

Gradient Descent

Review: Gradient Descent

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

$$\theta^* = \arg \min_{\theta} L(\theta) \quad L: \text{loss function} \quad \theta: \text{parameters}$$

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

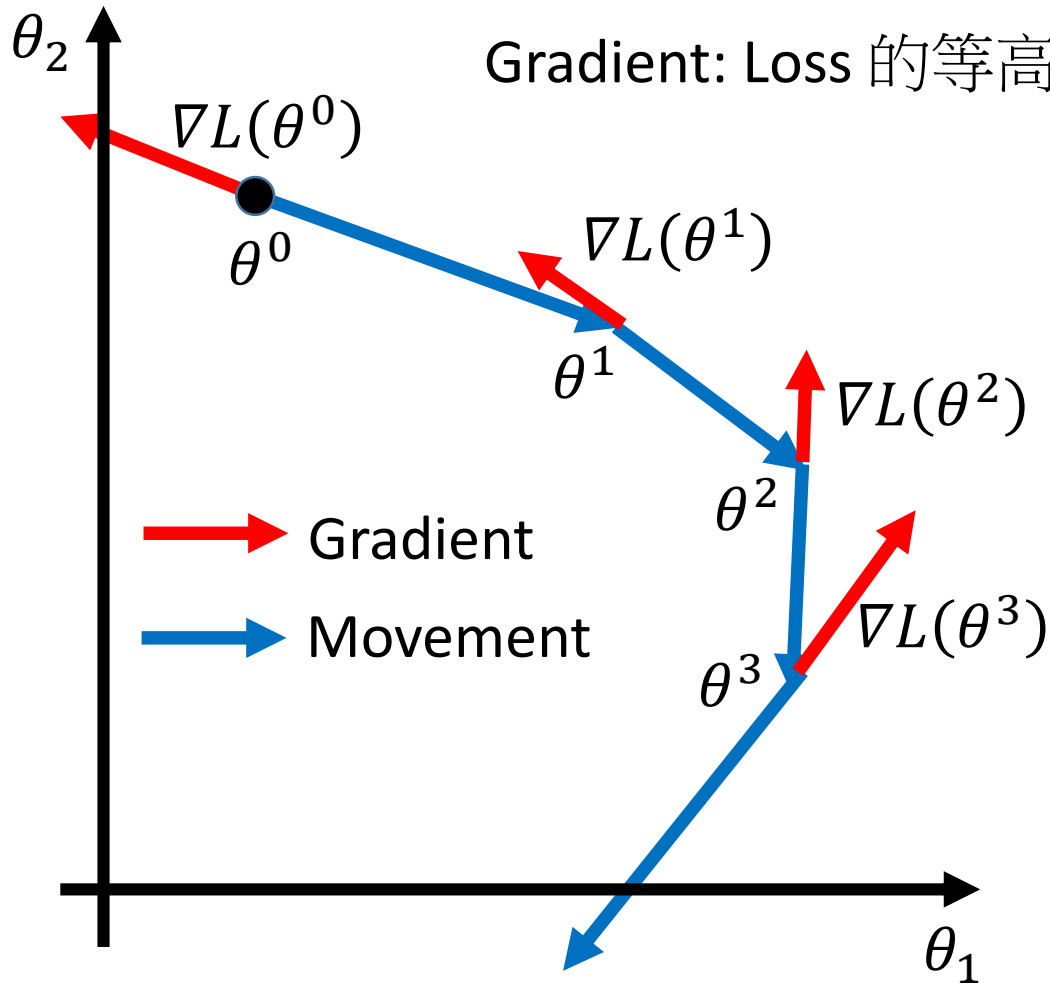
Randomly start at $\theta^0 = \begin{bmatrix} \theta_1^0 \\ \theta_2^0 \end{bmatrix}$

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

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

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

Review: Gradient Descent



Gradient Descent

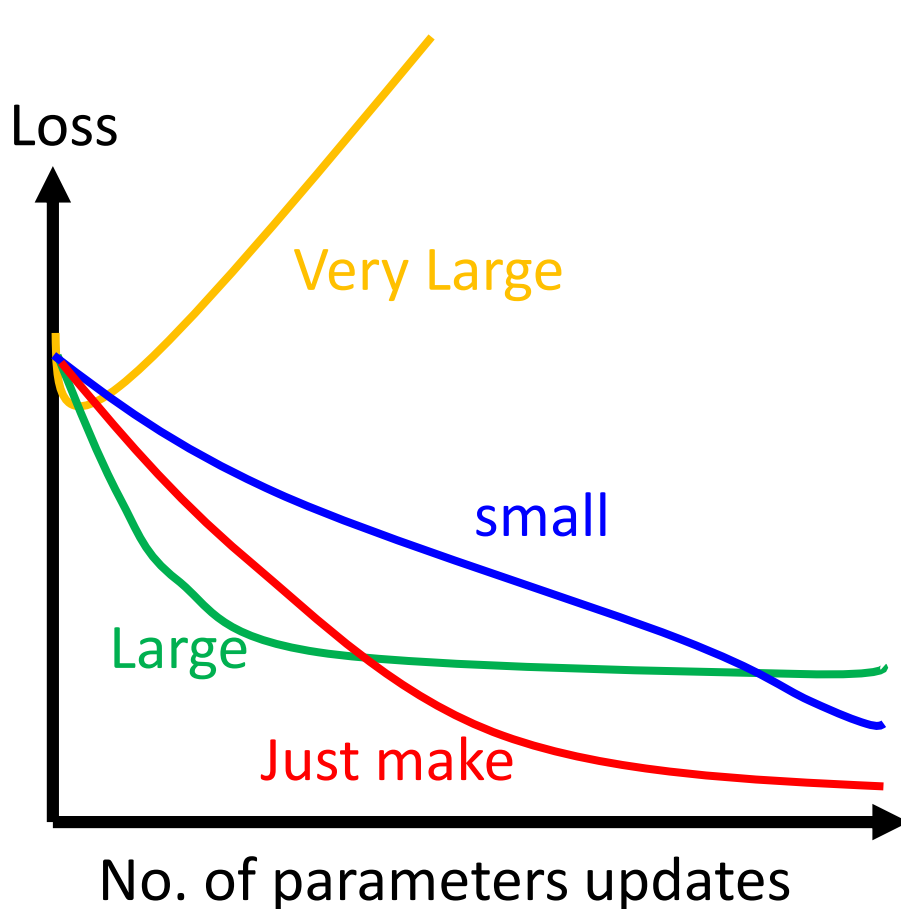
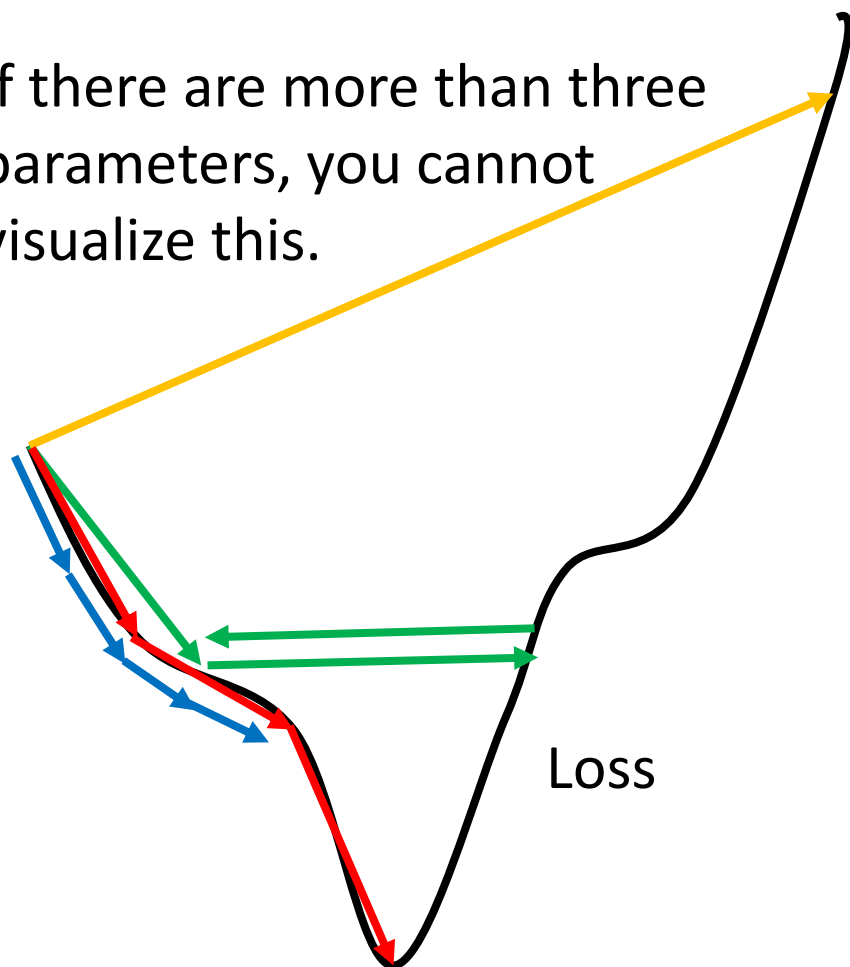
Tip 1: Tuning your
learning rates

Learning Rate

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

Set the learning rate η carefully

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



But you can always visualize this.

Adaptive Learning Rates

- 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: $\eta^t = \eta / \sqrt{t + 1}$
- Learning rate cannot be one-size-fits-all
 - Giving different parameters different learning rates

Adagrad

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

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

Vanilla Gradient descent

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

w is one parameters

Adagrad

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

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

$$w^1 \leftarrow w^0 - \frac{\eta^0}{\sigma^0} g^0$$

$$w^2 \leftarrow w^1 - \frac{\eta^1}{\sigma^1} g^1$$

$$w^3 \leftarrow w^2 - \frac{\eta^2}{\sigma^2} g^2$$

\vdots

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

$$\sigma^0 = \sqrt{(g^0)^2}$$

$$\sigma^1 = \sqrt{\frac{1}{2} [(g^0)^2 + (g^1)^2]}$$

$$\sigma^2 = \sqrt{\frac{1}{3} [(g^0)^2 + (g^1)^2 + (g^2)^2]}$$

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

Adagrad

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

The diagram illustrates the Adagrad update rule. It shows two versions of the weight update equation. The top version uses variable learning rate η^t and RMS σ^t , while the bottom version uses the fixed initial learning rate η and the cumulative RMS. A large blue arrow points from the top equation to the bottom one, indicating the simplification. Red arrows point from the boxes containing η^t and σ^t in the top equation to their respective definitions. Red lines are drawn through the $t+1$ terms in the denominators of the definitions for η^t and σ^t , with the text "1/t decay" in red next to the η^t definition.

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

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

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

→ Larger gradient, larger step

→ Larger gradient, smaller step

Intuitive Reason

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

- How surprise it is 反差

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

特别大

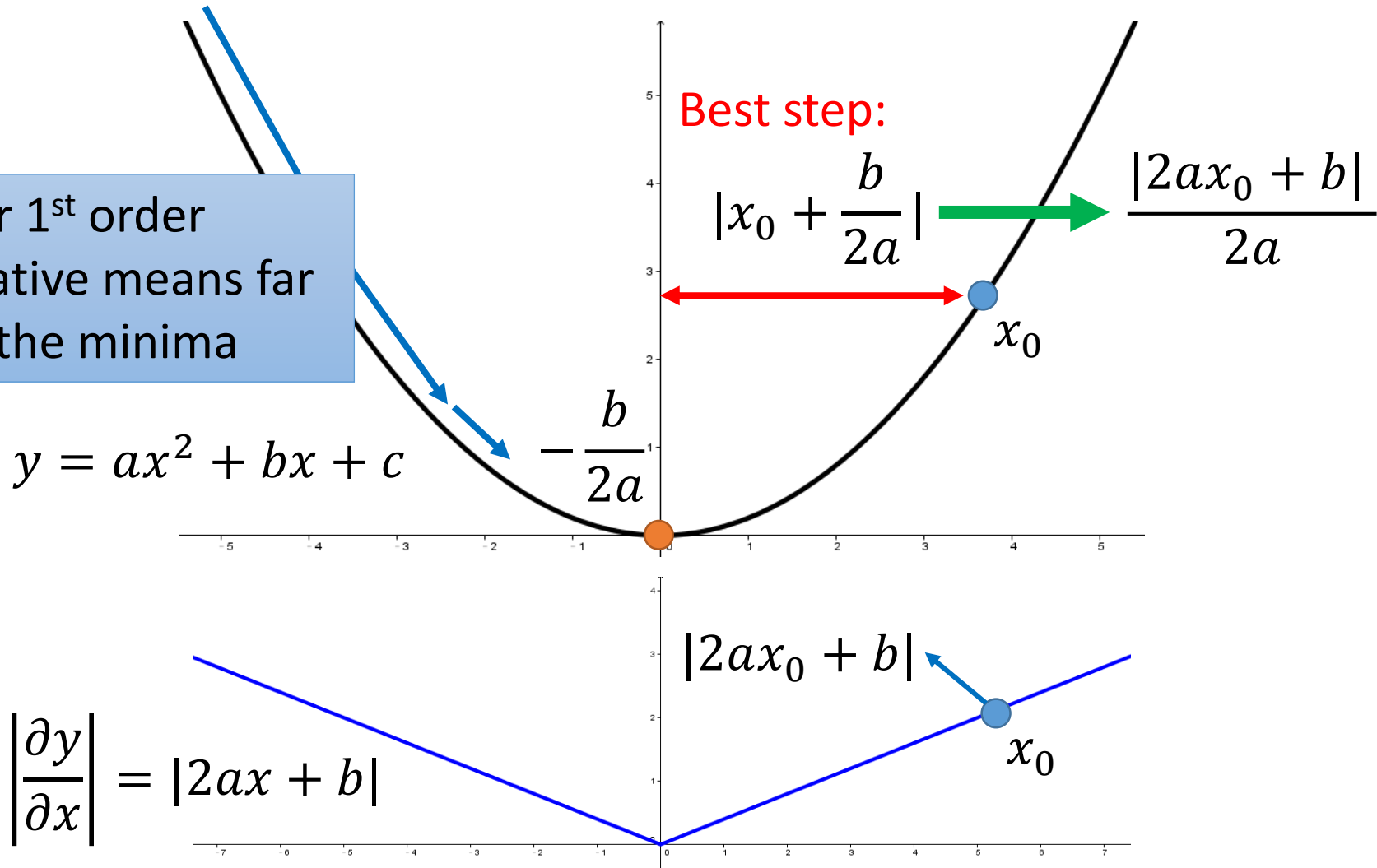
特别小

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

造成反差的效果

Larger gradient, larger steps?

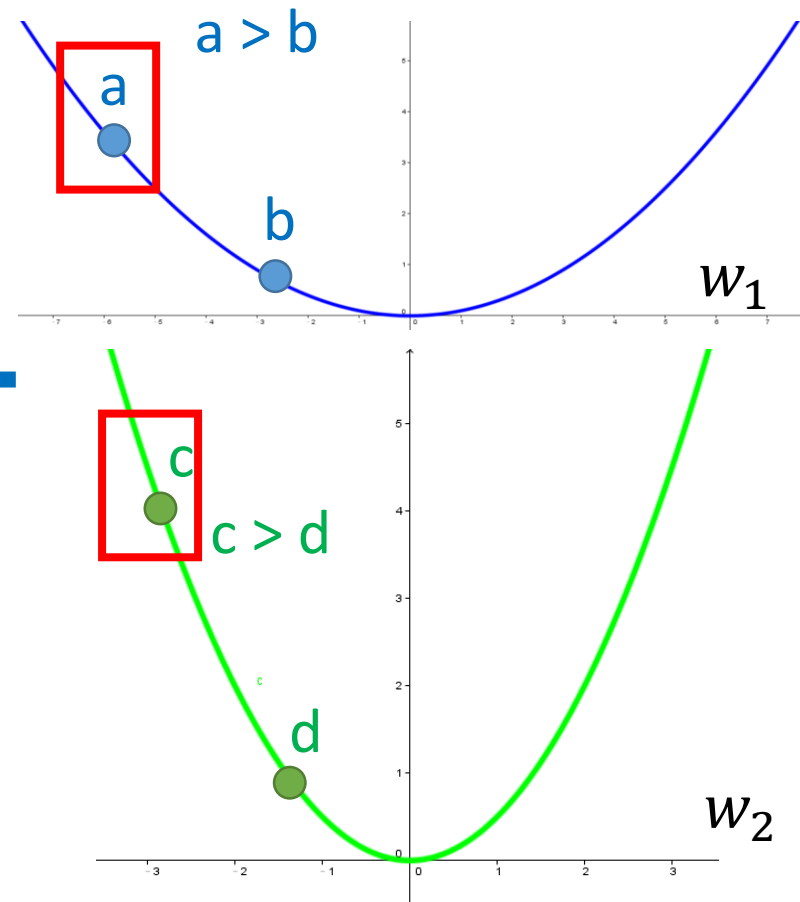
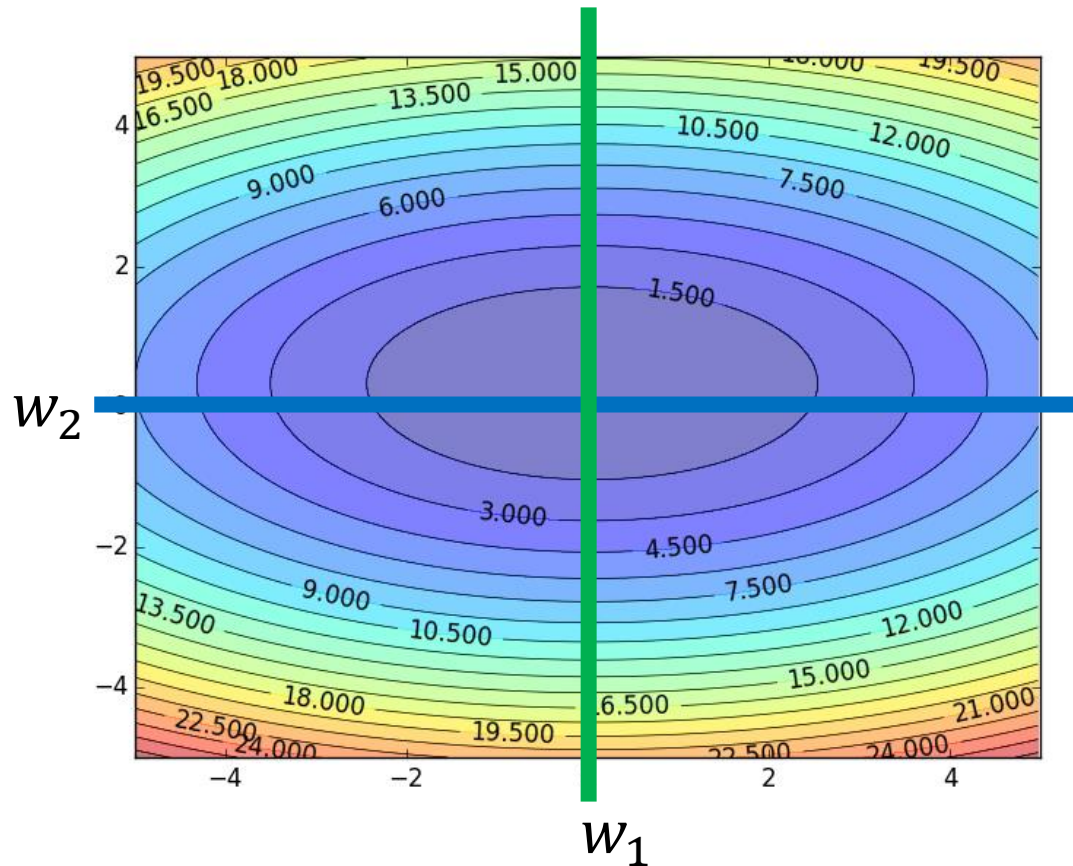
Larger 1st order derivative means far from the minima



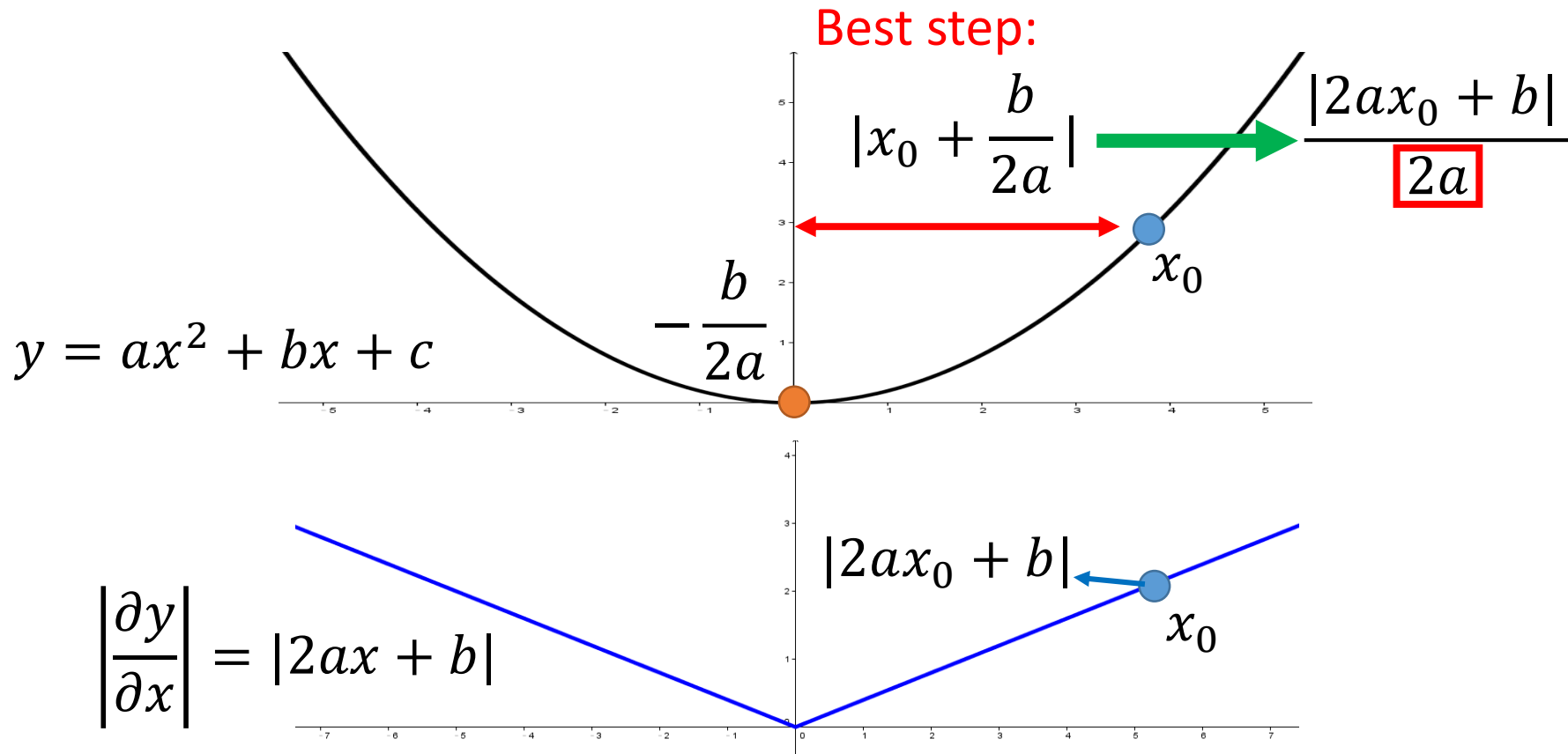
Comparison between different parameters

Larger 1st order derivative means far from the minima

Do not cross parameters



Second Derivative



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

The best step is

|First derivative|
Second derivative

Comparison between different parameters

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

Do not cross parameters

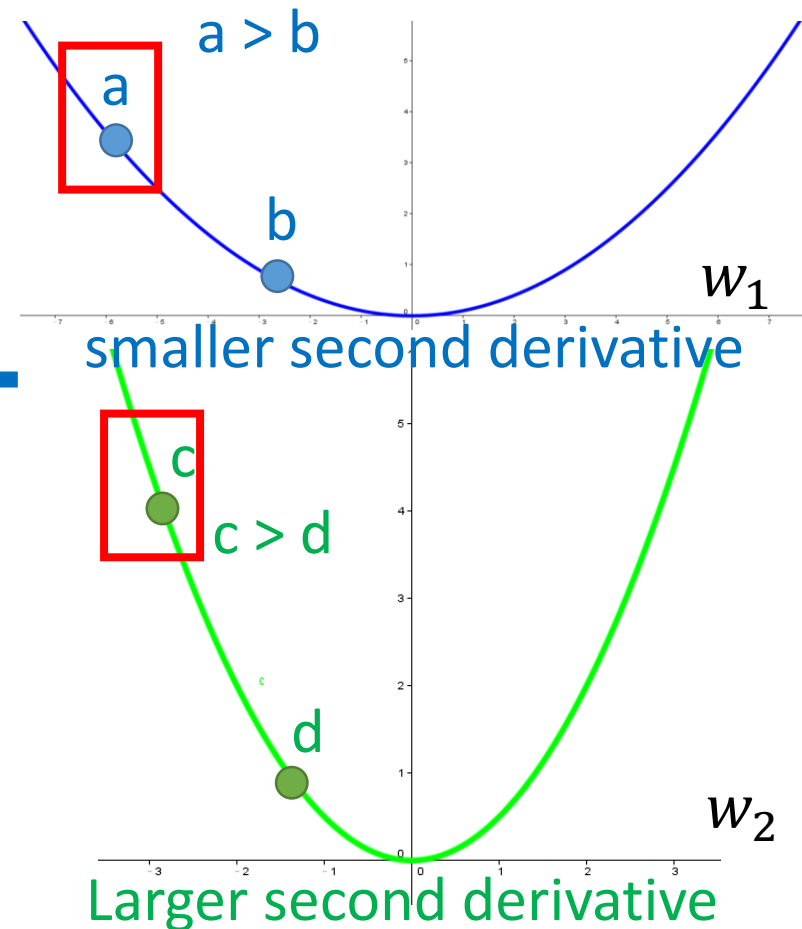
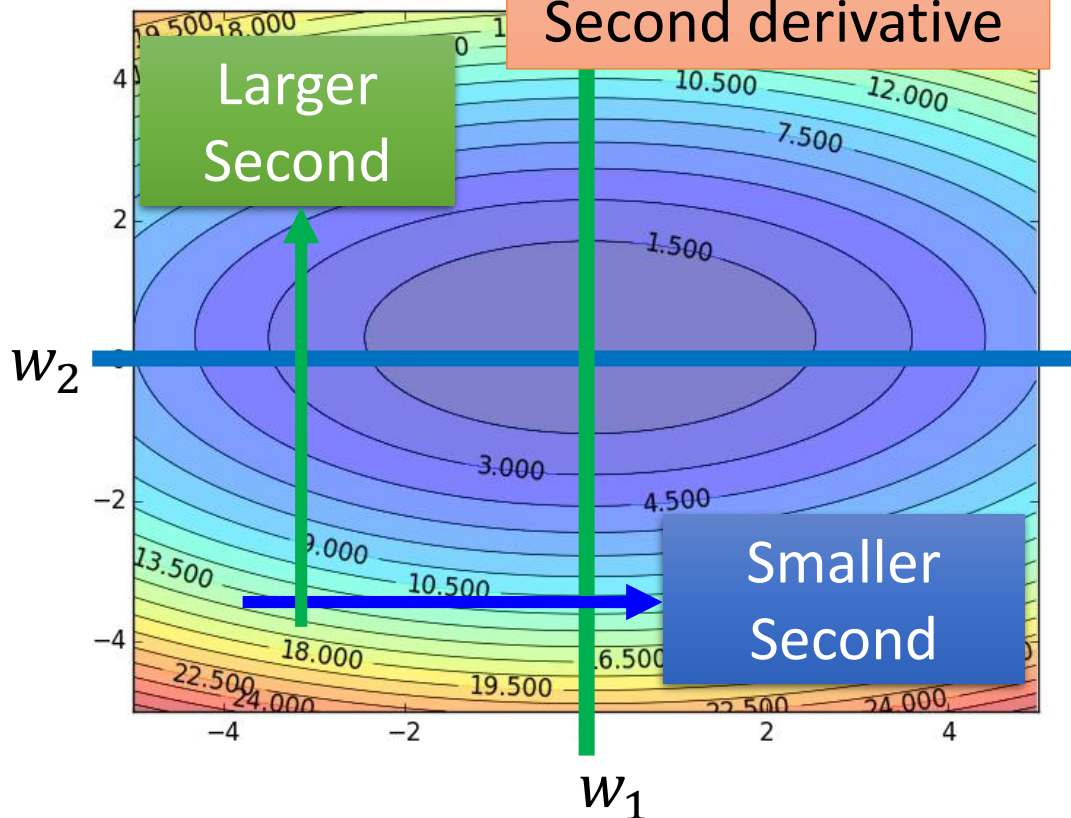
The best step is

| First derivative |

Second derivative

Larger Second

Smaller Second



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

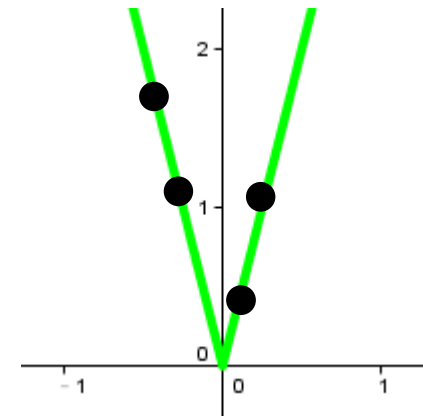
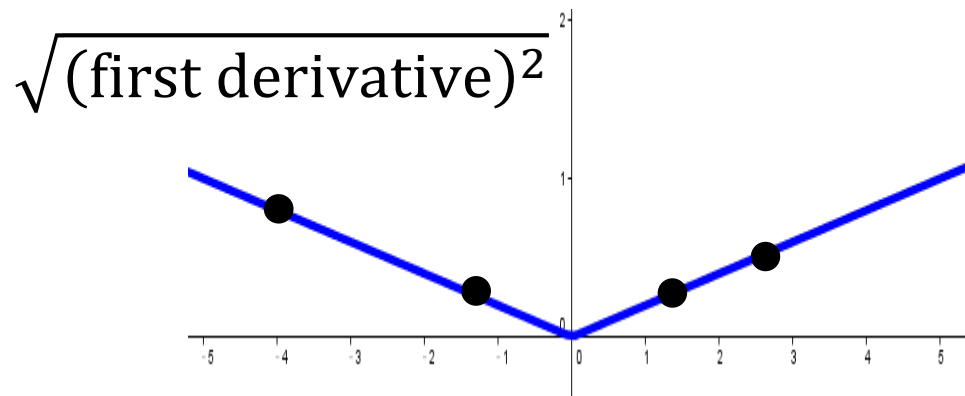
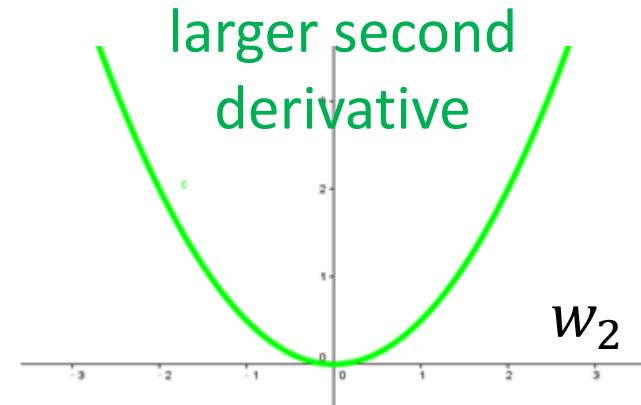
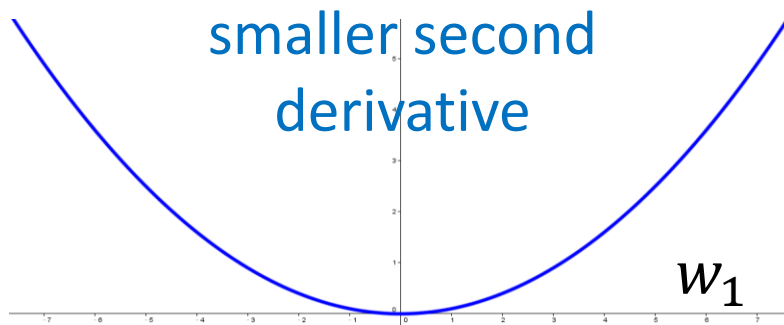
The best step is

| First derivative |

Second derivative

?

Use *first derivative* to estimate *second derivative*



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

◆ **Stochastic Gradient Descent**

Faster!

Pick an example x^n

$$L^n = \left(\hat{y}^n - \left(b + \sum w_i x_i^n \right) \right)^2 \quad \theta^i = \theta^{i-1} - \eta \nabla L^n(\theta^{i-1})$$

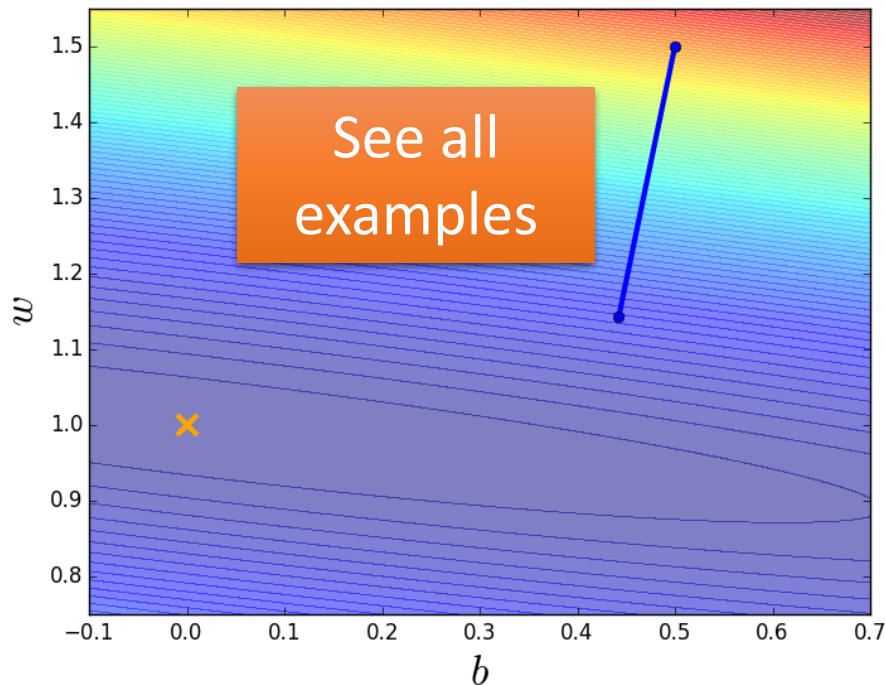
Loss for only one example

- Demo

Stochastic Gradient Descent

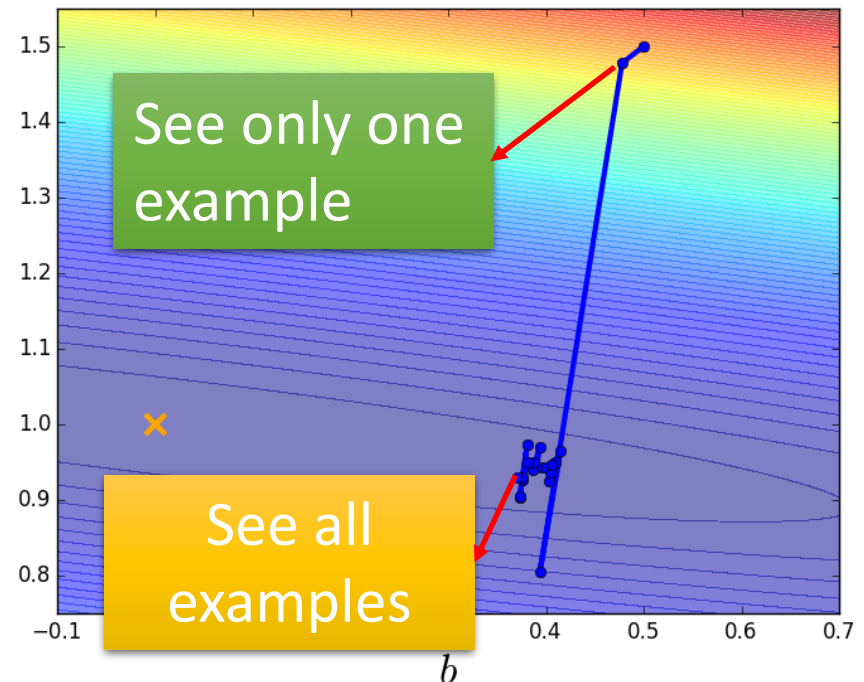
Gradient Descent

Update after seeing all examples



Stochastic Gradient Descent

Update for each example
If there are 20 examples,
20 times faster.



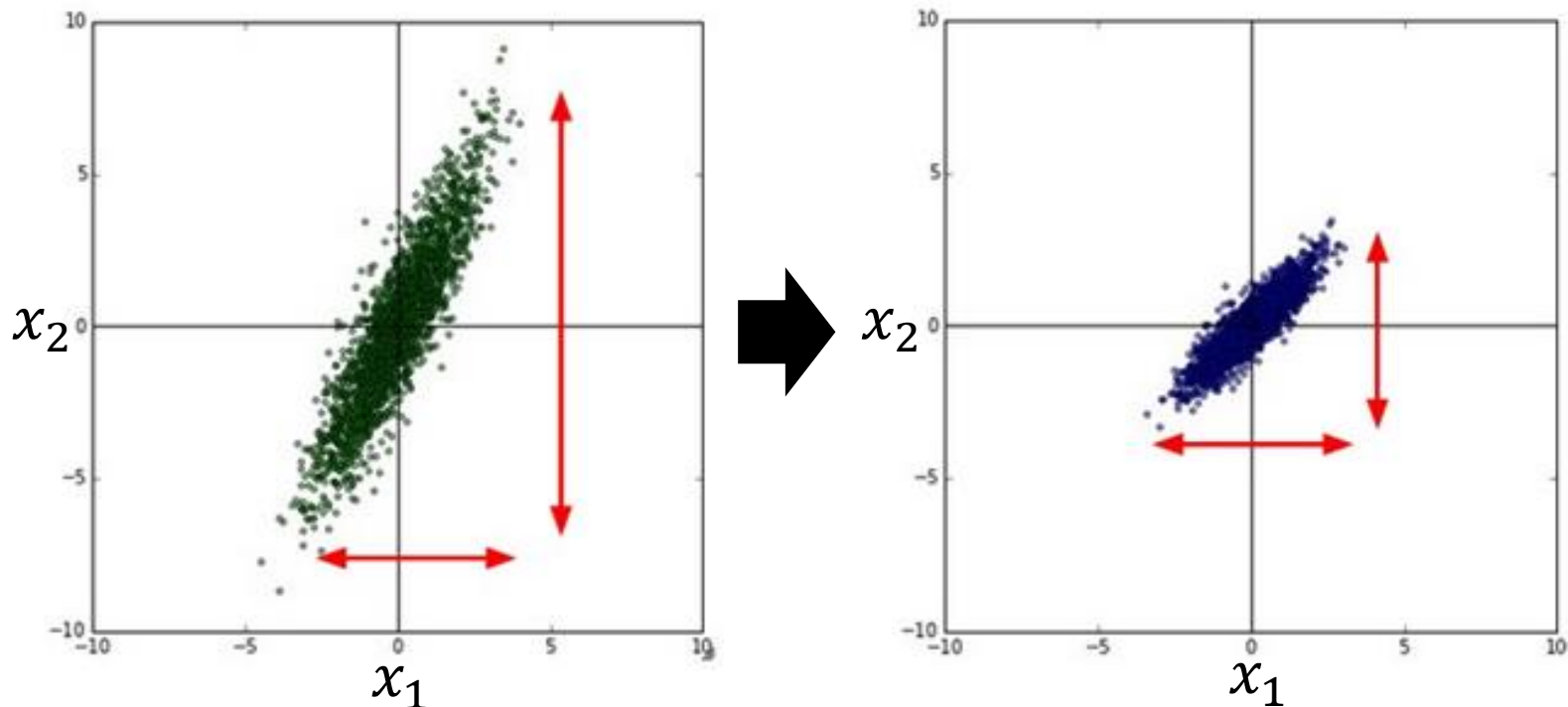
Gradient Descent

Tip 3: Feature Scaling

Feature Scaling

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

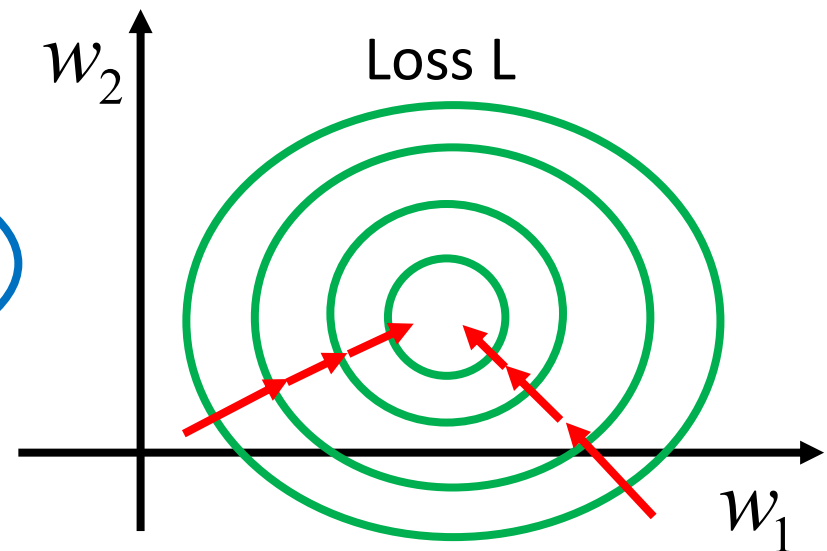
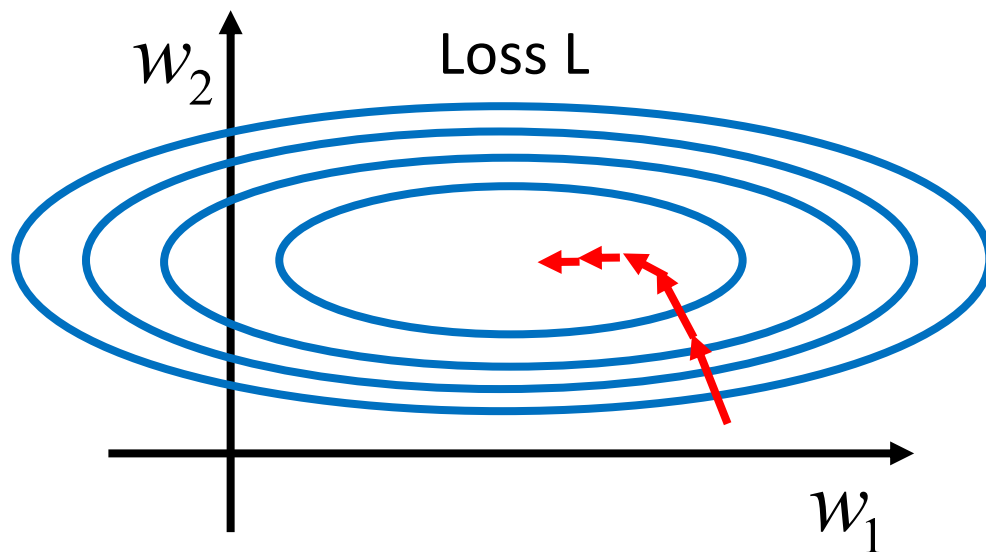
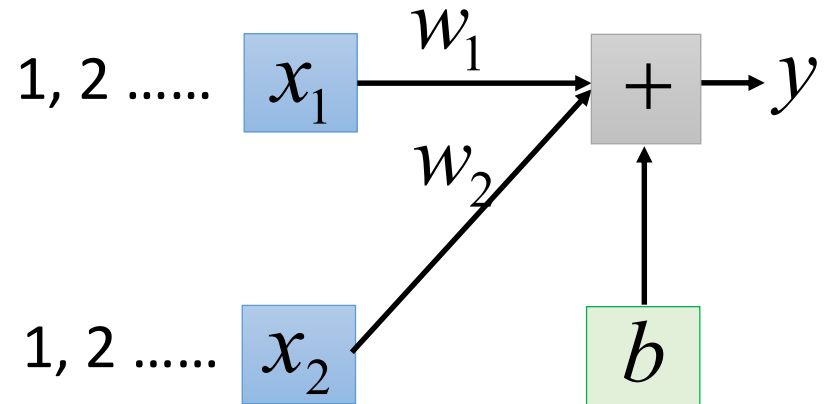
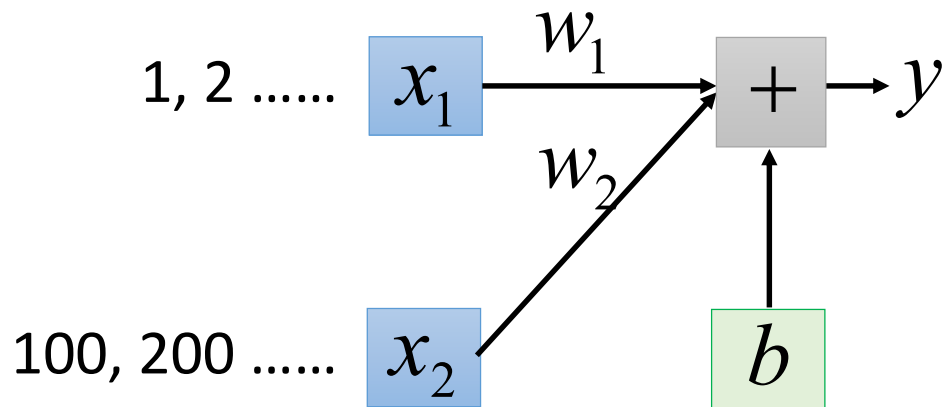
$$y = b + w_1x_1 + w_2x_2$$



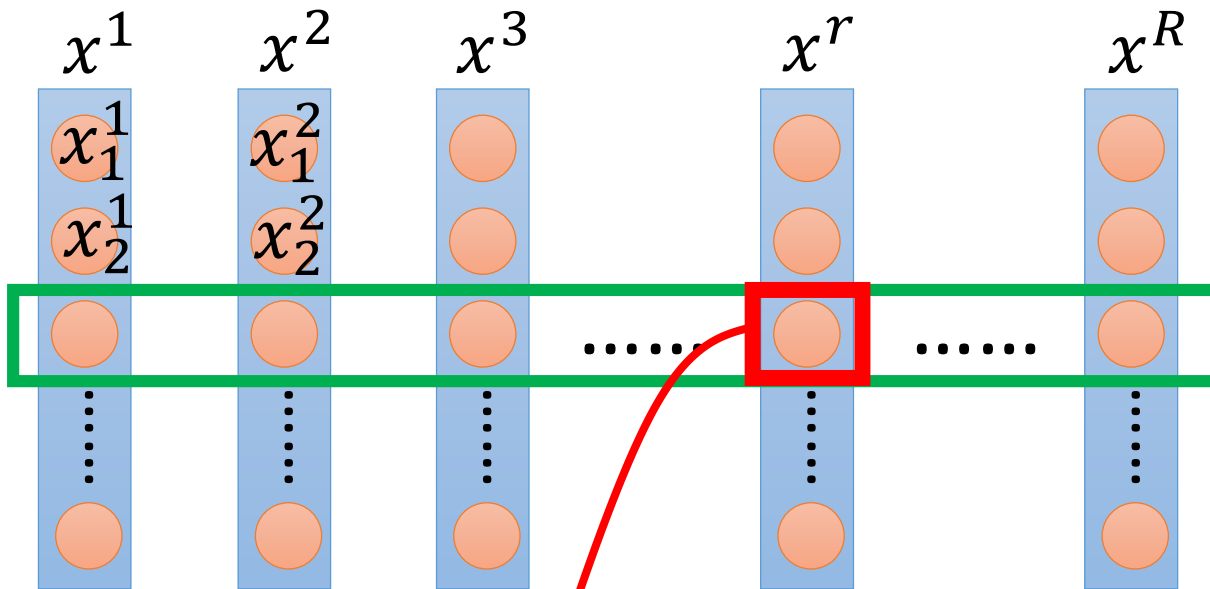
Make different features have the same scaling

Feature Scaling

$$y = b + w_1x_1 + w_2x_2$$



Feature Scaling



For each
dimension i :

mean: m_i

standard
deviation: σ_i

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

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.

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

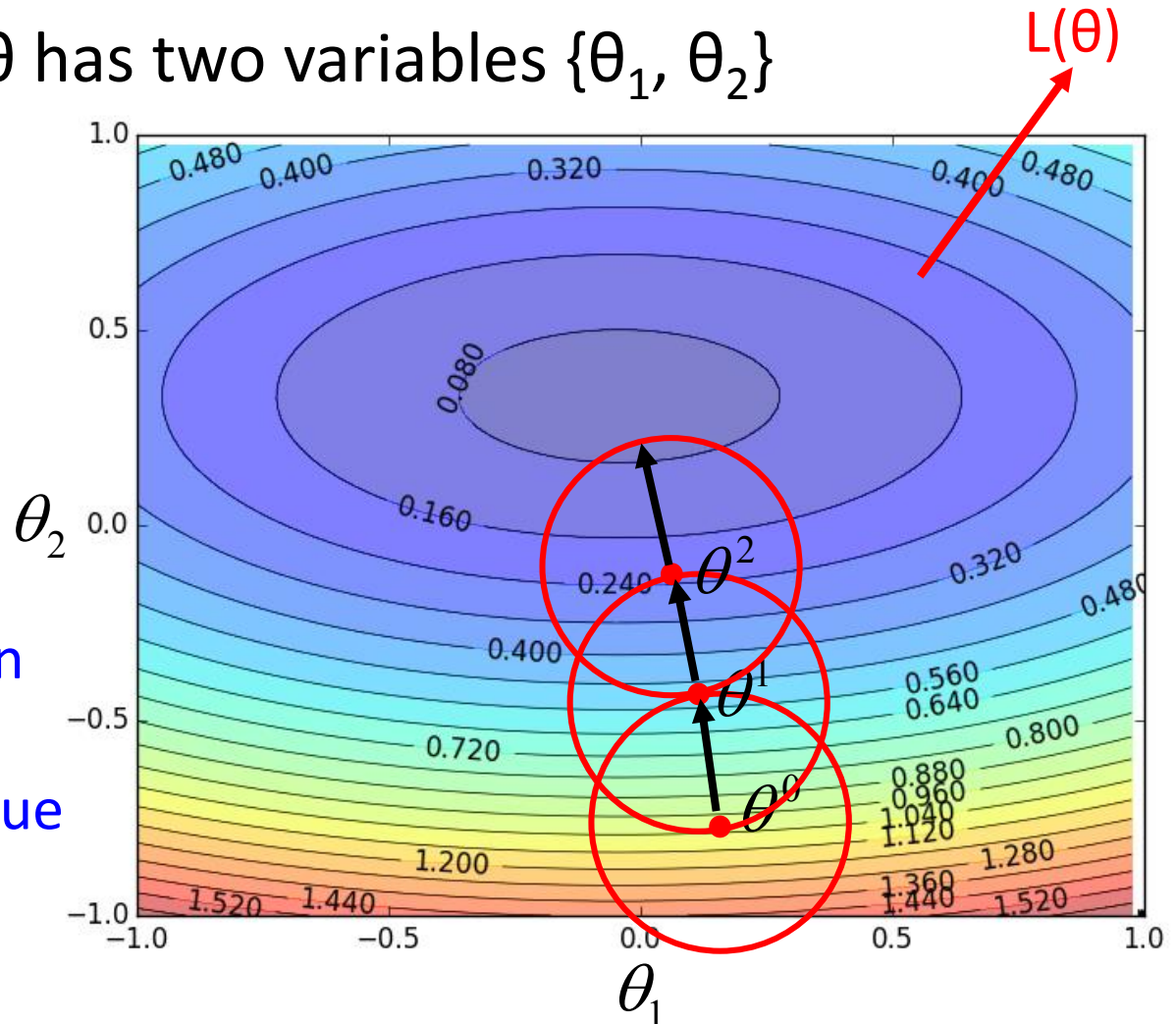
Is this statement correct?

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

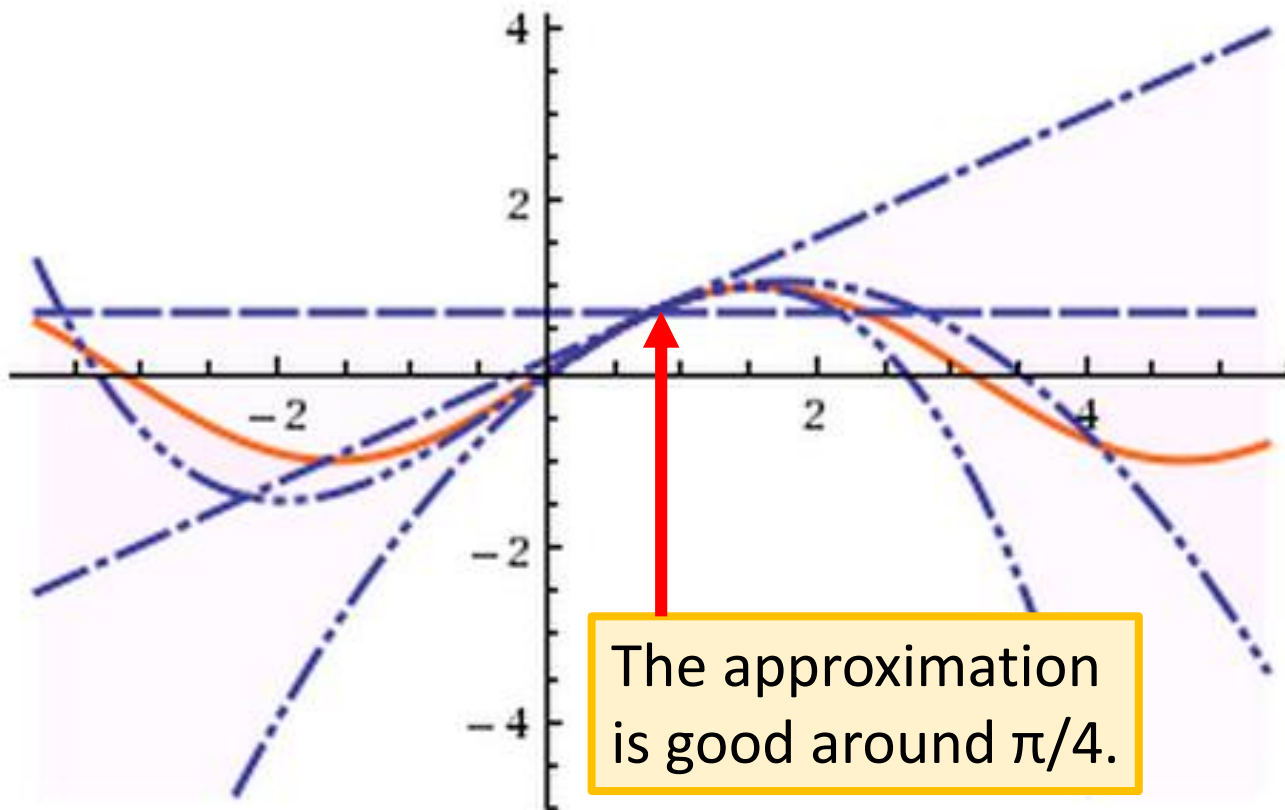
- **Taylor series:** Let $h(x)$ be any function infinitely differentiable around $x = x_0$.

$$\begin{aligned} h(x) &= \sum_{k=0}^{\infty} \frac{h^{(k)}(x_0)}{k!} (x - x_0)^k \\ &= h(x_0) + h'(x_0)(x - x_0) + \frac{h''(x_0)}{2!} (x - x_0)^2 + \dots \end{aligned}$$

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

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

$$\sin(x) = \frac{1}{\sqrt{2}} + \frac{x - \frac{\pi}{4}}{\sqrt{2}} - \frac{\left(x - \frac{\pi}{4}\right)^2}{2\sqrt{2}} - \frac{\left(x - \frac{\pi}{4}\right)^3}{6\sqrt{2}} + \frac{\left(x - \frac{\pi}{4}\right)^4}{24\sqrt{2}} + \frac{\left(x - \frac{\pi}{4}\right)^5}{120\sqrt{2}} - \frac{\left(x - \frac{\pi}{4}\right)^6}{720\sqrt{2}} - \frac{\left(x - \frac{\pi}{4}\right)^7}{5040\sqrt{2}} + \frac{\left(x - \frac{\pi}{4}\right)^8}{40320\sqrt{2}} + \frac{\left(x - \frac{\pi}{4}\right)^9}{362880\sqrt{2}} - \frac{\left(x - \frac{\pi}{4}\right)^{10}}{3628800\sqrt{2}} + \dots$$



The approximation is good around $\pi/4$.

Multivariable Taylor Series

$$h(x, y) = h(x_0, y_0) + \frac{\partial h(x_0, y_0)}{\partial x} (x - x_0) + \frac{\partial h(x_0, y_0)}{\partial y} (y - y_0) \\ + \text{something related to } (x - x_0)^2 \text{ and } (y - y_0)^2 + \dots$$

When x and y is close to x_0 and y_0



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

Back to Formal Derivation

Based on Taylor Series:

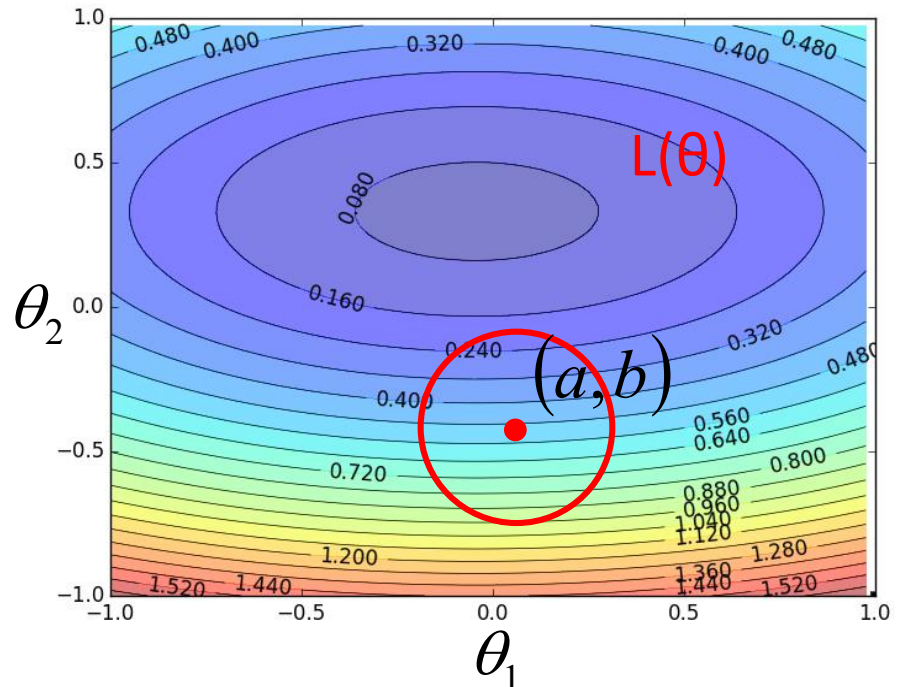
If the red circle is small enough, in the red circle

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

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

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

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



Back to Formal Derivation

Based on Taylor Series:

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

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

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

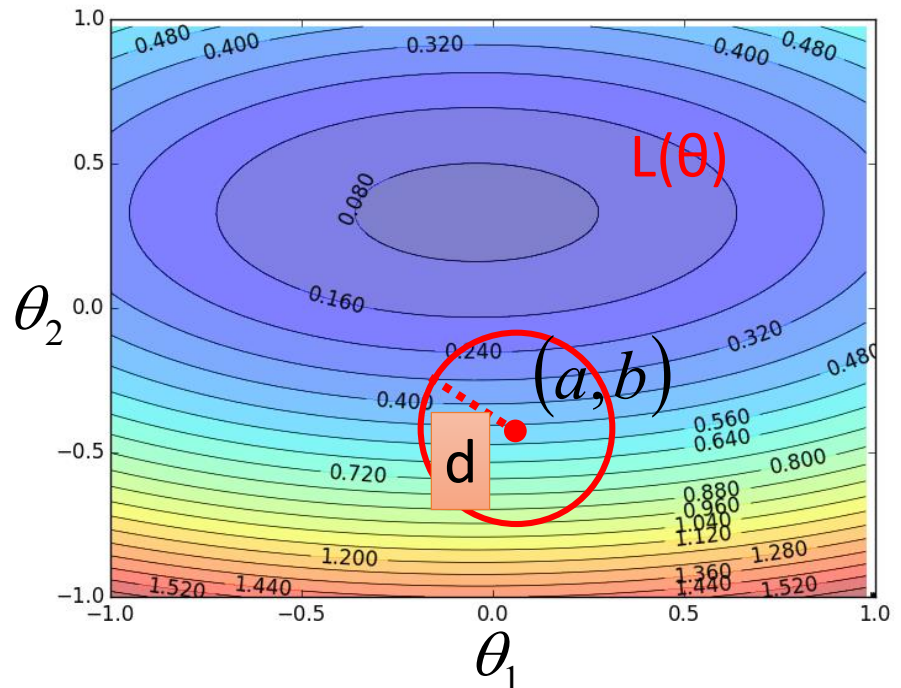
$$(\theta_1 - a)^2 + (\theta_2 - b)^2 \leq d^2$$

Simple, right?

constant

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

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



Gradient descent – two variables

Red Circle: (If the radius is small)

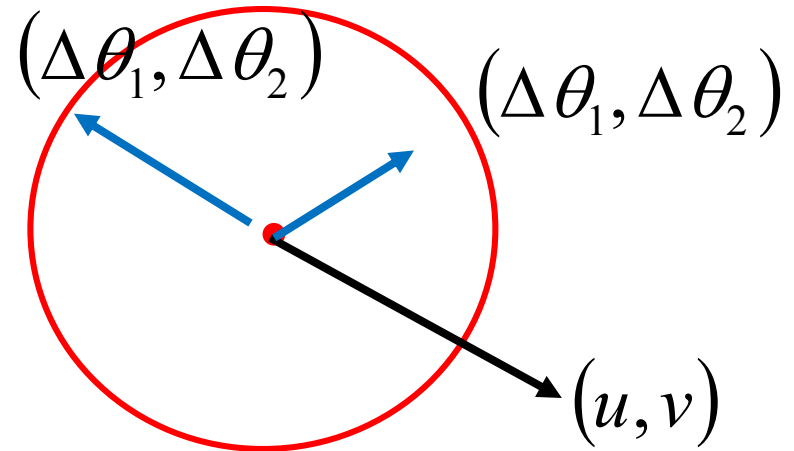
$$L(\theta) \approx \cancel{s} + u \underbrace{(\theta_1 - a)}_{\Delta \theta_1} + v \underbrace{(\theta_2 - b)}_{\Delta \theta_2}$$

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

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

To minimize $L(\theta)$

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



Back to Formal Derivation

Based on Taylor Series:

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

constant

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

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

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

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

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

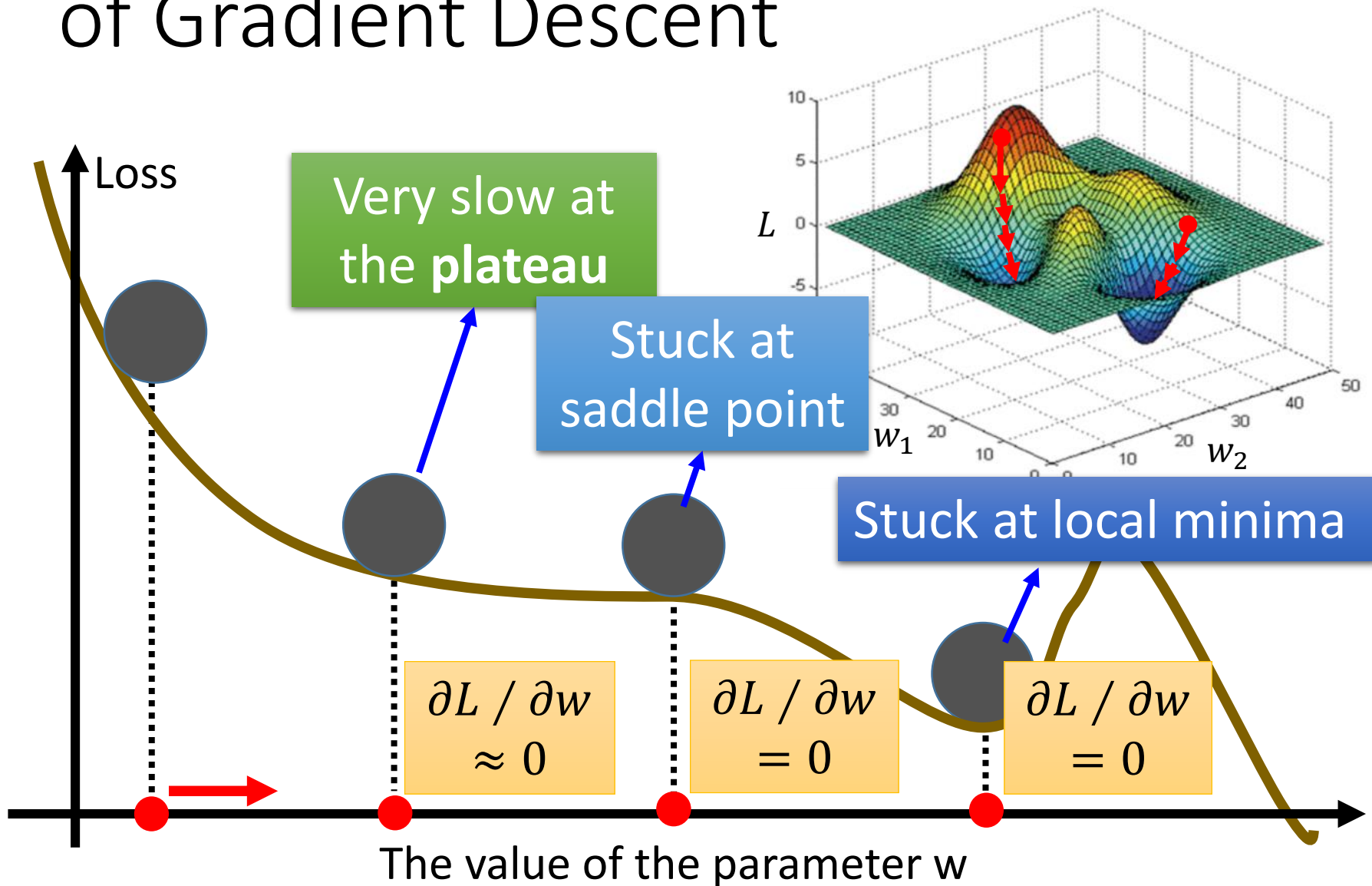
This is gradient descent.

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