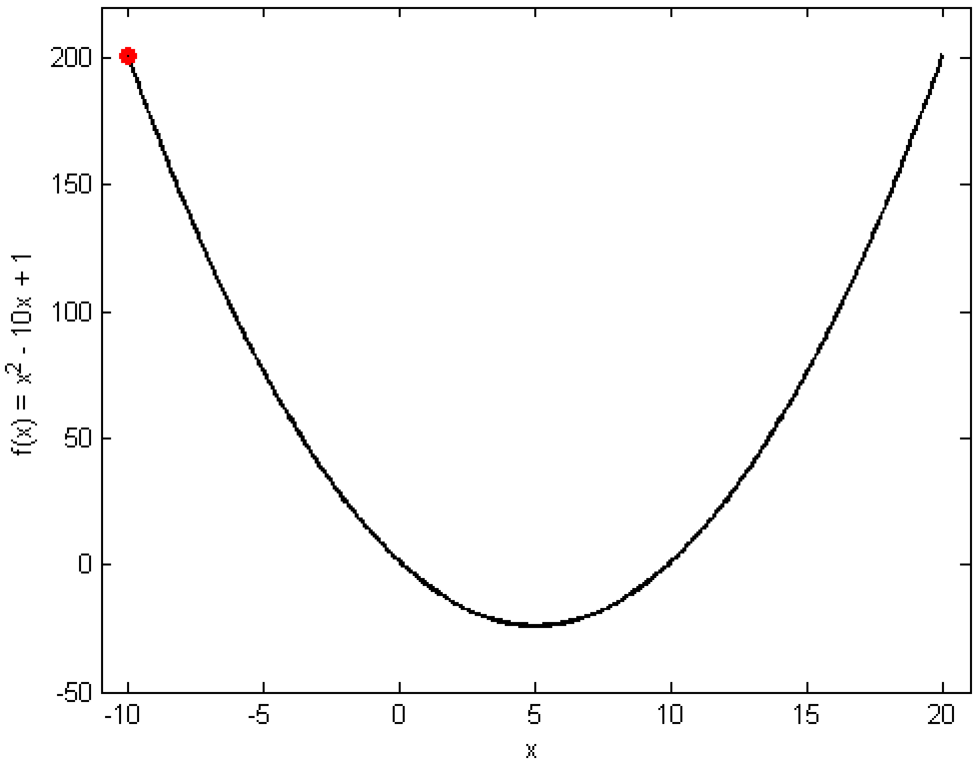
∇f:函數f的Gradient

γ: learning rate

梯度下降法-範例

$$f(x) = x^2 - 10x + 1$$
$$f'(x) = 2x - 10$$

$$x^{(t+1)} = x^{(t)} - \gamma f'\left(x^{(t)}\right)$$
$$\Rightarrow x^{(t+1)} = x^{(t)} - \gamma\left(2x^{(t)} - 10\right) = (1 - 2\gamma)x^{(t)} + 10\gamma$$
$$\Rightarrow x^{(t+1)} = (1 - 2\gamma)x^{(t)} + 10\gamma$$

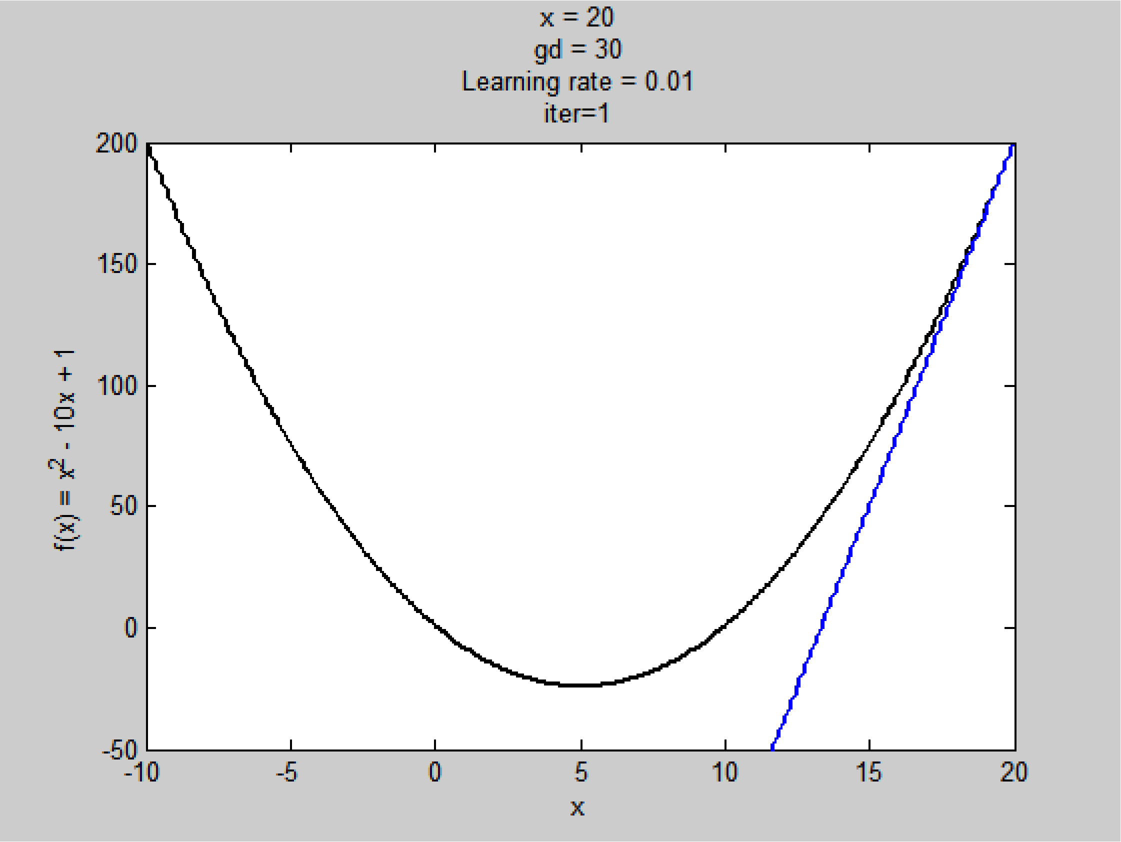


梯度下降法-範例

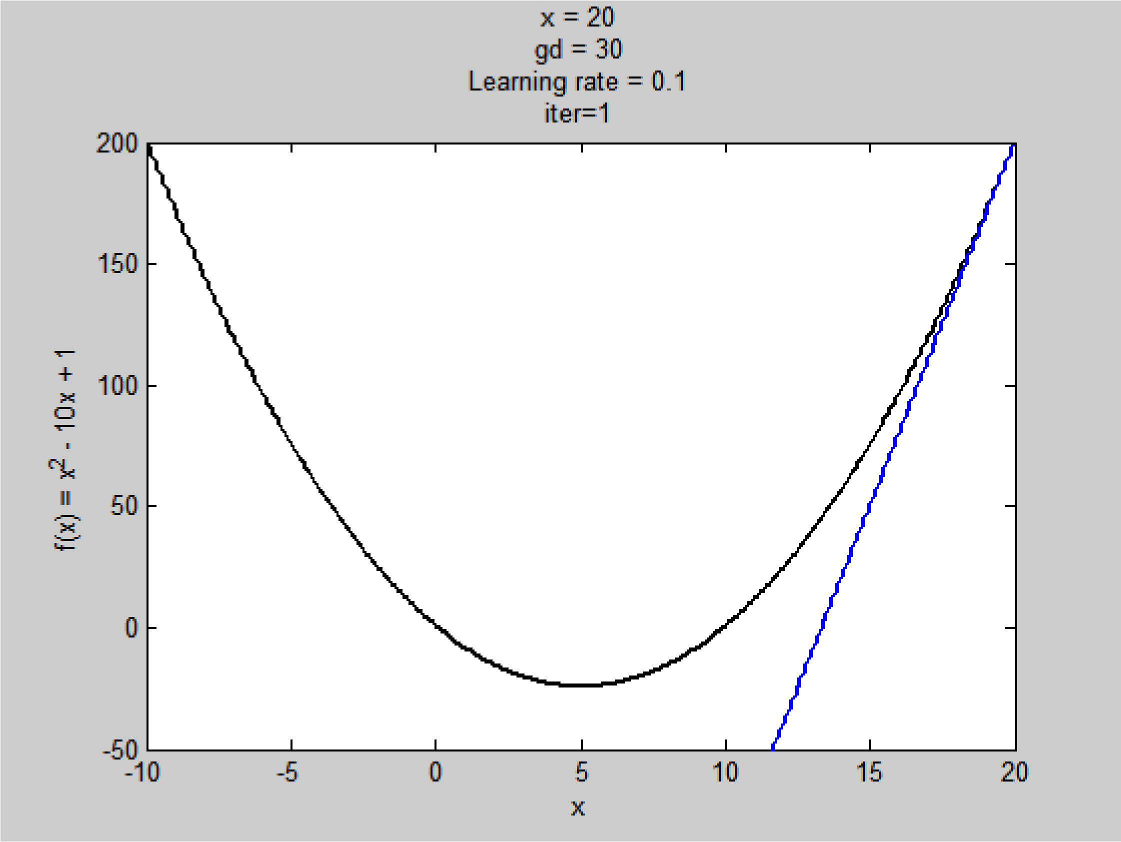
$$f(x) = x^2 - 10x + 1$$

Learning rate
越大越快找到解

Learning rate =0.01

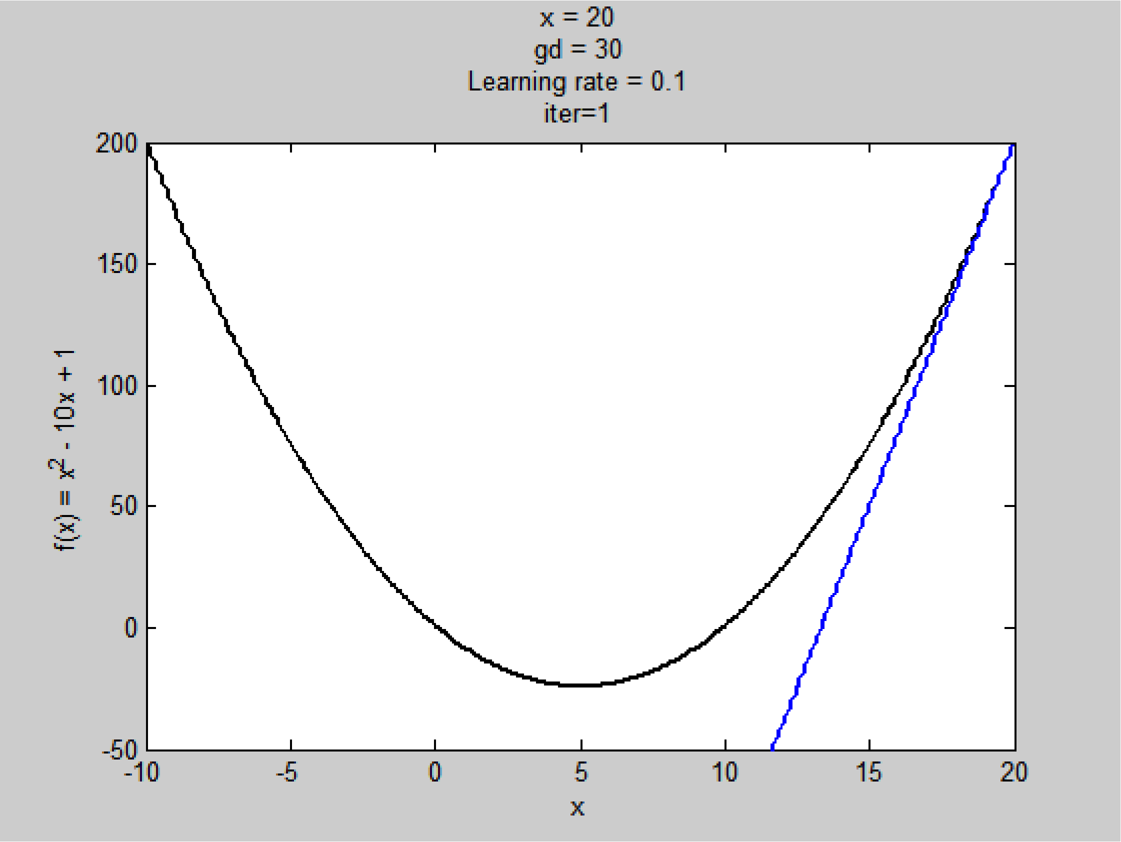


Learning rate =0.1

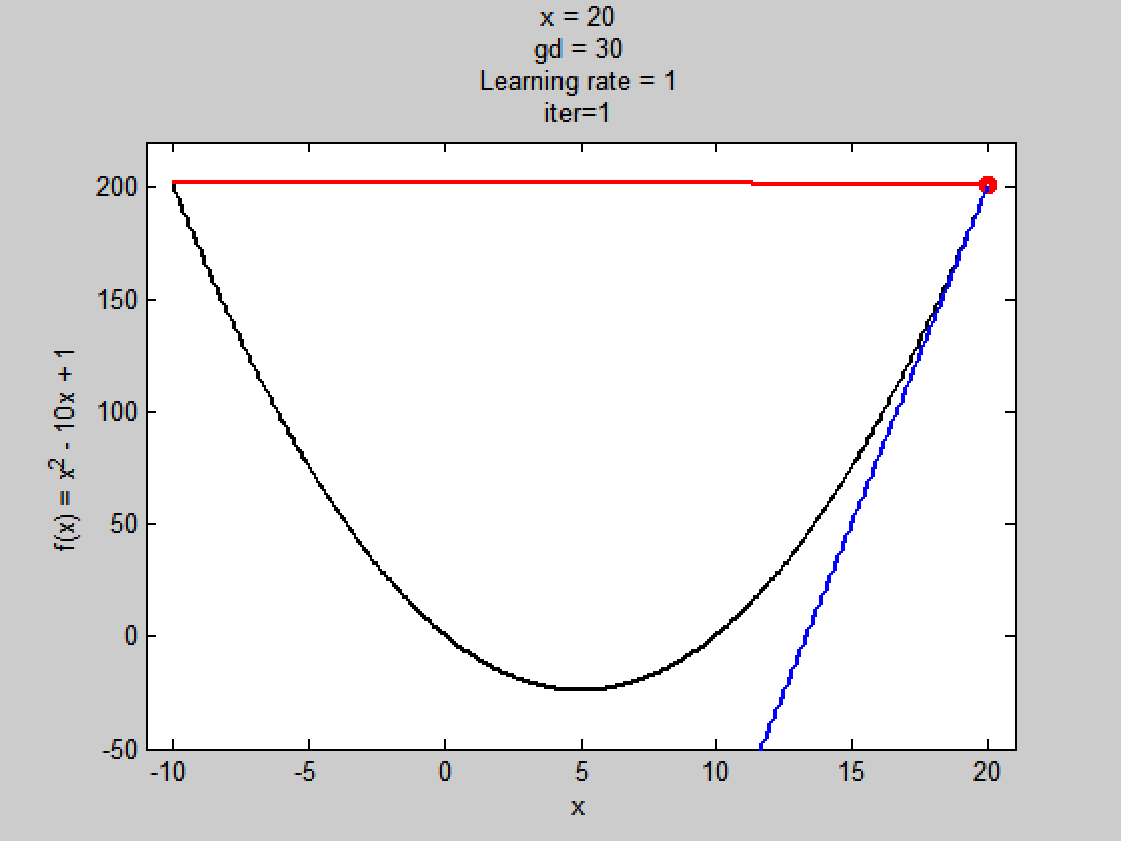


梯度下降法 (學習率過大)

Learning rate =0.1



Learning rate = 1



梯度下降法-範例2

$$f(x) = x^4 - 50x^3 - x + 1$$
$$f'(x) = 4x^3 - 150x^2 - 1$$

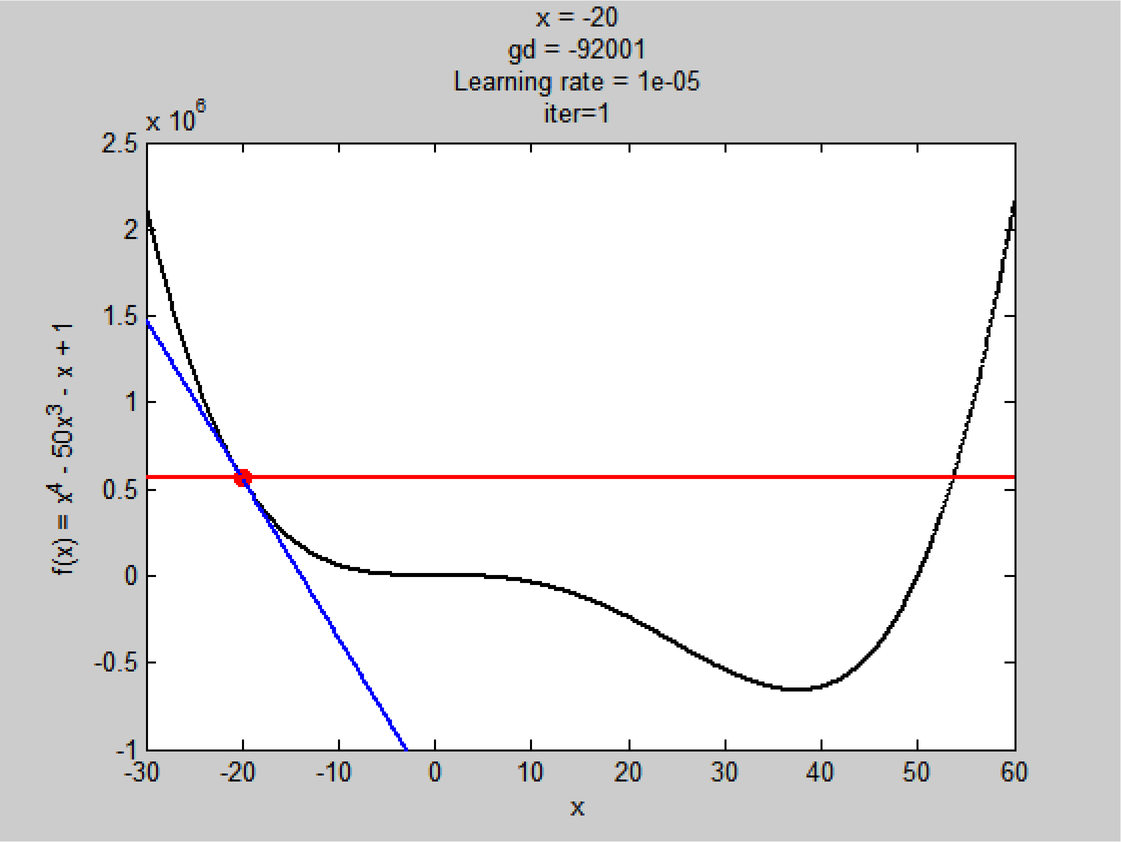
$$x^{(t+1)} = x^{(t)} - \gamma f'\left(x^{(t)}\right)$$
$$\Rightarrow x^{(t+1)} = x^{(t)} - \gamma\left(4{x^{(t)}}^3 - 150{x^{(t)}}^2 - 1\right)$$
$$\Rightarrow x^{(t+1)} = -4\gamma {x^{(t)}}^3 + 150\gamma {x^{(t)}}^2 + x^{(t)} + \gamma$$

梯度下降法-範例2

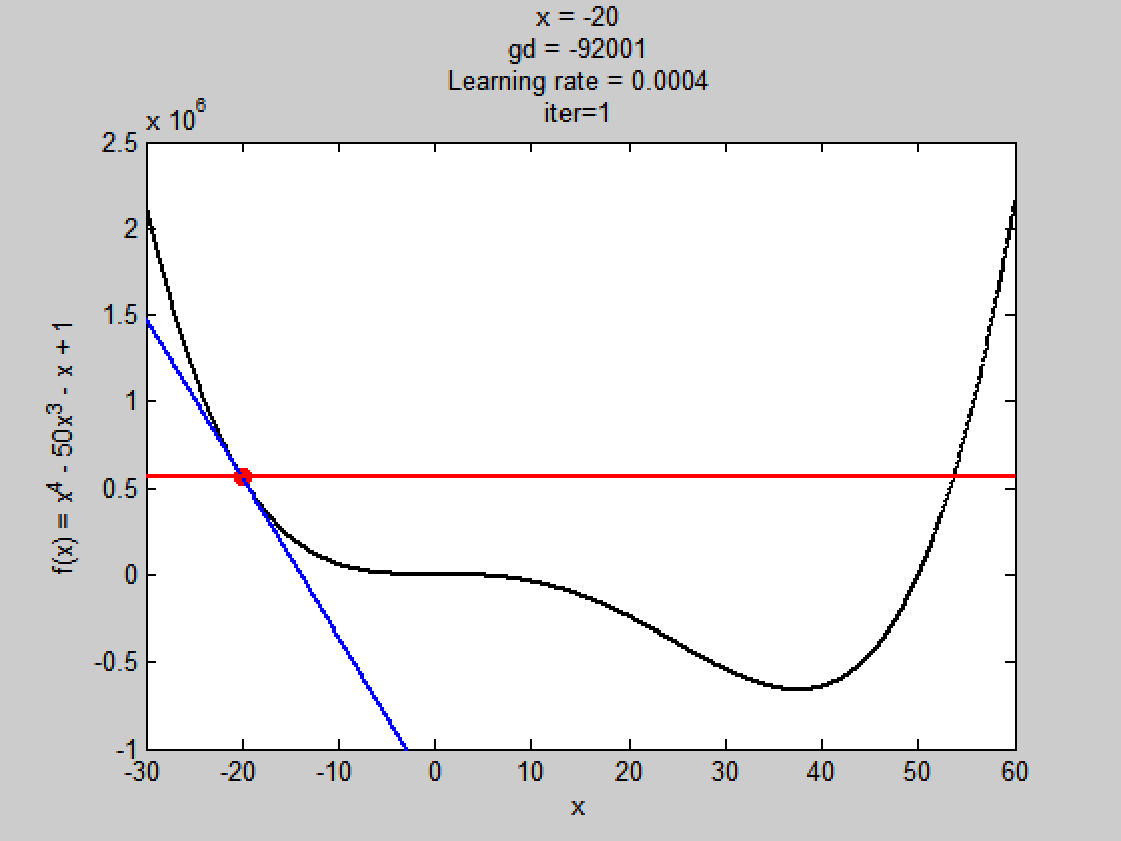
$$f(x) = x^4 - 50x^3 - x + 1$$

Learning rate 過小容易掉到 local minima

Learning rate =0.00001

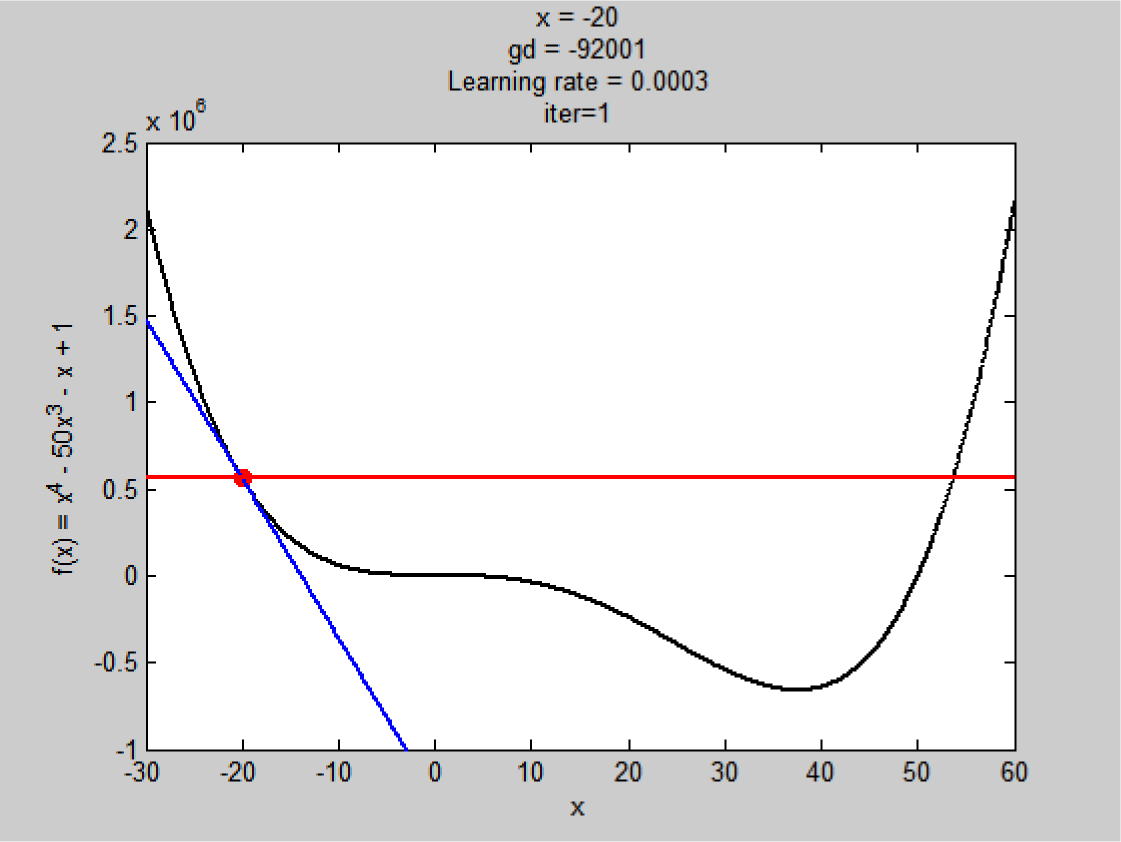


Learning rate =0.0004



梯度下降法-範例2

Learning rate =0.0003



Learning rate =0.00001
過小掉到local minima

Learning rate =0.0004
過大雖然跳出local minima，但走不進global minima

Learning rate =0.0003
最合適

梯度下降法

梯度下降法: 學習率是非常重要的一個參數因子。

因此梯度下降研究就可以展開了

梯度下降法

大家比較常看到的function為Stochastic gradient descent (SGD)、Momentum、Adagrad、RMSProp、Adam。

SGD跟前面提到GD的差異。

一般在神經網路/深度學習，訓練數量可能達到幾百萬筆

一筆資料就update模型一次(很沒有效率): 一筆資料訓練假設要1秒，一萬筆都跑過模型就要1萬秒(2.7個小時)

Mini-batch update模型一次: 假設100筆一個mini-batch，訓練要兩秒，一萬筆只需要3.3分鐘。

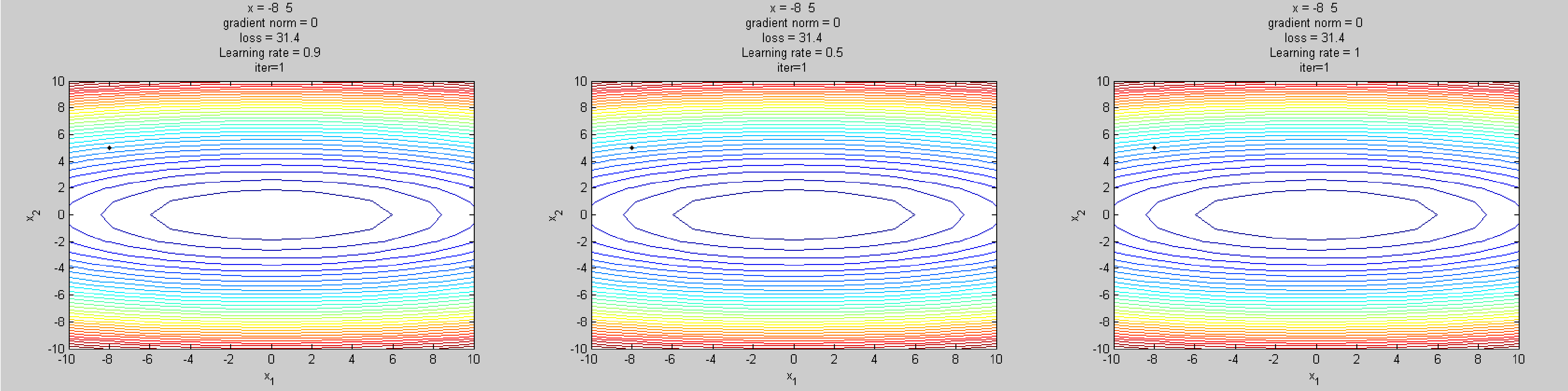
SGD: 就是一次跑一個小批次(Mini-batch)後的平均梯度模型即更新一次，這個mini-batch為隨機抽取出，所以用Stochastic。

SGD

$$f(x_1, x_2) = 0.1x_1^2 + x_2^2$$

Initial point=[-8,5]

Learning rate = 0.9 Learning rate = 0.5 Learning rate = 1



Momentum

Momentum是架構在SGD上的算法

$$\boldsymbol{v}^{(t)} = \begin{cases} \gamma \boldsymbol{g}_t & t = 0 \\ m v^{(t-1)} + \gamma \boldsymbol{g}_t & t \geq 1 \end{cases}$$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \boldsymbol{v}^{(t)}$$

$$\boldsymbol{g}_t = \nabla f\left(\boldsymbol{x}^{(t)}\right)$$

$\boldsymbol{g}_t$:第t次迭代的gradient。

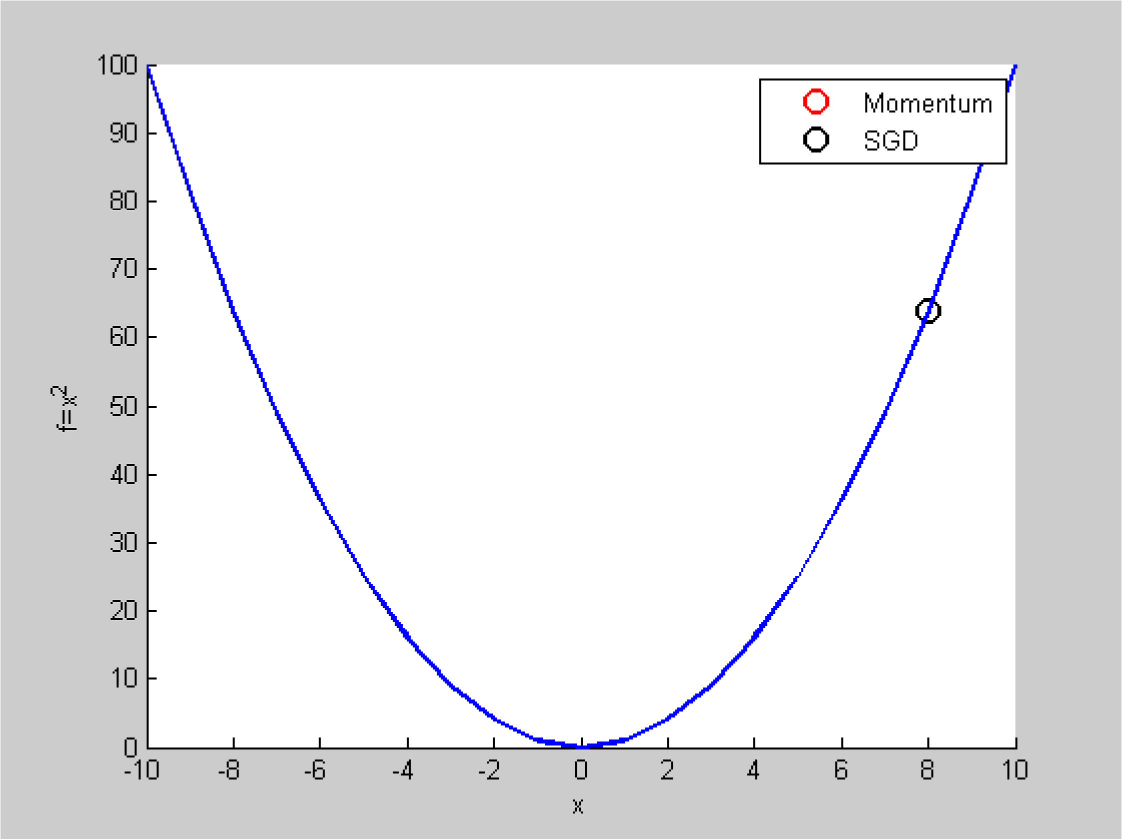
$\boldsymbol{v}^{(t)}$:第t次參數要更新的幅度(歷史的梯度和)。

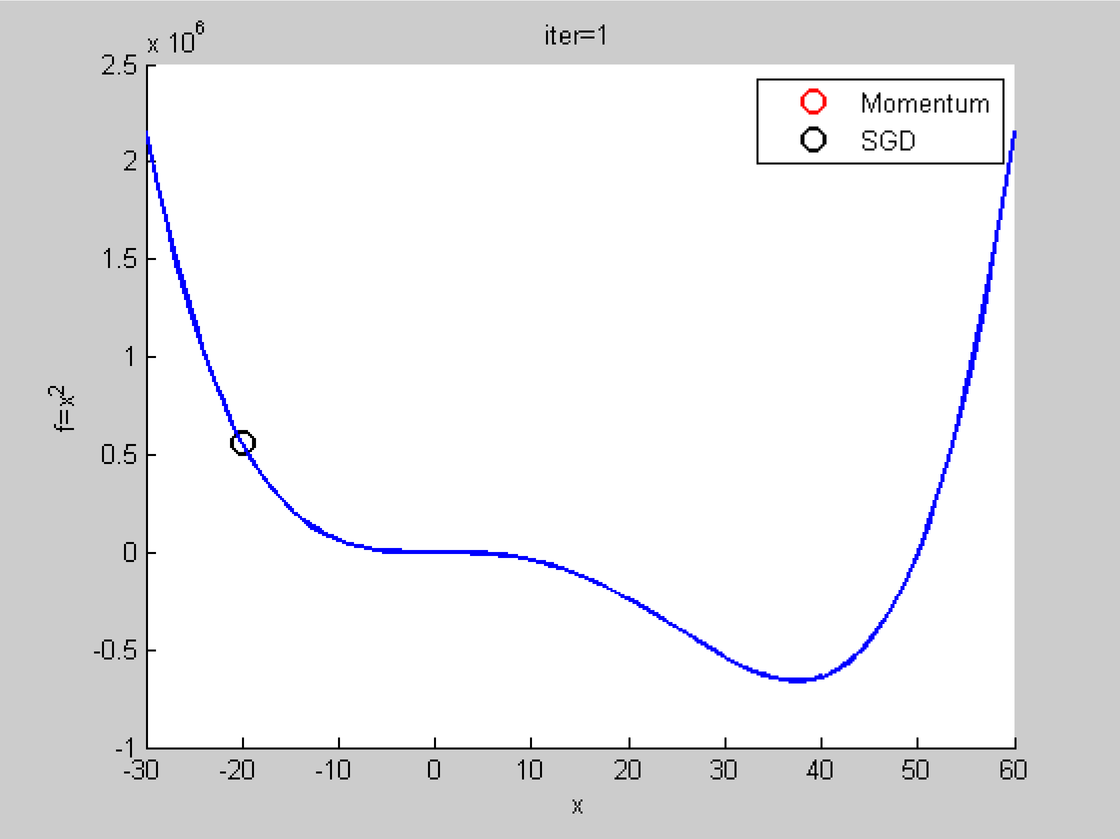
如果過去的梯度方向和當下的梯度方向一致，代表這個更新方向是對的，會增強這個方向的梯度。

如果過去的和當下方向不一致，則梯度會有消融作用(衰退)，只會微幅調整參數。

Momentum

Learning rate = 0.00001





基本上API用的SGD都有Momentum可以設定。

Adagrad

- SGD和momentum在更新參數時，都是用同一個學習率(γ)
- Adagrad算法則是在學習過程中對學習率不斷的調整，這種技巧叫做「學習率衰減(Learning rate decay)」。
- Ada這個字跟是Adaptive縮寫。

Gradient: $g_{t,i} = \nabla_{x_i} f\left(x_i^{(t)}\right)$

SGD: $x_i^{(t+1)} = x_i^{(t)} - \gamma g_{t,i}$

Adagrad : $x_i^{(t+1)} = x_i^{(t)} - \frac{\gamma}{\boxed{\sqrt{G_{t,ii}} + \varepsilon}} g_{t,i}$

Adagrad

$$x_i^{(t+1)} = x_i^{(t)} - \frac{\gamma}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i} \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d\times 1}$$

$$G_t = \begin{bmatrix} \sum_{t'=1}^{t} (g_{t',1} \times g_{t',1}) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & \sum_{t'=1}^{t} (g_{t',d} \times g_{t',d}) \end{bmatrix} \in \mathbb{R}^{d\times d}$$

$$G_{t,ii} = \sum_{t'=1}^{t} (g_{t',i} \times g_{t',i})$$

Adagrad

$$x_i^{(t+1)} = x_i^{(t)} - \frac{\gamma}{\sqrt{G_{t,ii} + \varepsilon}} g_{t,i} \qquad \boldsymbol{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_d \end{bmatrix} \in \mathbb{R}^{d\times 1}$$

$$G_{t,ii} = \sum_{t'=1}^{t} \left(g_{t',i} \times g_{t',i}\right)$$

- 隨著迭代次數越多，分母(Gradient平方和開根號)越大，學習率會越低，達到前期學習率高，後期學習率低的目的。
- 缺點: 學習中後期，分母有可能因為累積過大導致最後更新的參數趨近於0，所以無法有效學習。

RMSProp

- RMSprop是Geoff Hinton 提出未發表的方法，和Adagrad一樣是自適應的方法，但**Adagrad的分母是從第1次梯度到第t次梯度的和**，所以和可能過大，**RMSprop則是算對應的平均值**，因此可以緩解Adagrad學習率下降過快的問題。

Adagrad

$$x_i^{(t+1)} = x_i^{(t)} - \frac{\gamma}{\sqrt{G_{t,ii}} + \varepsilon} g_{t,i}$$

$$G_{t,ii} = \sum_{t'=1}^{t} \left(g_{t',i} \times g_{t',i} \right)$$

RMSprop

$$x_i^{(t+1)} = x_i^{(t)} - \frac{\gamma}{\sqrt{E\left[g_i^2\right]_t} + \varepsilon} g_{t,i}$$

$$E\left[g_i^2\right]_t = \rho E\left[g_i^2\right]_{t-1} + (1-\rho) g_{t,i}^2$$

Adam

- **Momentum**: 考慮過去梯度的方向和當前梯度的方向做合成 (沒直接修改learning rate)
- **Adagrad、 RMSprop**: 考慮過去梯度的大小用來修改learning rate (Learning rate decay)
- **Adam(Adaptive Moment Estimation)**則是兩者合併加強版本(Momentum+RMSprop+各自做偏差的修正)

Adam

Momentum

$$\boldsymbol{v}^{(t)} = \begin{cases} \gamma \boldsymbol{g}_t & t = 0 \\ \boxed{m v^{(t-1)} + \gamma \boldsymbol{g}_t} & t \geq 1 \end{cases}$$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \boldsymbol{v}^{(t)}$$

$$\boldsymbol{g}_t = \nabla f\left(\boldsymbol{x}^{(t)}\right)$$

RMSProp

$$x_i^{(t+1)} = x_i^{(t)} - \frac{\gamma}{\sqrt{E\left[g_i^2\right]_t + \varepsilon}} g_{t,i}$$

$$E\left[g_i^2\right]_t = \boxed{\rho E\left[g_i^2\right]_{t-1} + (1-\rho) g_{t,i}^2}$$

$$\boldsymbol{m}_t = \boxed{\beta_1 \boldsymbol{m}_{t-1} + (1-\beta_1) \boldsymbol{g}_t}$$

$$\boldsymbol{v}_t = \boxed{\beta_2 \boldsymbol{v}_{t-1} + (1-\beta_2) \boldsymbol{g}_t^2}$$

$$\widehat{\boldsymbol{m}}_t = \frac{\boldsymbol{m}_t}{1-\beta_1^t}$$

$$\widehat{\boldsymbol{v}}_t = \frac{\boldsymbol{v}_t}{1-\beta_2^t}$$

$$\boldsymbol{x}^{(t+1)} = \boldsymbol{x}^{(t)} - \frac{\gamma}{\sqrt{\widehat{\boldsymbol{v}}_t} + \varepsilon} \widehat{\boldsymbol{m}}_t$$

Example

$$f(x_1, x_2) = x_2^2 - x_1^2$$