Gradient Descent

Review: Gradient Descent

- In step 3, we have to solve the following optimization problem:

$$\theta^* = \arg\min_{\theta} L(\theta)$$

L: loss function θ: parameters

Suppose that θ has two variables $\{\theta_1, \theta_2\}$

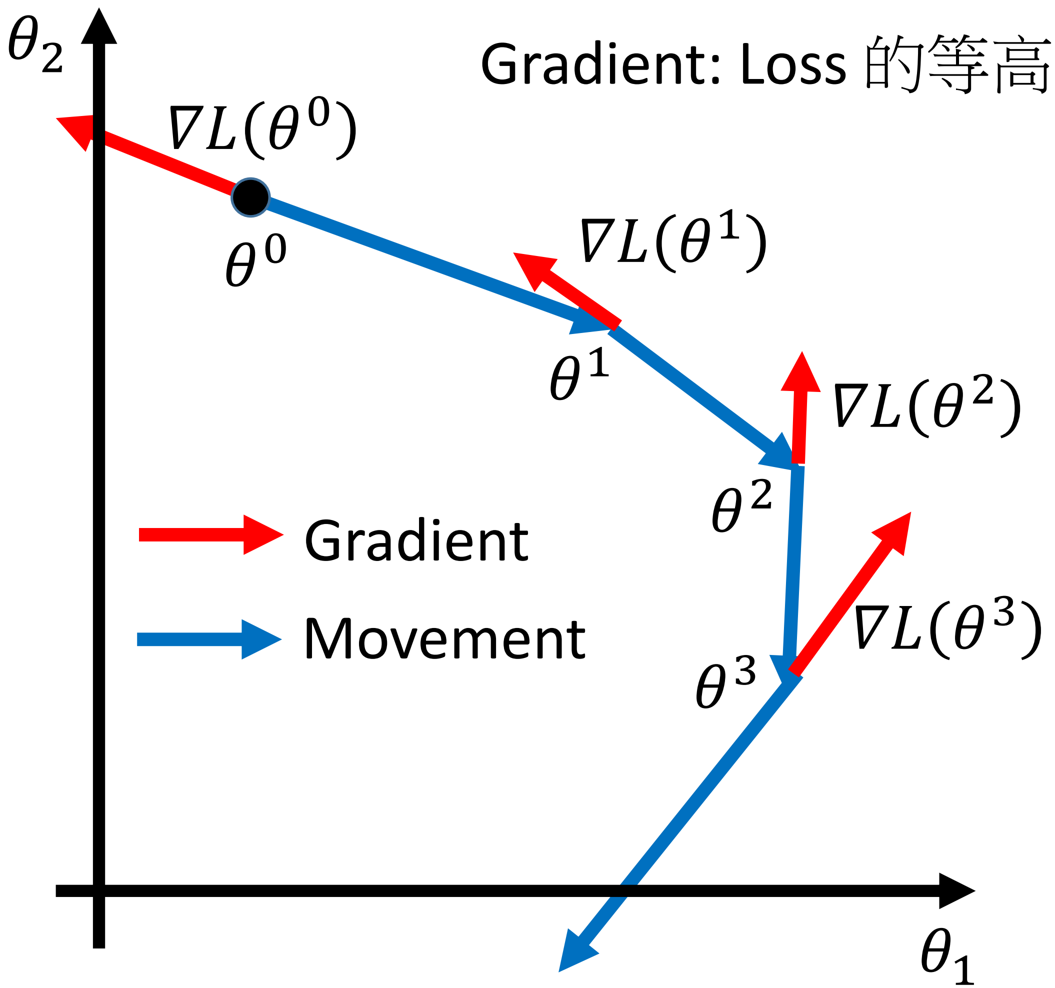
Randomly start at $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$

$$\nabla L(\theta) = \begin{bmatrix} \partial L(\theta_1)/\partial \theta_1 \\ \partial L(\theta_2)/\partial \theta_2 \end{bmatrix}$$

$$\begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix} - \eta \begin{bmatrix} \partial L(\theta_1^0)/\partial \theta_1 \\ \partial L(\theta_2^0)/\partial \theta_2 \end{bmatrix} \Rightarrow \theta^1 = \theta^0 - \eta \nabla L(\theta^0)$$

$$\begin{bmatrix} \theta_1^2 \\ \theta_2^2 \end{bmatrix} = \begin{bmatrix} \theta_1^1 \\ \theta_2^1 \end{bmatrix} - \eta \begin{bmatrix} \partial L(\theta_1^1)/\partial \theta_1 \\ \partial L(\theta_2^1)/\partial \theta_2 \end{bmatrix} \Rightarrow \theta^2 = \theta^1 - \eta \nabla L(\theta^1)$$

Review: Gradient Descent



Gradient: Loss 的等高線的法線方向

Start at position θ^0

Compute gradient at θ^0

Move to $\theta^1 = \theta^0 - \eta\nabla L(\theta^0)$

Compute gradient at θ^1

Move to $\theta^2 = \theta^1 - \eta\nabla L(\theta^1)$

⋮

Gradient Descent

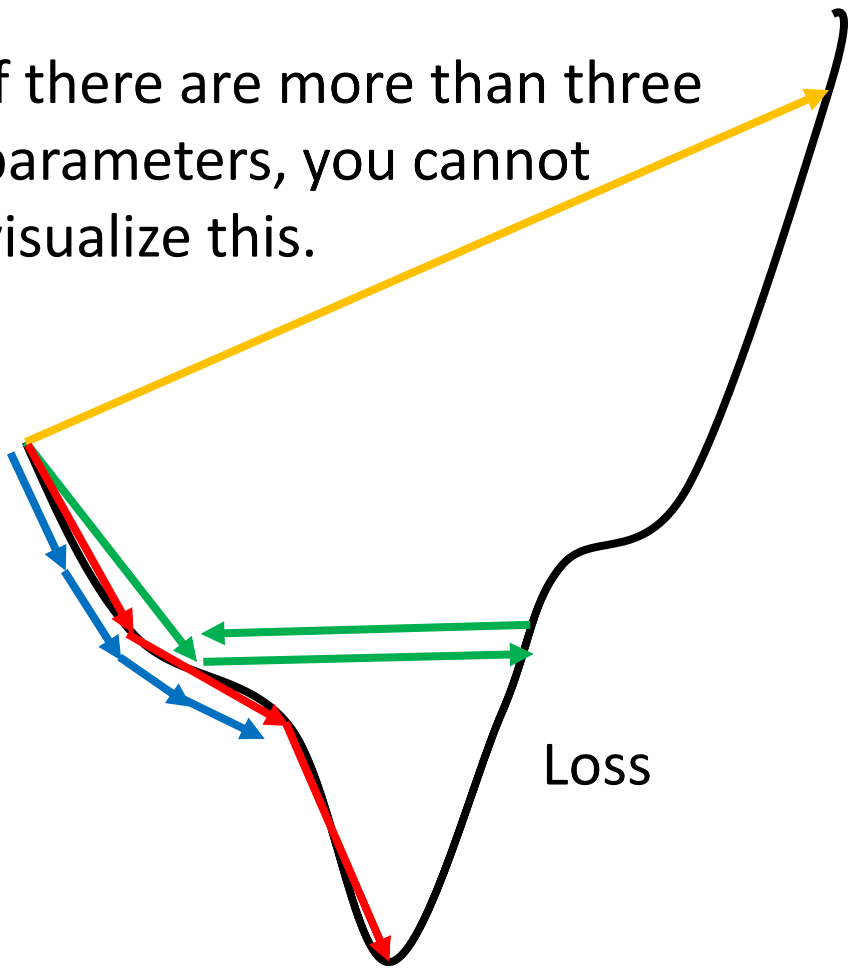
Tip 1: Tuning your learning rates

Learning Rate

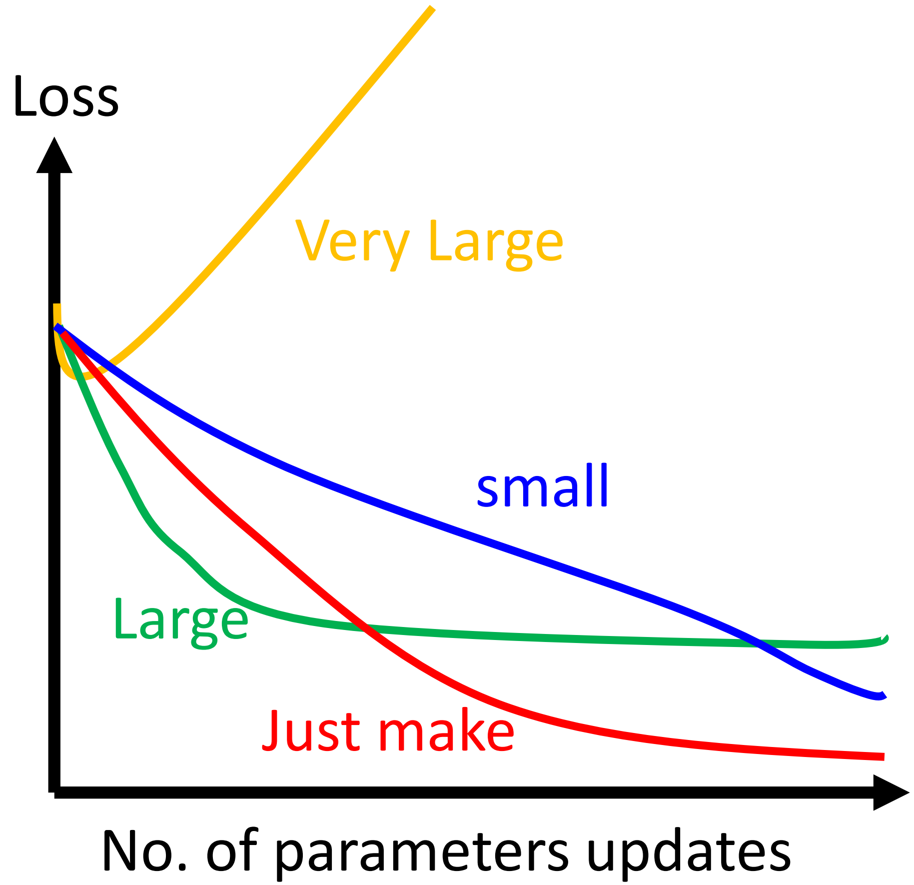
$$\theta^i = \theta^{i-1} - \eta \nabla L\left(\theta^{i-1}\right)$$

Set the learning rate η carefully

If there are more than three parameters, you cannot visualize this.



可以將Loss Function做視覺化



But you can always visualize this.

Adaptive Learning Rates

起始learning rate要大一點，之後要越來越小比較好

- Popular & Simple Idea: Reduce the learning rate by some factor every few epochs.
  - At the beginning, we are far from the destination, so we use larger learning rate
  - After several epochs, we are close to the destination, so we reduce the learning rate
  - E.g. 1/t decay: $\eta^t = \eta / \sqrt{t+1}$ t dependant
- Learning rate cannot be one-size-fits-all
  - Giving different parameters different learning rates

針對不同參數給訂不同learning rate

Adagrad

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \qquad g^t = \frac{\partial L(\theta^t)}{\partial w}$$

- Divide the learning rate of each parameter by the ***root mean square of its previous derivatives***

<u>Vanilla Gradient descent</u>

$$w^{t+1} \leftarrow w^t - \eta^t g^t$$

w is one parameters

<u>Adagrad</u>

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t$$

σ^t: ***root mean square*** of the previous derivatives of parameter w

Parameter dependent

Adagrad

σ^t: ***root mean square*** of the previous derivatives of parameter w

$$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0 \qquad \sigma^0 = \sqrt{(g^0)^2}$$

$$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1 \qquad \sigma^1 = \sqrt{\frac{1}{2}[(g^0)^2 + (g^1)^2]}$$

$$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2 \qquad \sigma^2 = \sqrt{\frac{1}{3}[(g^0)^2 + (g^1)^2 + (g^2)^2]}$$

$$\vdots$$

$$w^{t+1} \leftarrow w^t - \frac{\eta^t}{\sigma^t} g^t \qquad \sigma^t = \sqrt{\frac{1}{t+1}\sum_{i=0}^{t}(g^i)^2}$$

Adagrad

Adam方法是目前比較穩定的
adaptive learning rate方法

- Divide the learning rate of each parameter by the ***root mean square of its previous derivatives***

$$w^{t+1} \leftarrow w^t - \frac{\boxed{\eta^t}}{\boxed{\sigma^t}} g^t$$

$$\eta^t = \frac{\eta}{\cancel{\sqrt{t+1}}}$$

1/t decay

$$\sigma^t = \sqrt{\cancel{\frac{1}{t+1}}\sum_{i=0}^{t}(g^i)^2}$$

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

Contradiction? $\eta^t = \frac{\eta}{\sqrt{t+1}}$ $g^t = \frac{\partial L(\theta^t)}{\partial w}$

Vanilla Gradient descent

$$w^{t+1} \leftarrow w^t - \eta^t \underline{g^t}$$

Larger gradient, larger step

Adagrad

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\underline{\sum_{i=0}^{t} (g^i)^2}}} \underline{g^t}$$

參數調整步伐越大

Larger gradient, larger step

Larger gradient, smaller step

參數調整步伐越小

Intuitive Reason

$$\eta^t = \frac{\eta}{\sqrt{t+1}} \quad g^t = \frac{\partial C(\theta^t)}{\partial w}$$

- How surprise it is 反差

特別大

g^0	g^1	g^2	g^3	g^4	……
0.001	0.001	0.003	0.002	0.1	……

g^0	g^1	g^2	g^3	g^4	……
10.8	20.9	31.7	12.1	0.1	……

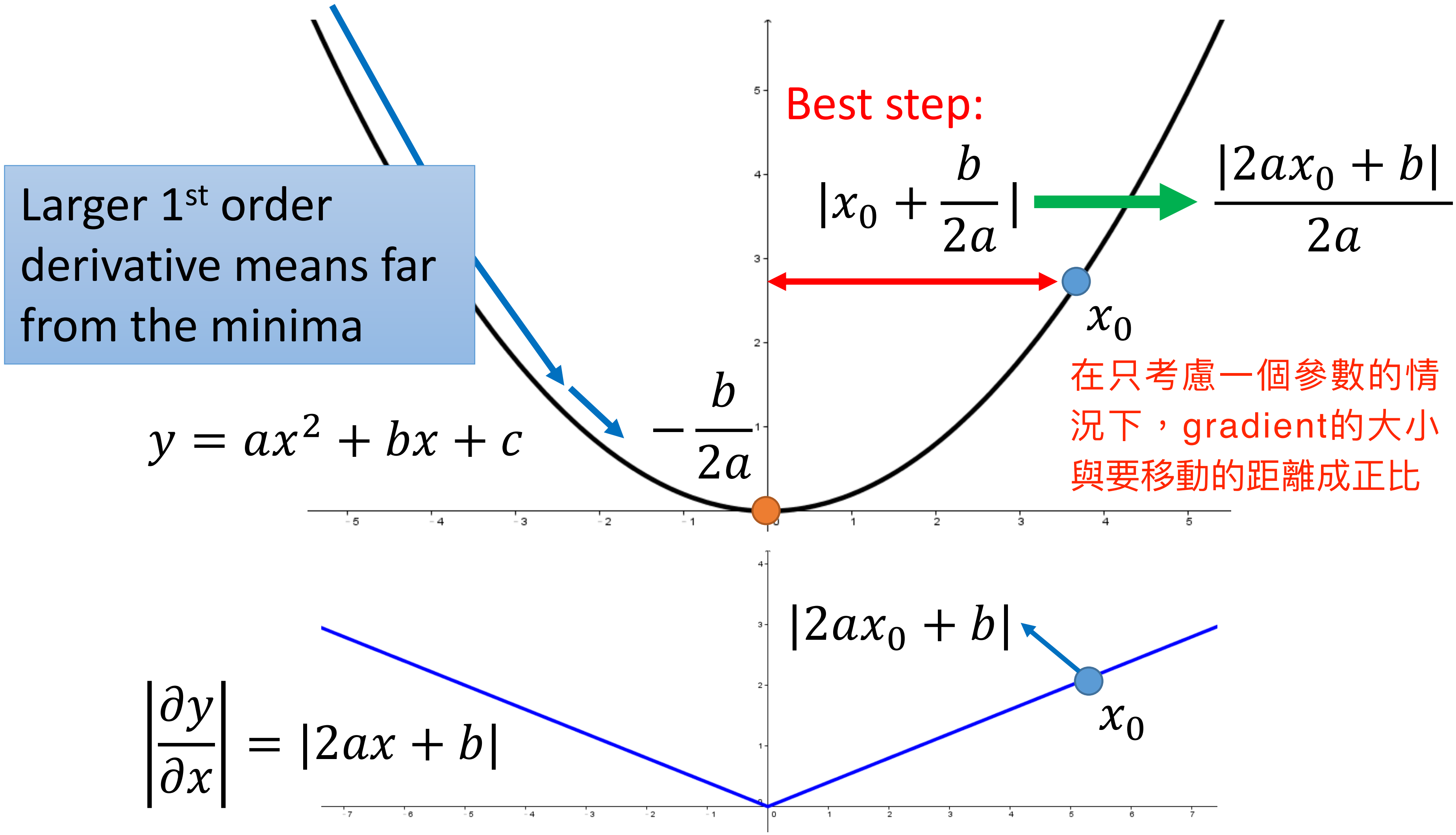
特別小

$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\underline{\sum_{i=0}^{t}(g^i)^2}}} g^t$$

造成反差的效果

強調反差(gradient變化)的感覺

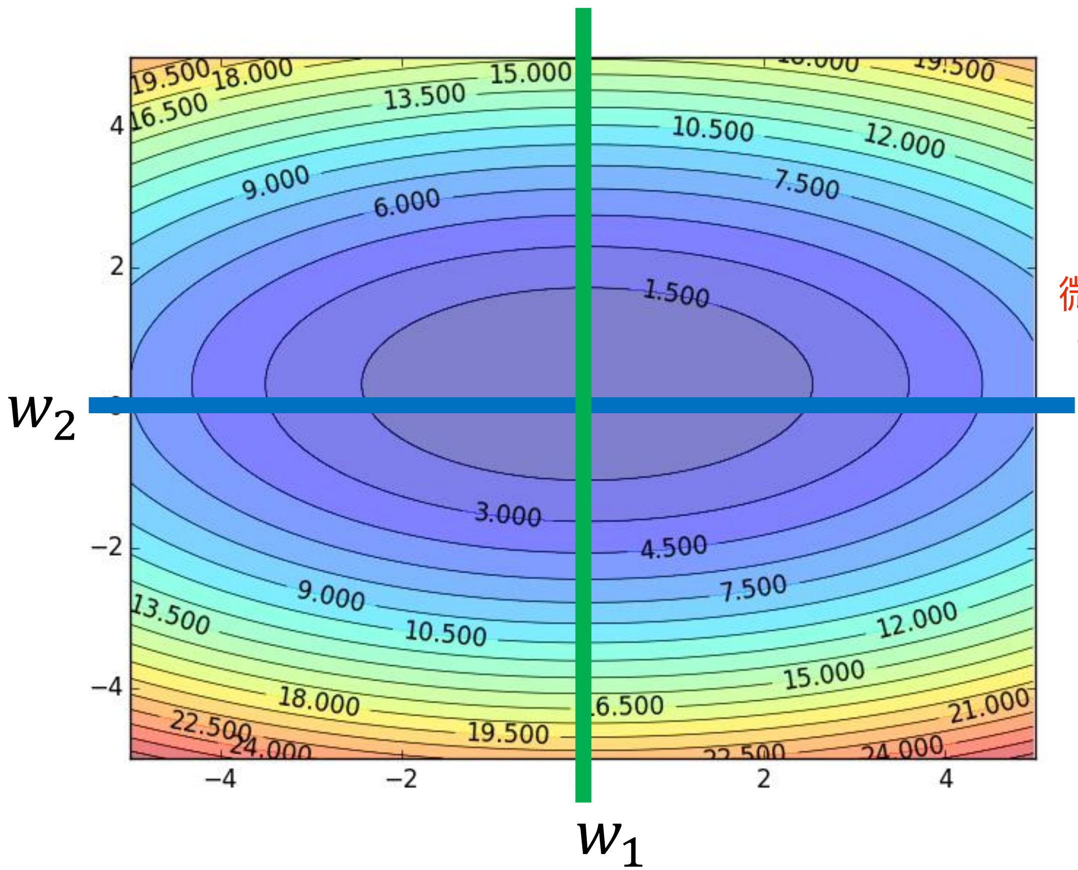
Larger gradient, larger steps?

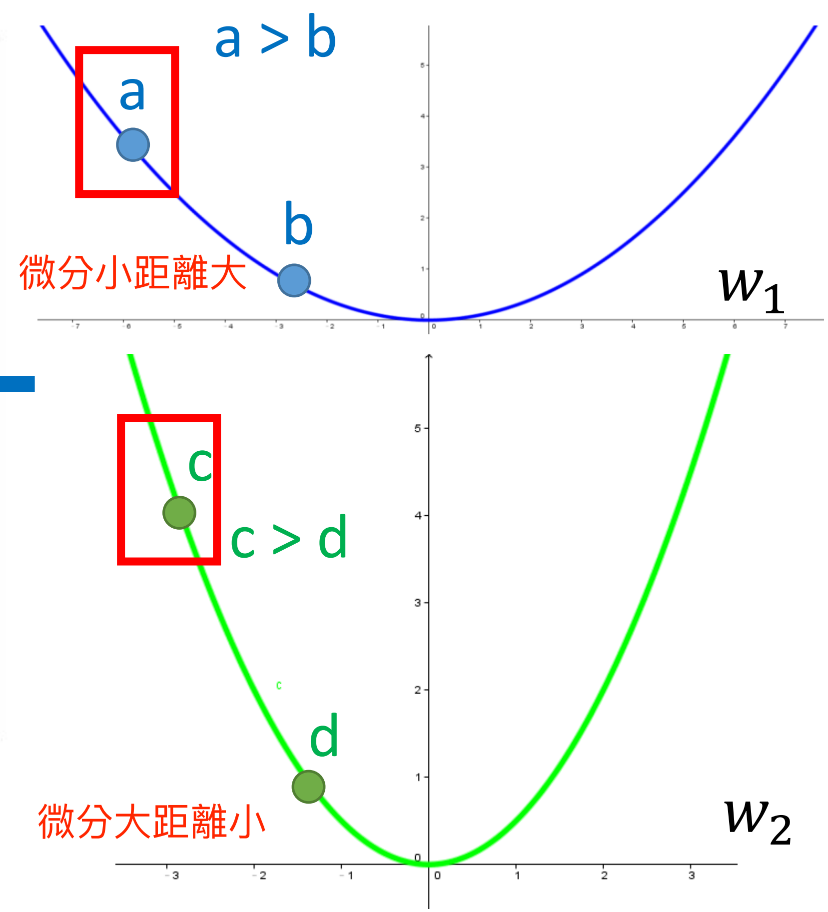


Comparison between different parameters

~~Larger 1st order derivative means far from the minima~~

Do not cross parameters





Second Derivative

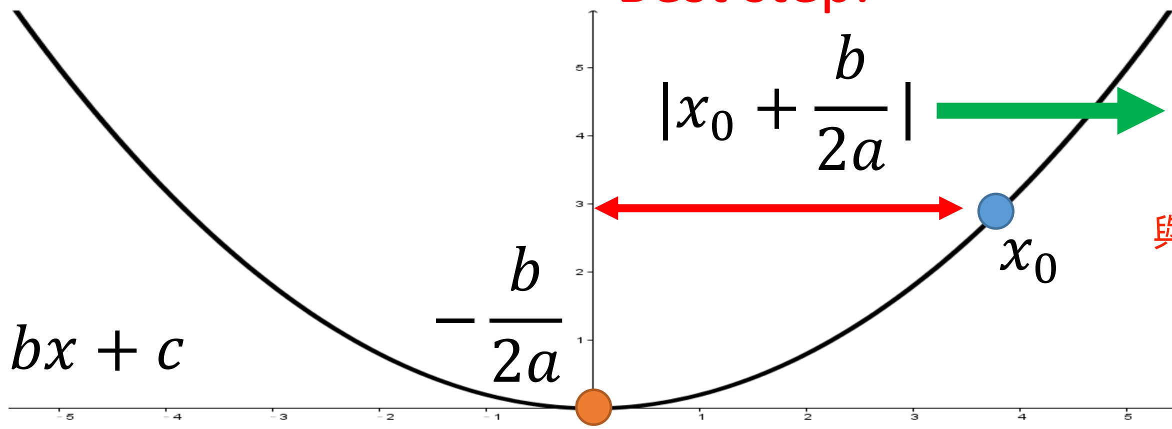
最好的調整方式為：與一次微分成正比，與二次微分成反比

Best step:

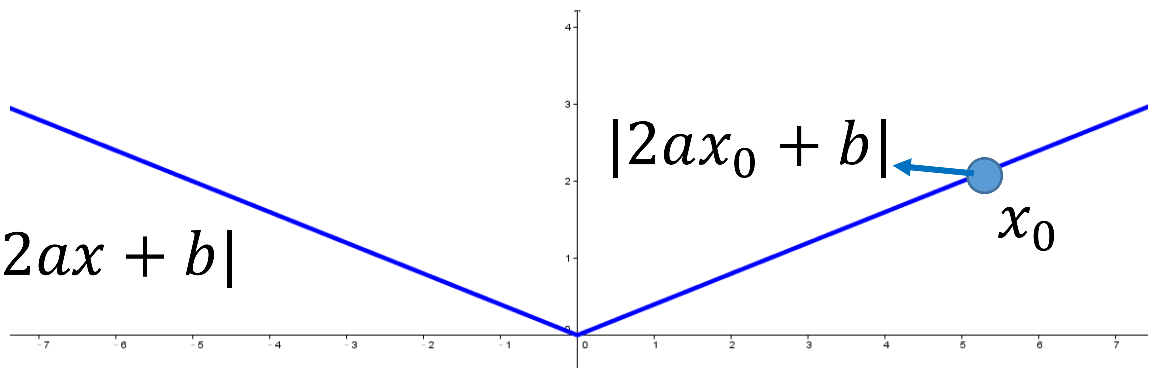
$$\left|x_0 + \frac{b}{2a}\right| \rightarrow \frac{|2ax_0 + b|}{\boxed{2a}}$$

與二次微分成反比

$$y = ax^2 + bx + c$$



$$\left|\frac{\partial y}{\partial x}\right| = |2ax + b|$$



$$\frac{\partial^2 y}{\partial x^2} = 2a$$

二次微分

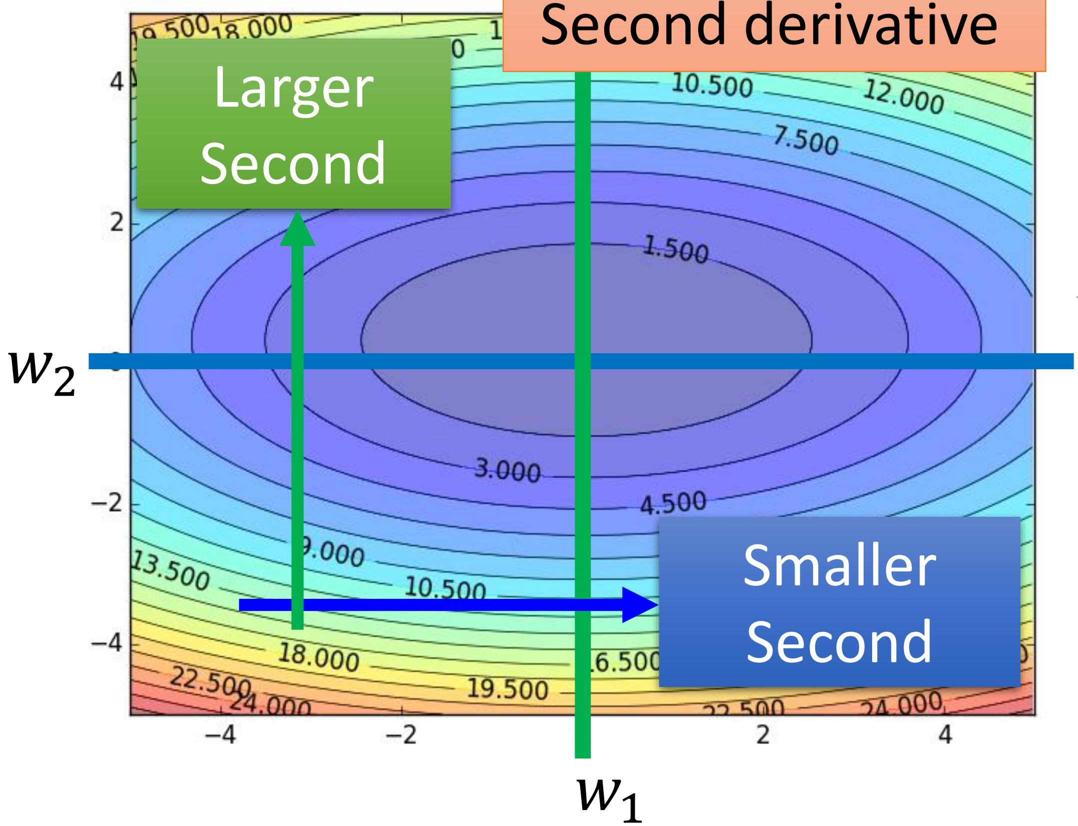
The best step is $\dfrac{|\text{First derivative}|}{\text{Second derivative}}$

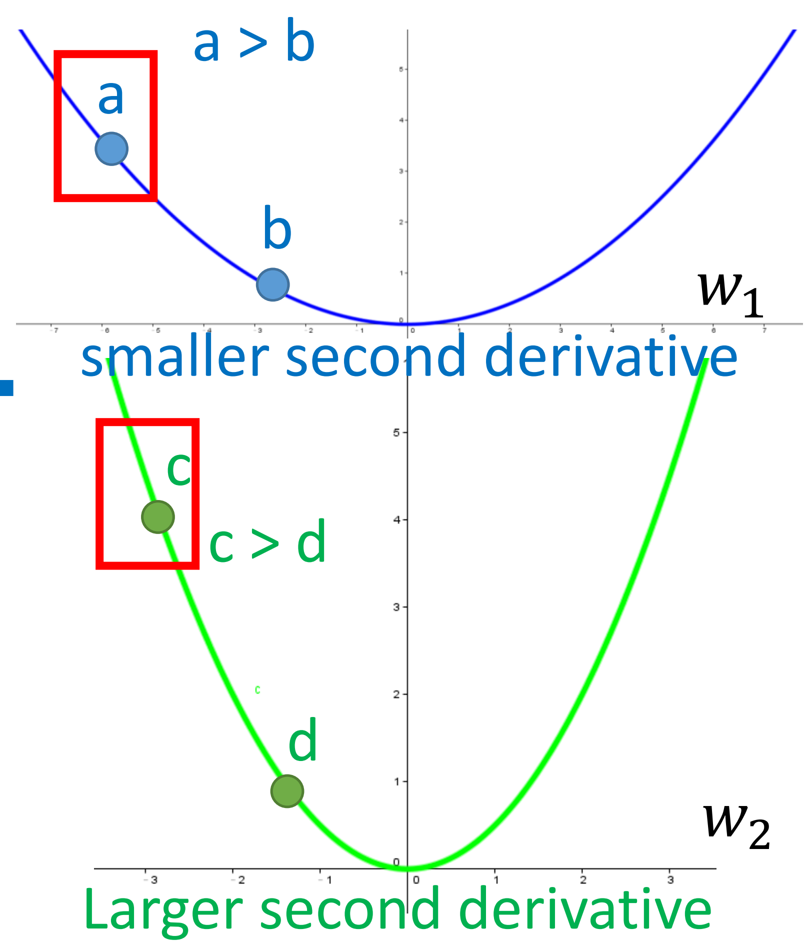
Comparison between different parameters

~~Larger 1st order derivative means far from the minima~~

Do not cross parameters

The best step is $\dfrac{|\text{First derivative}|}{\text{Second derivative}}$





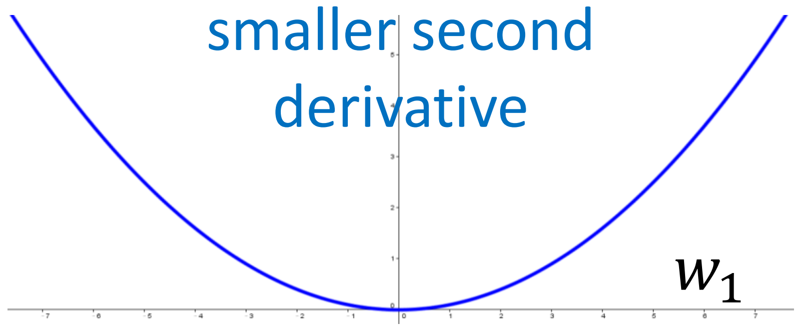
$$w^{t+1} \leftarrow w^t - \frac{\eta}{\sqrt{\sum_{i=0}^{t}(g^i)^2}} g^t$$

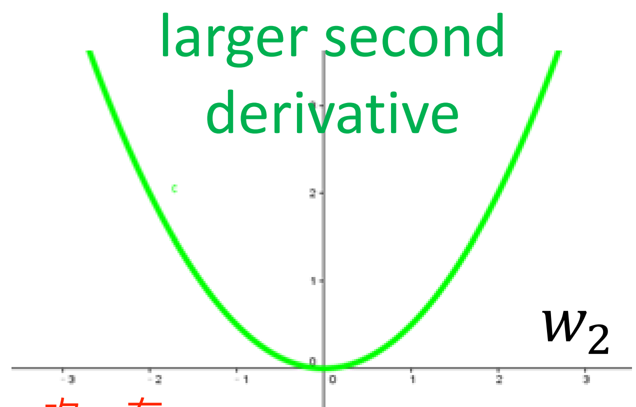
?

The best step is

$$\frac{|\text{First derivative}|}{\text{Second derivative}}$$

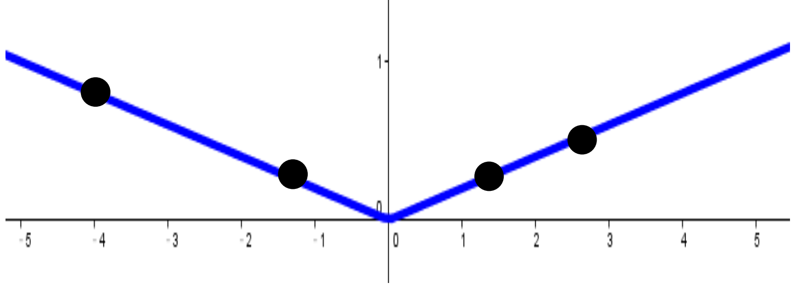
Use *first derivative* to estimate *second derivative*

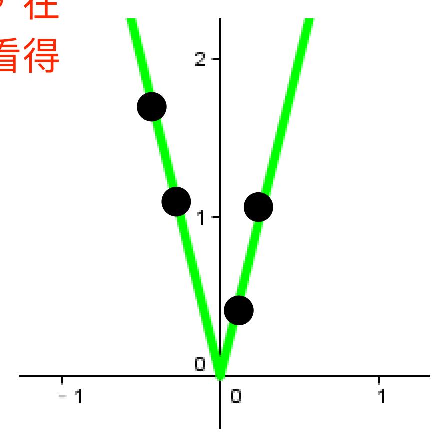




每一個點就好像sample一次，在sample很多個點的時候就可以看得出其二次微分特徵

$$\sqrt{(\text{first derivative})^2}$$





Gradient Descent

Tip 2: Stochastic Gradient Descent

Make the training faster

Stochastic Gradient Descent

$$L=\sum_{n}\left(\hat{y}^{n}-\left(b+\sum w_{i}x_{i}^{n}\right)\right)^{2}$$

Loss is the summation over all training examples

◆ ***Gradient Descent*** $\theta^{i}=\theta^{i-1}-\eta\nabla L\left(\theta^{i-1}\right)$

◆ ***Stochastic Gradient Descent***

Faster!

Pick an example x^n 只考慮一個example

$$L^{n}=\left(\hat{y}^{n}-\left(b+\sum w_{i}x_{i}^{n}\right)\right)^{2}$$

$$\theta^{i}=\theta^{i-1}-\eta\nabla L^{n}\left(\theta^{i-1}\right)$$

Loss for only one example

只針對一個example就調整參數

- Demo

Stochastic Gradient Descent

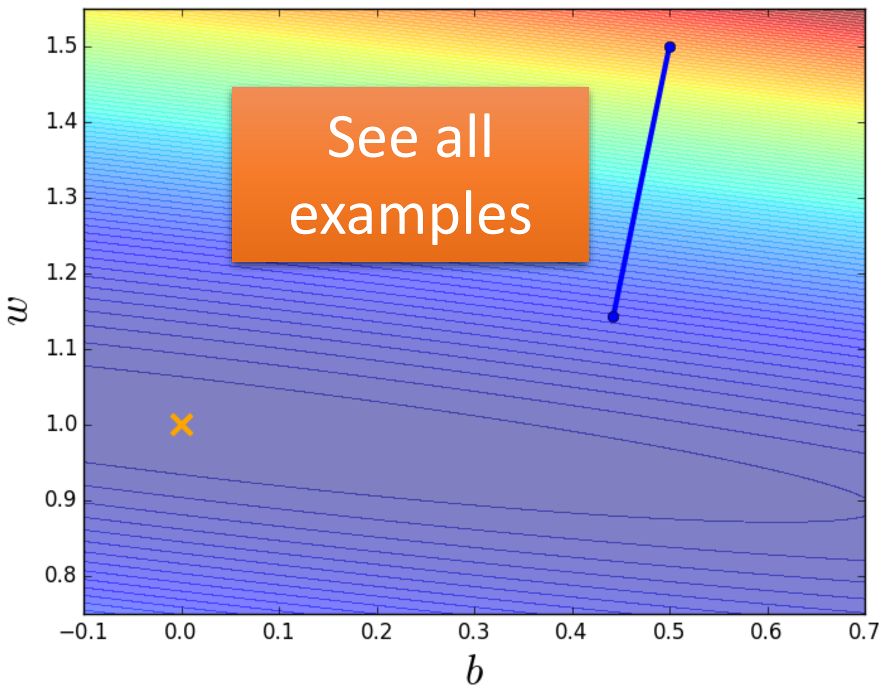
Gradient Descent

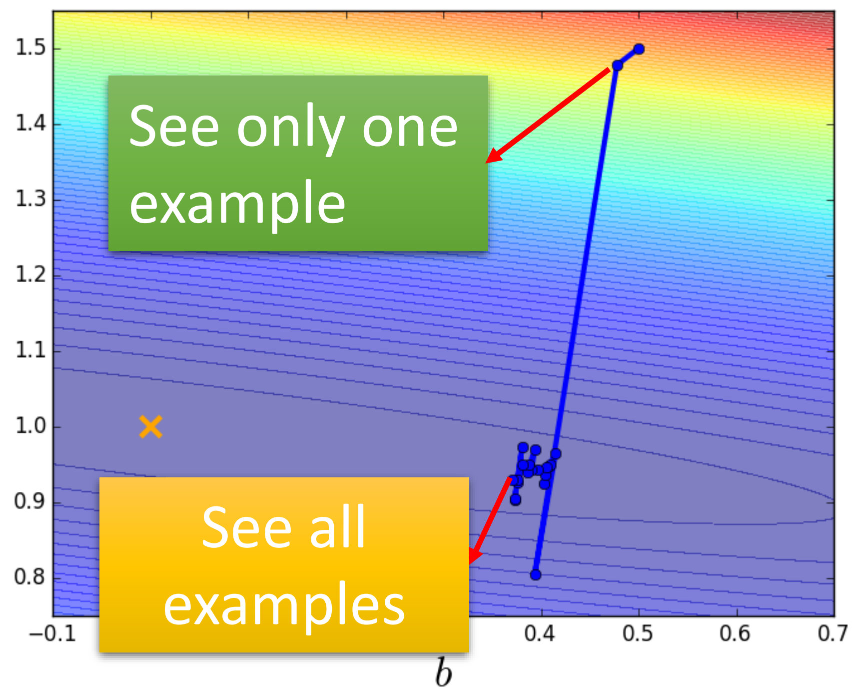
Update after seeing all examples

Stochastic Gradient Descent

Update for each example

If there are 20 examples, 20 times faster.





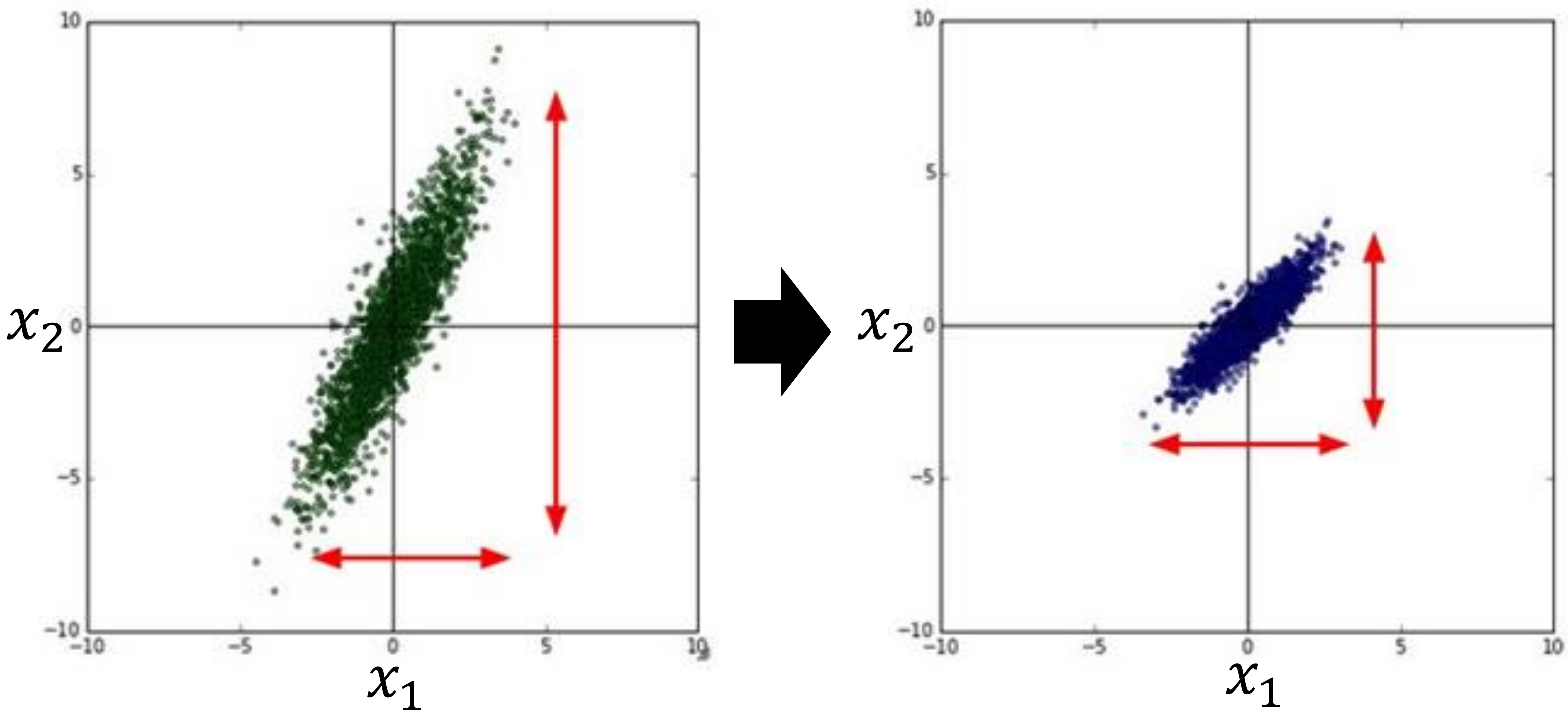
Gradient Descent

Tip 3: Feature Scaling

Feature Scaling

Source of figure: http://cs231n.github.io/neural-networks-2/

$$y = b + w_1 x_1 + w_2 x_2$$

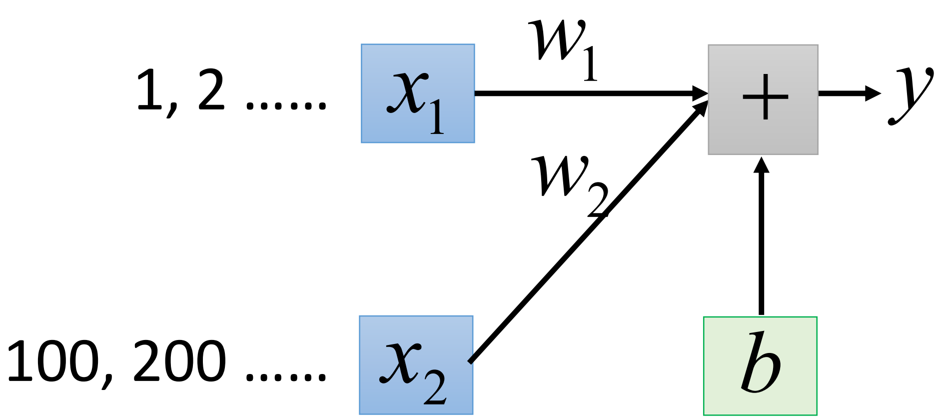


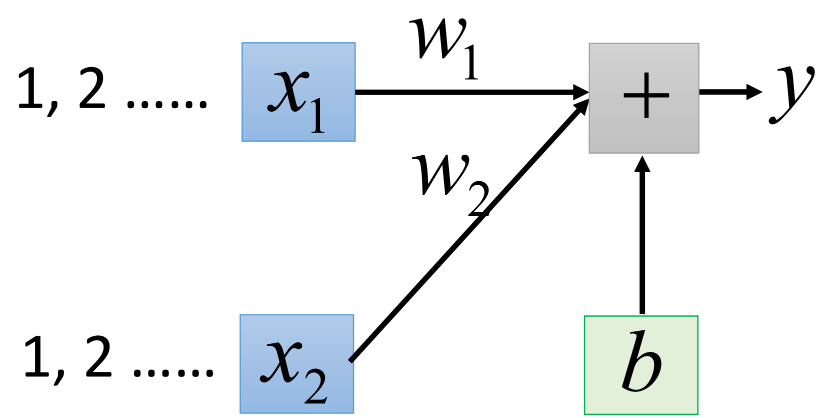
Make different features have the same scaling

將所有的參數分布範圍re-scaling成一樣

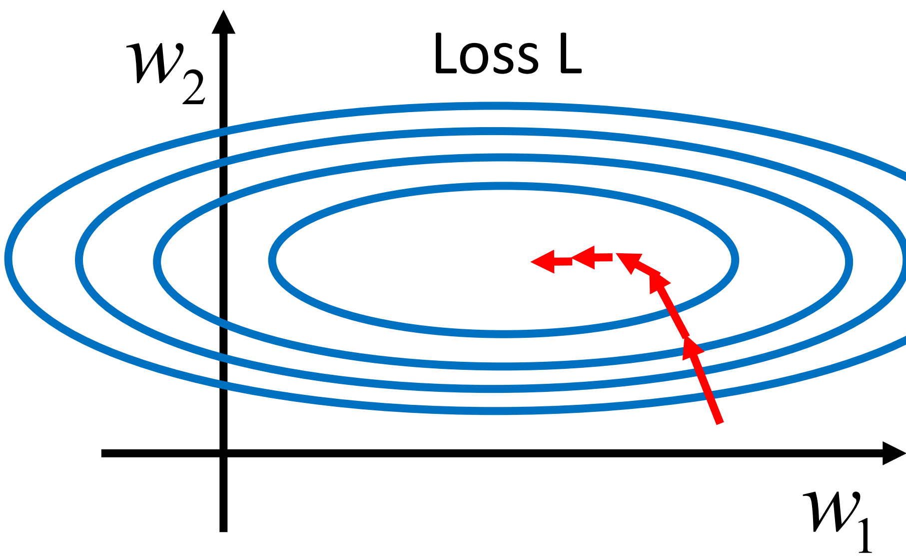
Feature Scaling

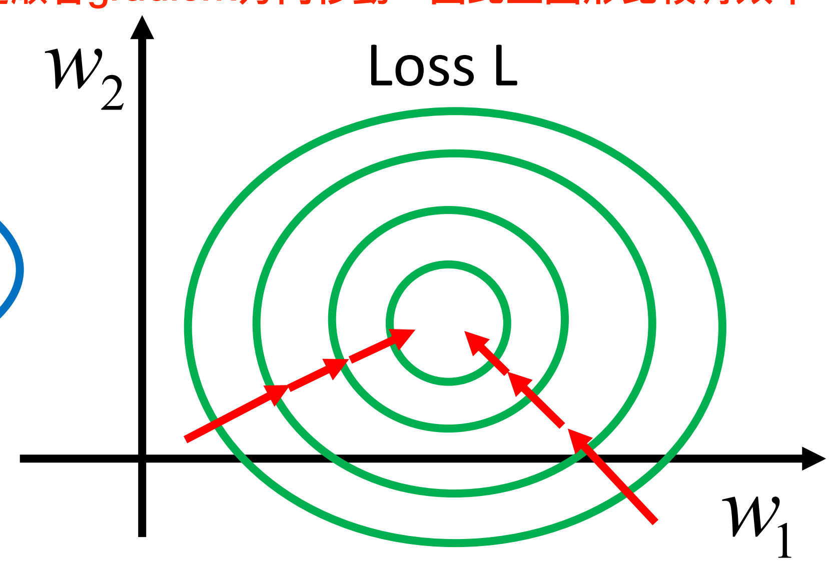
$$y = b + w_1x_1 + w_2x_2$$



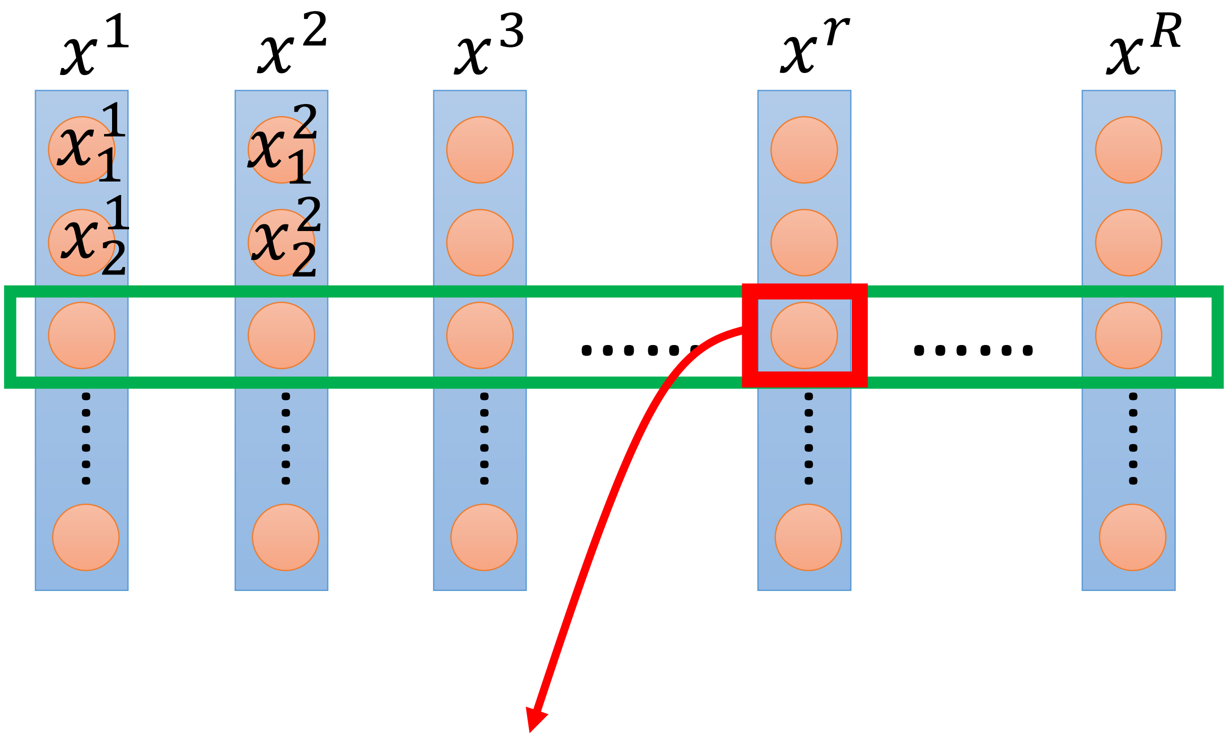


update參數時是順著gradient方向移動，因此正圓形比較有效率





Feature Scaling



For each dimension i:

mean: m_i

standard deviation: σ_i

$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

Normalization

The means of all dimensions are 0, and the variances are all 1

Gradient Descent

Theory

Question

- When solving:

$$\theta^* = \arg\min_{\theta} L(\theta) \quad \text{by gradient descent}$$

- Each time we update the parameters, we obtain θ that makes $L(\theta)$ smaller.

$$L(\theta^0) > L(\theta^1) > L(\theta^2) > \cdots$$

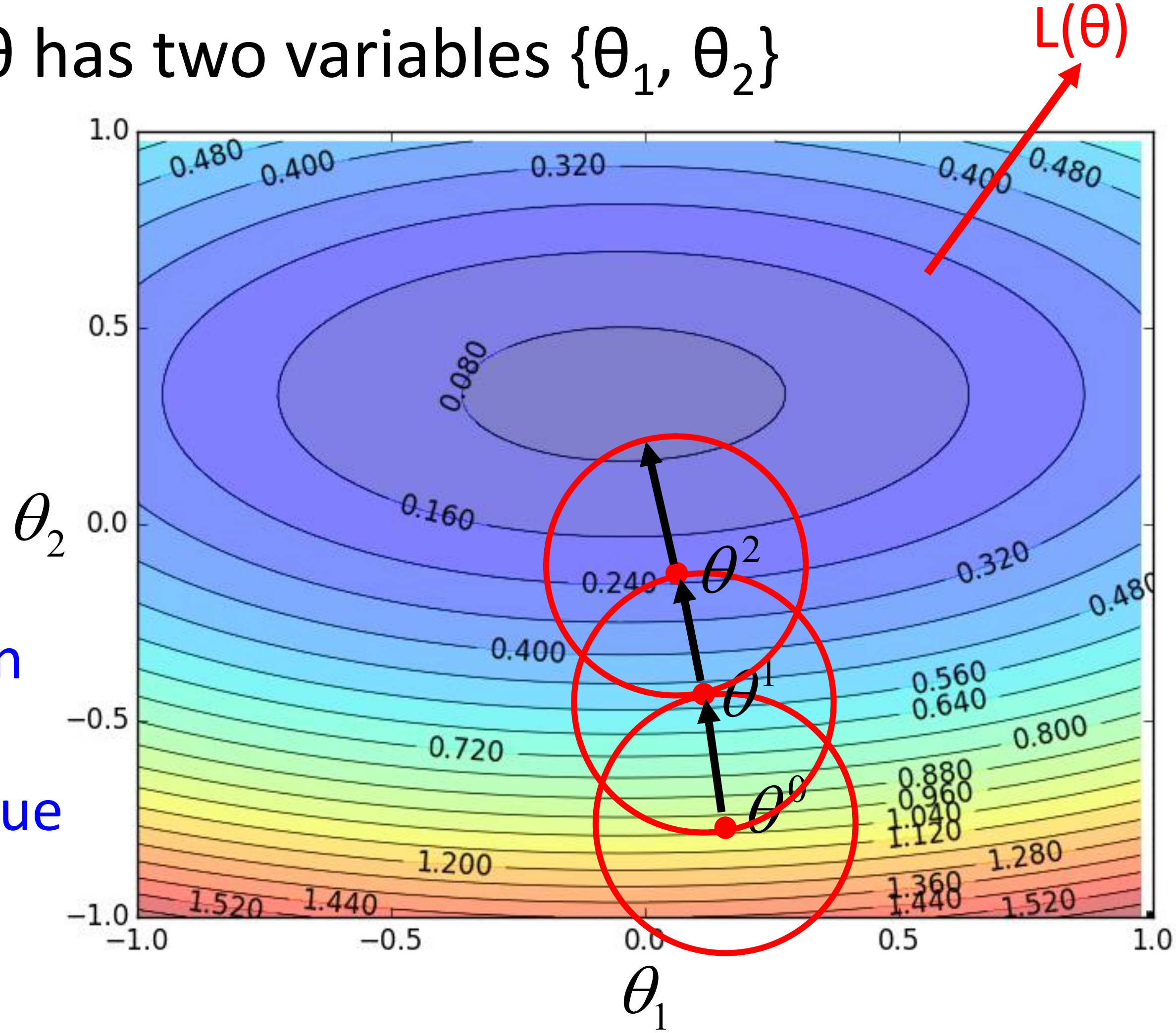
Is this statement correct?

不見得loss會下降

Warning of Math

Formal Derivation

- Suppose that θ has two variables $\{\theta_1, \theta_2\}$



Given a point, we can easily find the point with the smallest value nearby. How?

Taylor Series

- **Taylor series**: Let h(x) be any function infinitely differentiable around x = x_0.

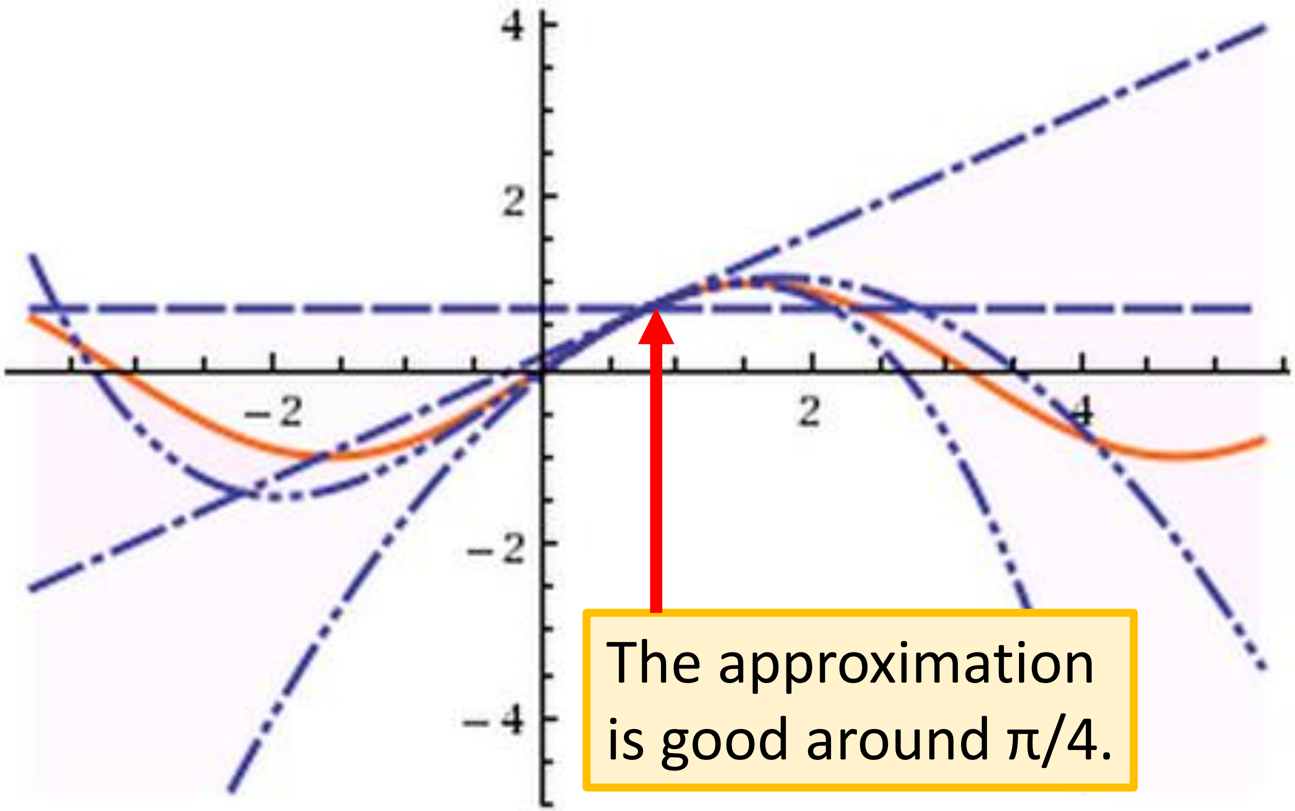
$$\mathrm{h}(\mathrm{x}) = \sum_{k=0}^{\infty} \frac{\mathrm{h}^{(k)}(x_0)}{k!}(x - x_0)^k$$

$$= h(x_0) + h'(x_0)(x - x_0) + \frac{h''(x_0)}{2!}(x - x_0)^2 + \ldots$$

When x is close to x_0 $\Rightarrow$ $h(x) \approx h(x_0) + h'(x_0)(x - x_0)$

E.g. Taylor series for h(x)=sin(x) around $x_0=\pi/4$

$$\sin(x)= \frac{1}{\sqrt{2}} + \frac{x-\frac{\pi}{4}}{\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^2}{2\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^3}{6\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^4}{24\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^5}{120\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^6}{720\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^7}{5040\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^8}{40\,320\sqrt{2}} + \frac{\left(x-\frac{\pi}{4}\right)^9}{362\,880\sqrt{2}} - \frac{\left(x-\frac{\pi}{4}\right)^{10}}{3\,628\,800\sqrt{2}} + \ldots\ldots$$



Multivariable Taylor Series

$$h(x,y)=h(x_0,y_0)+\frac{\partial h(x_0,y_0)}{\partial x}(x-x_0)+\frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)$$

+ something related to $(x-x_0)^2$ and $(y-y_0)^2$ + ……

When x and y is close to x_0 and y_0

$$h(x,y)\approx \underbrace{h(x_0,y_0)}_{\text{零次微分}}+\underbrace{\frac{\partial h(x_0,y_0)}{\partial x}(x-x_0)+\frac{\partial h(x_0,y_0)}{\partial y}(y-y_0)}_{\text{一次微分}}$$

Back to Formal Derivation

Based on Taylor Series:

If the red circle is ***small enough***, in the red circle

$$L(\theta) \approx L(a,b) + \frac{\partial L(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial L(a,b)}{\partial \theta_2}(\theta_2 - b)$$

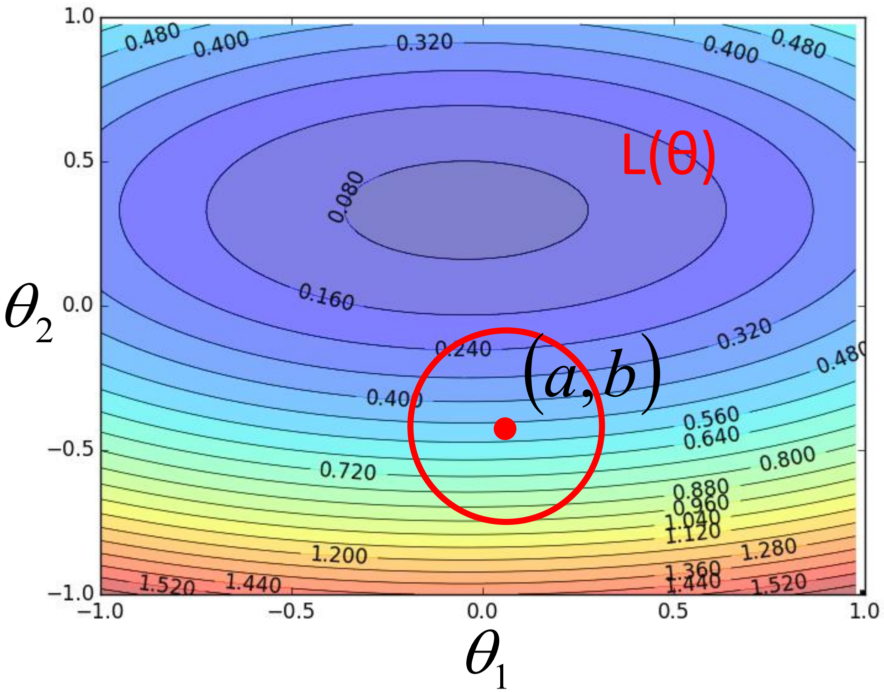
$$s = L(a,b)$$

$$u = \frac{\partial L(a,b)}{\partial \theta_1}, v = \frac{\partial L(a,b)}{\partial \theta_2}$$

$$L(\theta)$$

$$\approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

表示成一個圓



Back to Formal Derivation

Based on Taylor Series:

If the red circle is ***small enough***, in the red circle

$$L(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

constant

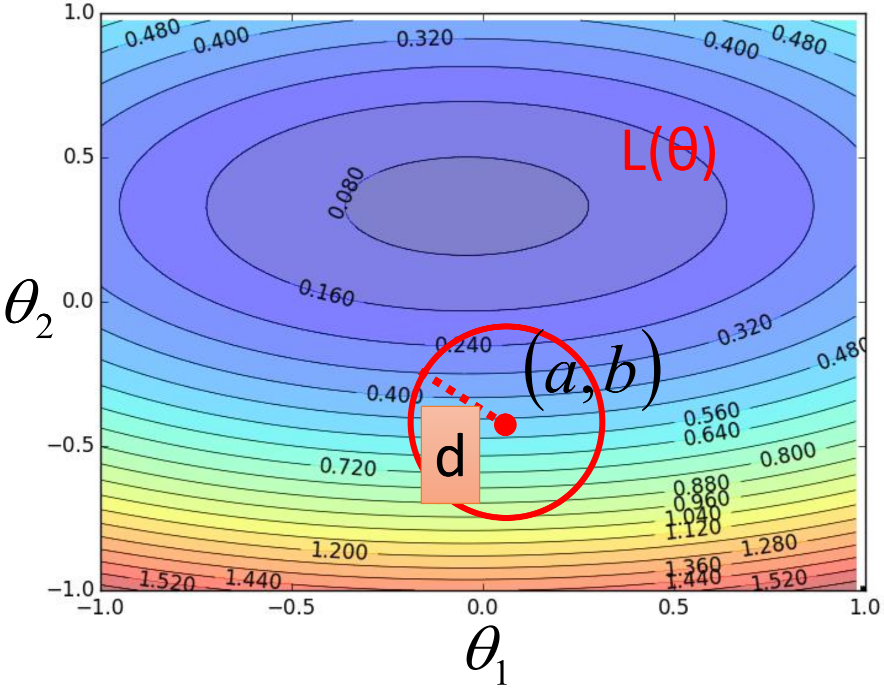
$$s = L(a, b)$$

$$u = \frac{\partial L(a, b)}{\partial \theta_1}, v = \frac{\partial L(a, b)}{\partial \theta_2}$$

Find θ_1 and θ_2 in the red circle ***minimizing*** $L(\theta)$

$$(\theta_1 - a)^2 + (\theta_2 - b)^2 \le d^2$$

Simple, right?



Gradient descent – two variables

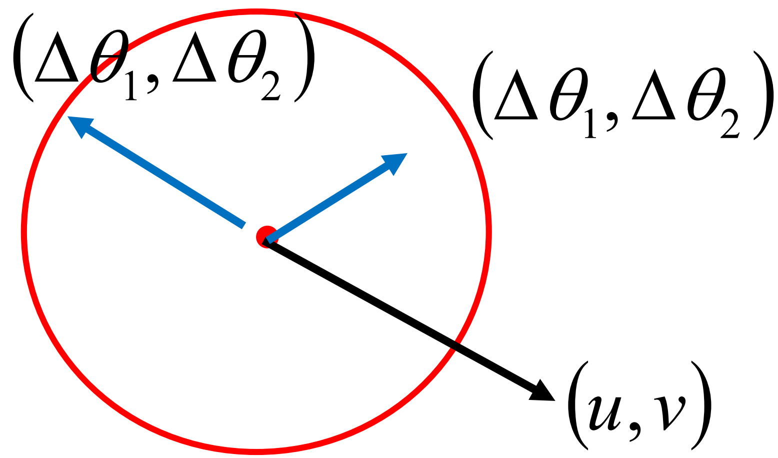
Red Circle: (If the radius is small)

因為是找min值因此s這個常數項不考慮

$$\mathrm{L}(\theta) \approx \cancel{s} + u\underbrace{(\theta_1 - a)}_{\Delta\theta_1} + v\underbrace{(\theta_2 - b)}_{\Delta\theta_2}$$

Find θ_1 and θ_2 in the red circle ***minimizing*** L(θ)

$$\underbrace{(\theta_1 - a)}_{\Delta\theta_1}{}^2 + \underbrace{(\theta_2 - b)}_{\Delta\theta_2}{}^2 \le d^2$$



To minimize L(θ)

$$\begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} = -\eta \begin{bmatrix} u \\ v \end{bmatrix} \Rightarrow \begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \begin{bmatrix} u \\ v \end{bmatrix}$$

雖然使用taylor series二次向可以更精確，但是運算量太大導致deep learning時無法tradeoff

Back to Formal Derivation

Based on Taylor Series:

If the red circle is ***small enough***, in the red circle

constant

$$s = \mathrm{L}(a, b)$$

$$u = \frac{\partial \mathrm{L}(a, b)}{\partial \theta_1}, v = \frac{\partial \mathrm{L}(a, b)}{\partial \theta_2}$$

$$\mathrm{L}(\theta) \approx s + u(\theta_1 - a) + v(\theta_2 - b)$$

前提是Taylor Series夠小夠精確才能使用
->圈圈要夠小->learning rate越小越小越好

Find θ_1 and θ_2 yielding the smallest value of $L(\theta)$ in the circle

$$\begin{bmatrix} \theta_1 \\ \theta_2 \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a \\ b \end{bmatrix} - \eta \begin{bmatrix} \frac{\partial \mathrm{L}(a, b)}{\partial \theta_1} \\ \frac{\partial \mathrm{L}(a, b)}{\partial \theta_2} \end{bmatrix}$$

This is gradient descent.

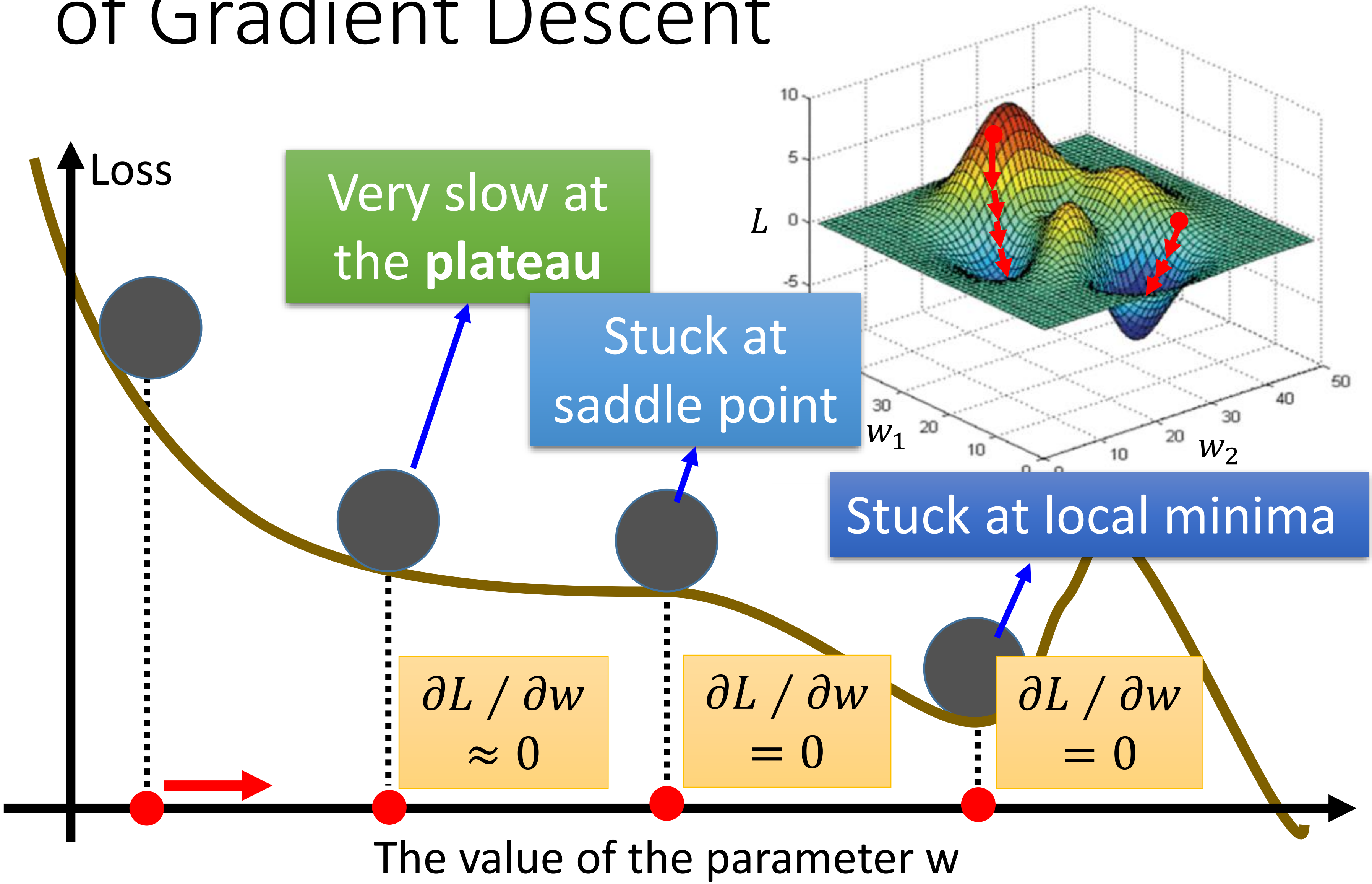
偏微分

Not satisfied if the red circle (learning rate) is not small enough

You can consider the second order term, e.g. Newton's method.

End of Warning

More Limitation of Gradient Descent



Acknowledgement

- 感謝 Victor Chen 發現投影片上的打字錯誤