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 $\{\theta_1, \theta_2\}$ Index of features (Ignore bias b)

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

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

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

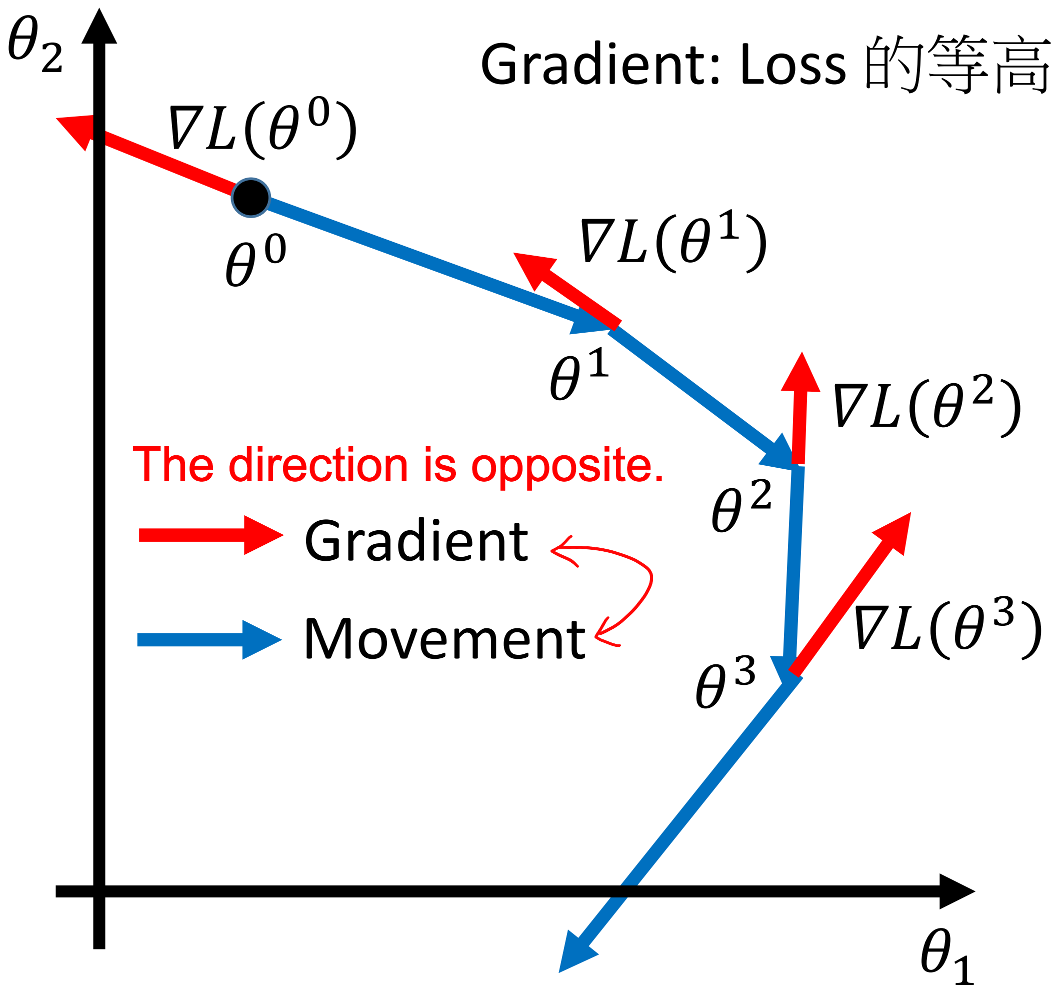
Index of iterations

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

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

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

⋮

The solver for weight optimization

1. Tuning your learning rates
2. Stochastic gradient descent
3. Feature scaling

Standardization

Gradient Descent

Tip 1: Tuning your learning rates

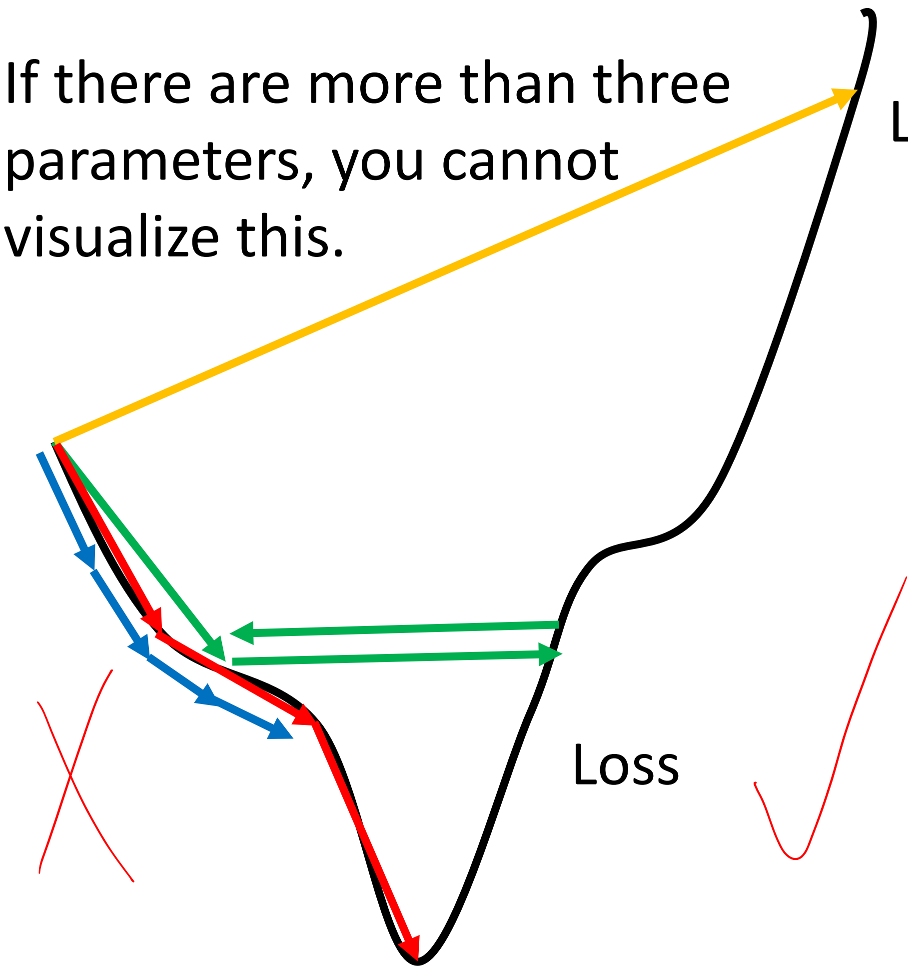
Learning Rate

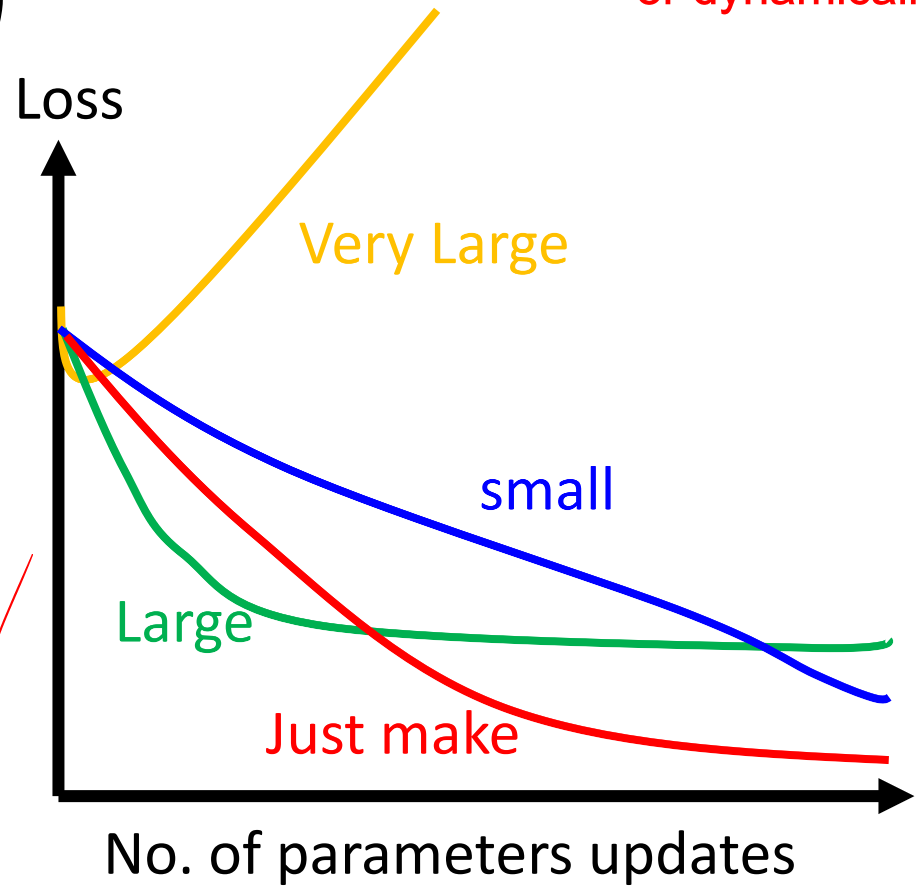
$$\theta^{i} = \theta^{i-1} - \eta \nabla L\left(\theta^{i-1}\right)$$

Set the learning rate η carefully

or dynamically

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 ⇒ It should decay
  - Giving different parameters different learning rates

η: scalar

Adagrad

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

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

Trivial

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

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

Adagrad

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

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

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

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

$$\vdots$$

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

Adagrad

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

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

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

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

Simplify

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

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

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

Larger gradient, larger step

Larger gradient, smaller step

Intuitive Reason

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

- How surprise it is 反差

特别大

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	……

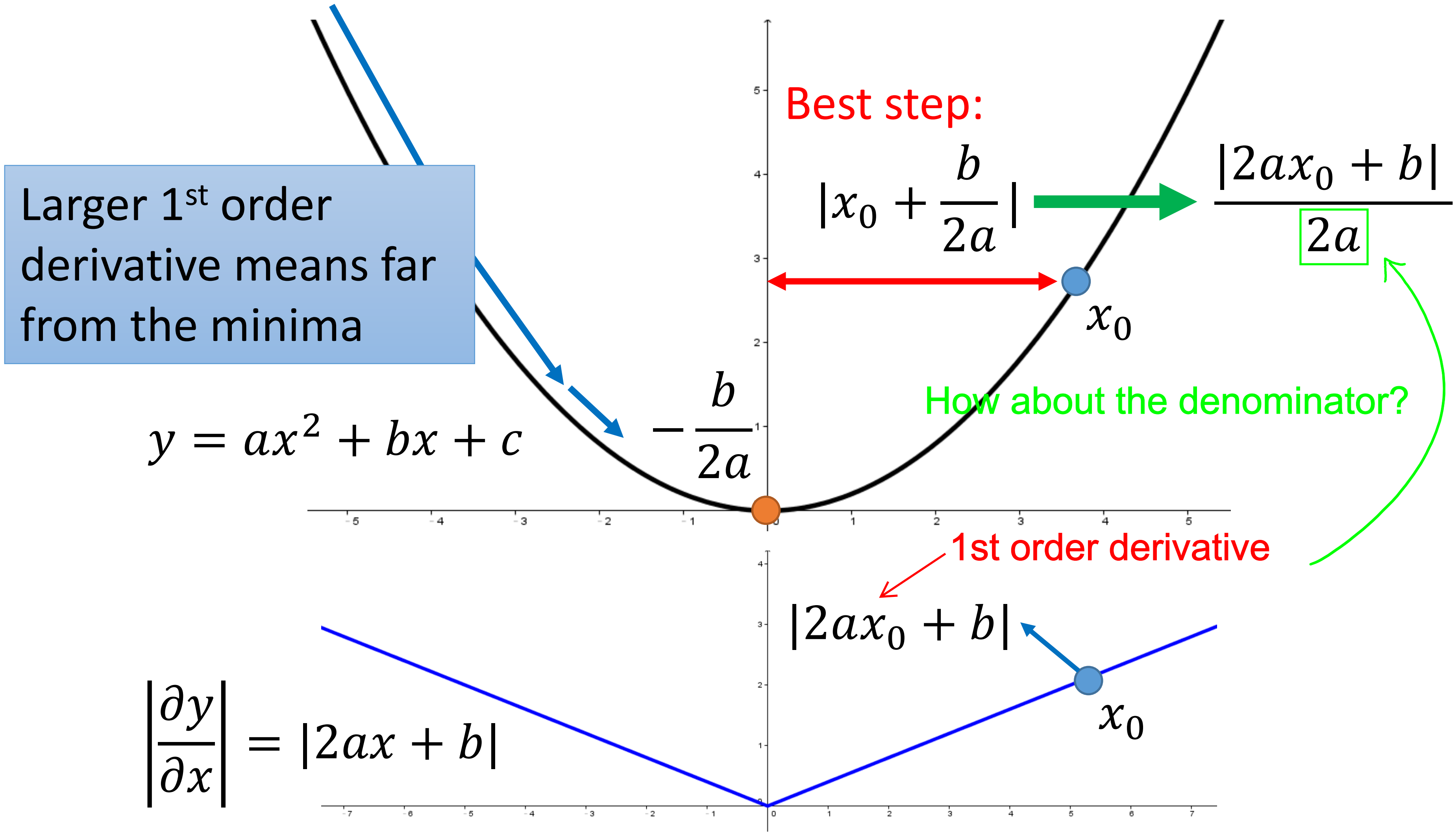
特别小

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

造成反差的效果

$$\frac{\text{Very big}}{\text{Very small}} = \text{Super big}$$

Larger gradient, larger steps?

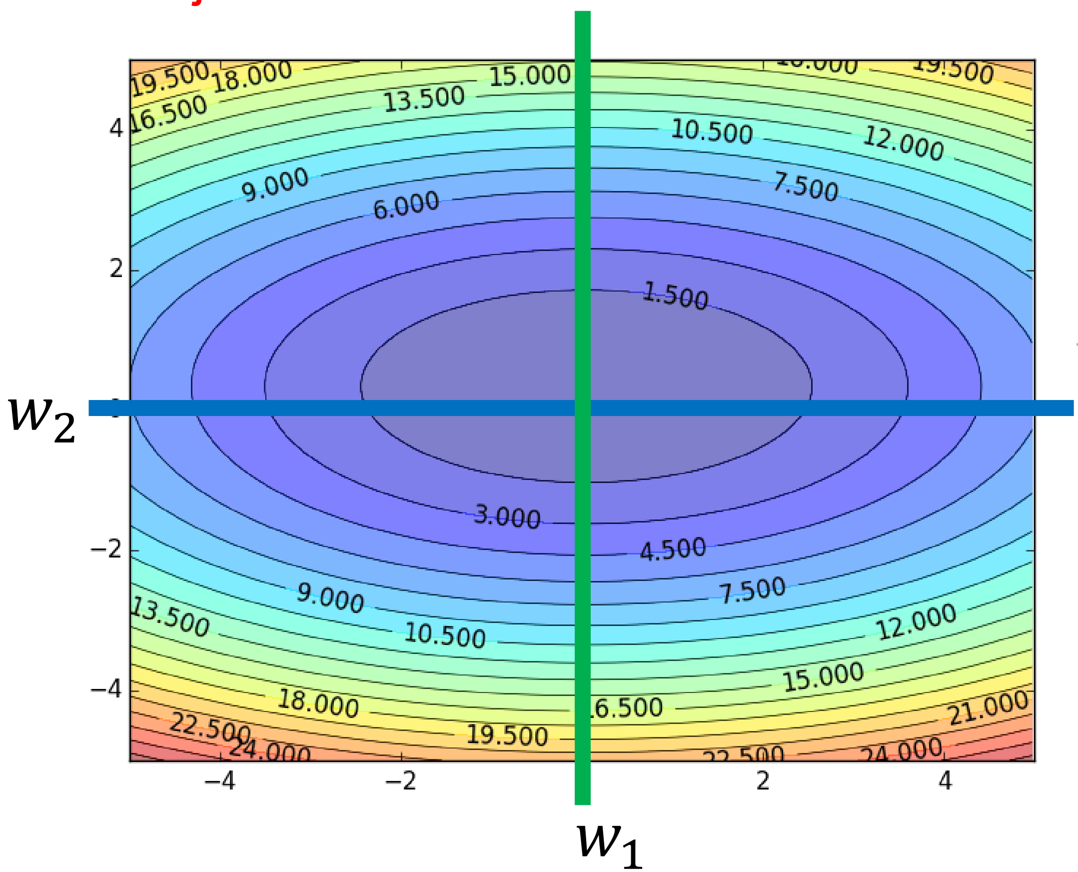


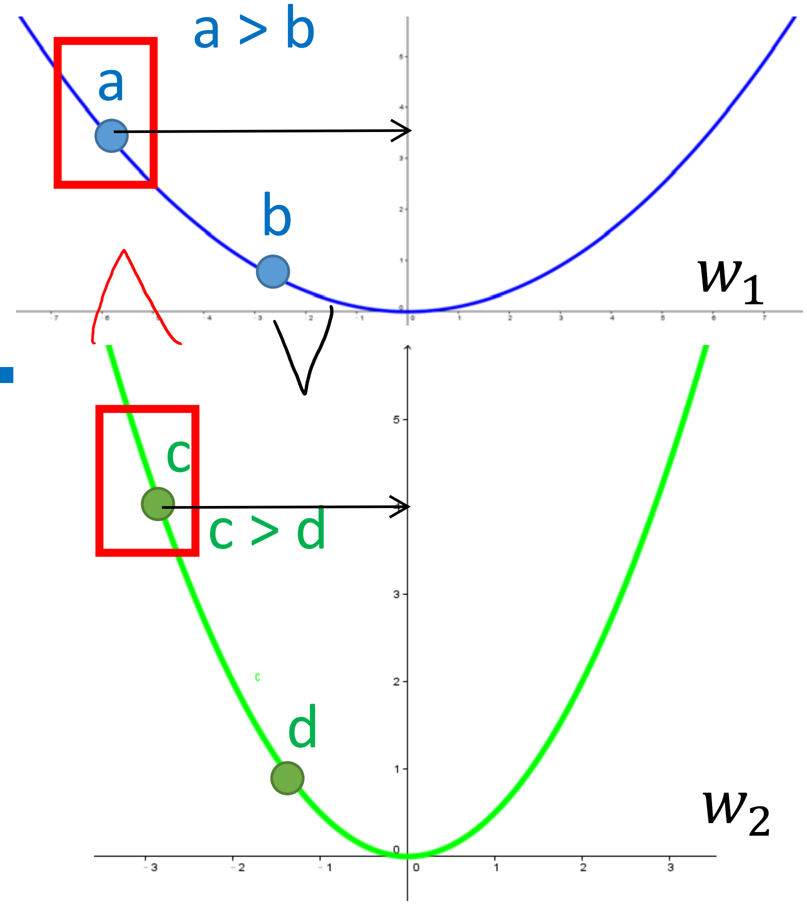
Comparison between different parameters

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

Do not cross parameters

We can't just use the 1st order derivative to determine the steps.

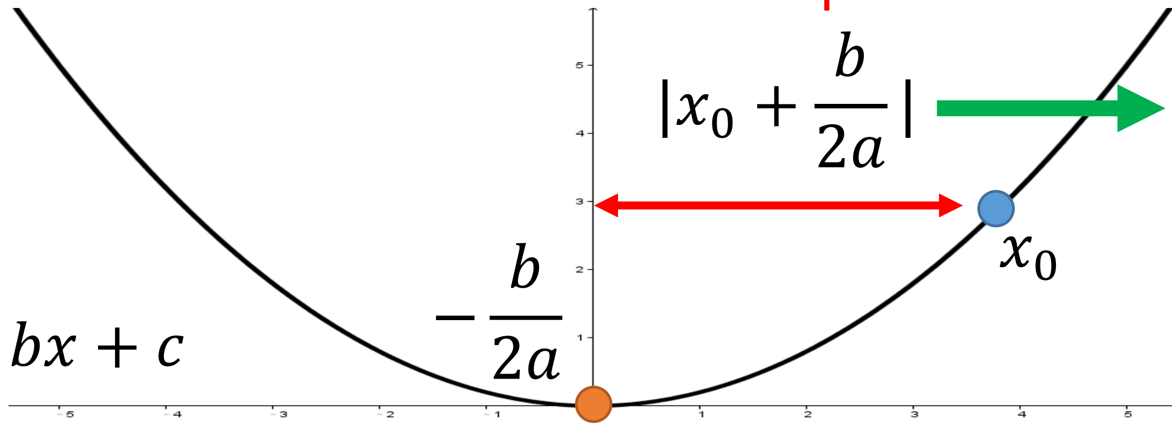




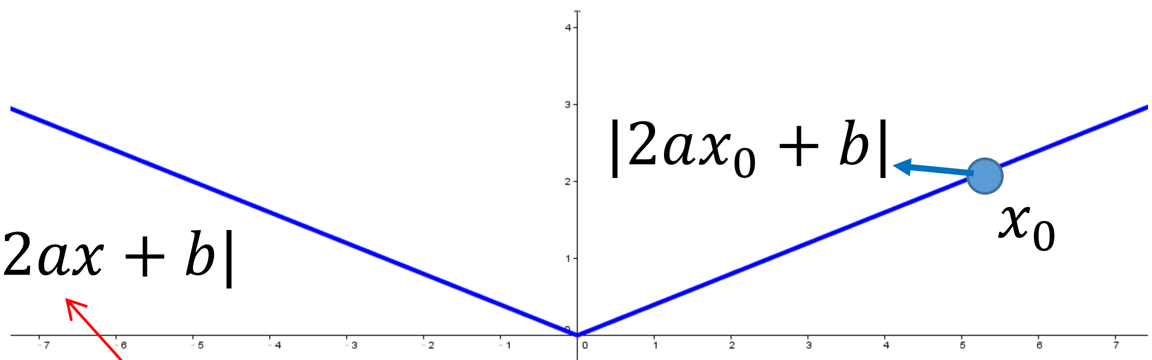
Second Derivative

Best step:

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



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



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

1st order derivative

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

2nd order derivative

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

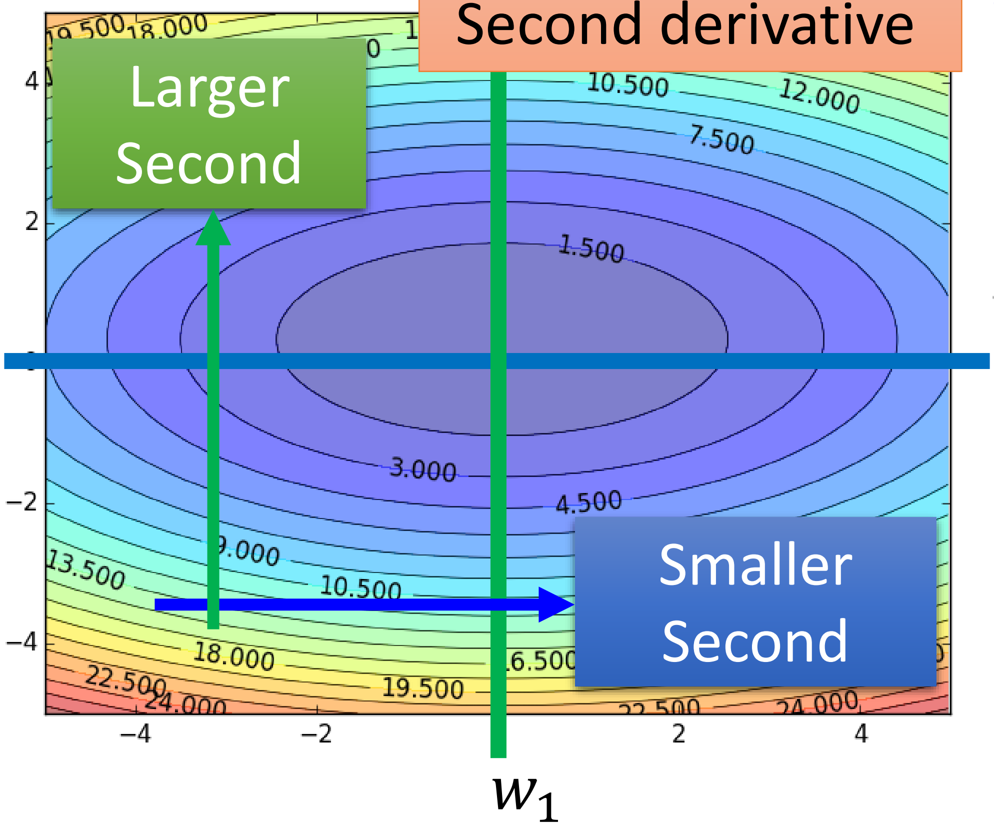
Comparison between different parameters

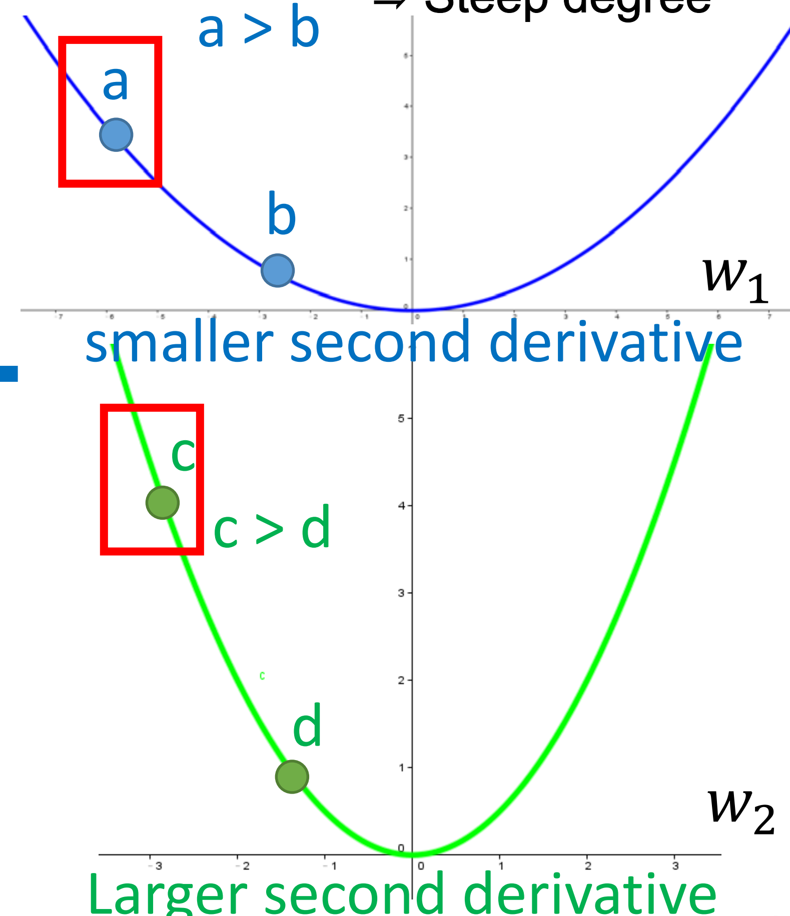
~~Larger 1st order derivative means far from the minima~~

Do not cross parameters

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

2nd order derivative
⇒ Steep degree





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

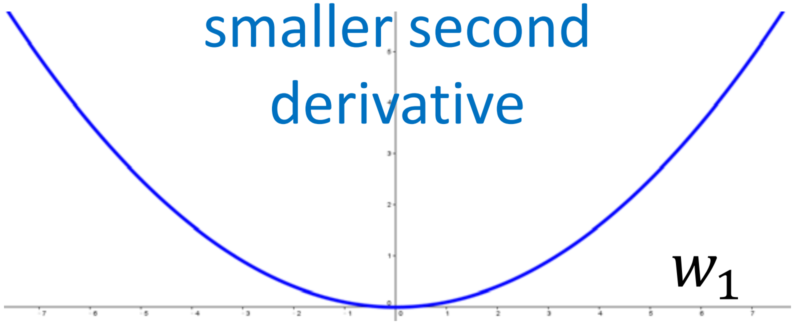
?

The best step is

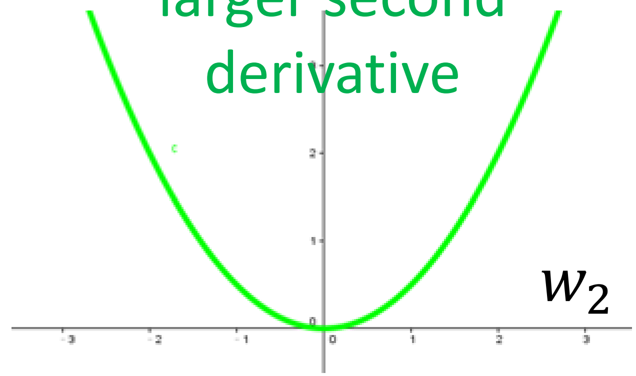
$$\frac{|\text{First derivative}|}{\text{Second derivative}}$$

<u>Use *first derivative* to estimate *second derivative*</u>

smaller second derivative

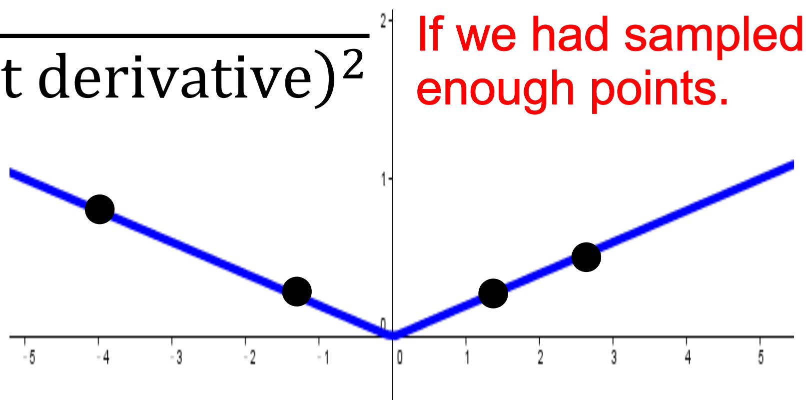


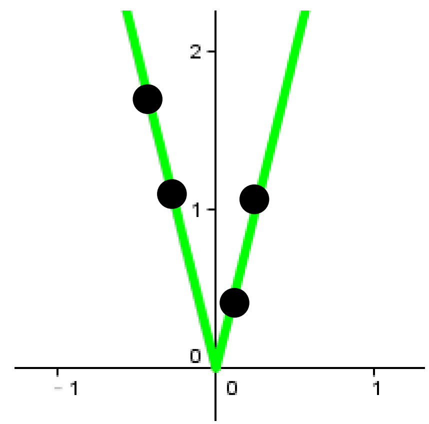
larger second derivative



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

If we had sampled enough points.





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

It would be very helpful if the error surface was not convex.

We can choose the sample data randomly or in order.

◆ ***Stochastic Gradient Descent***

Faster!

Pick an example x^n

Update the model per sample data instead of the whole data set.

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

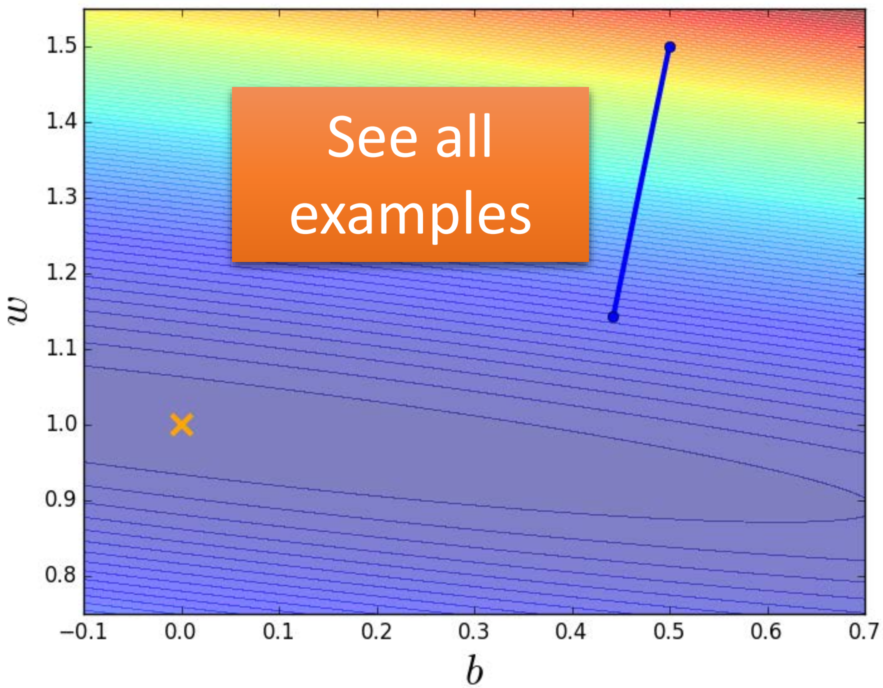
Loss for only one example

We still need to use up all sample data before using the same sample data again.

Stochastic Gradient Descent

Gradient Descent

Update after seeing all examples



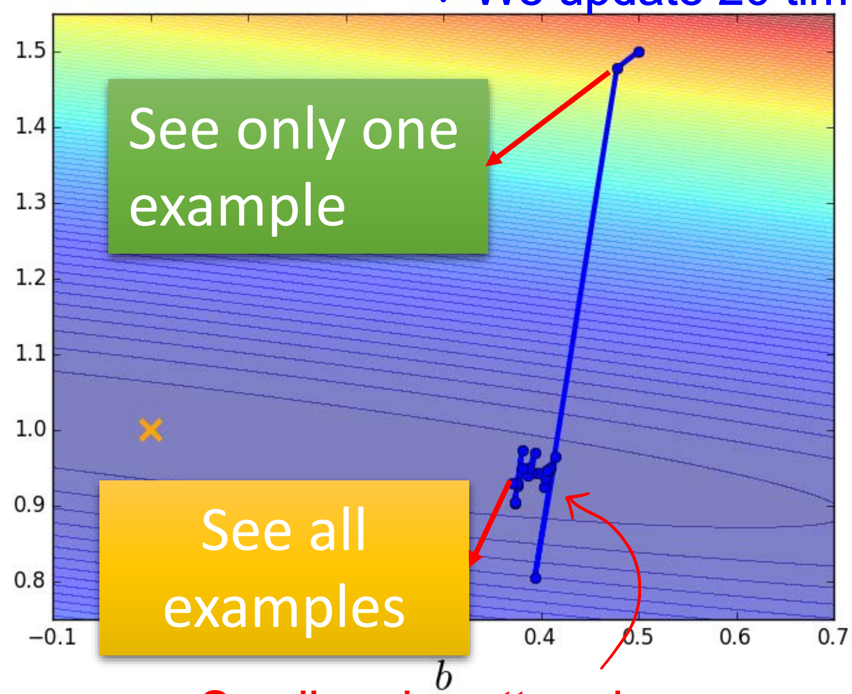
Stochastic Gradient Descent

Update for each example

If there are 20 examples,

20 times faster.

∵ We update 20 times.



Small and scattered paces

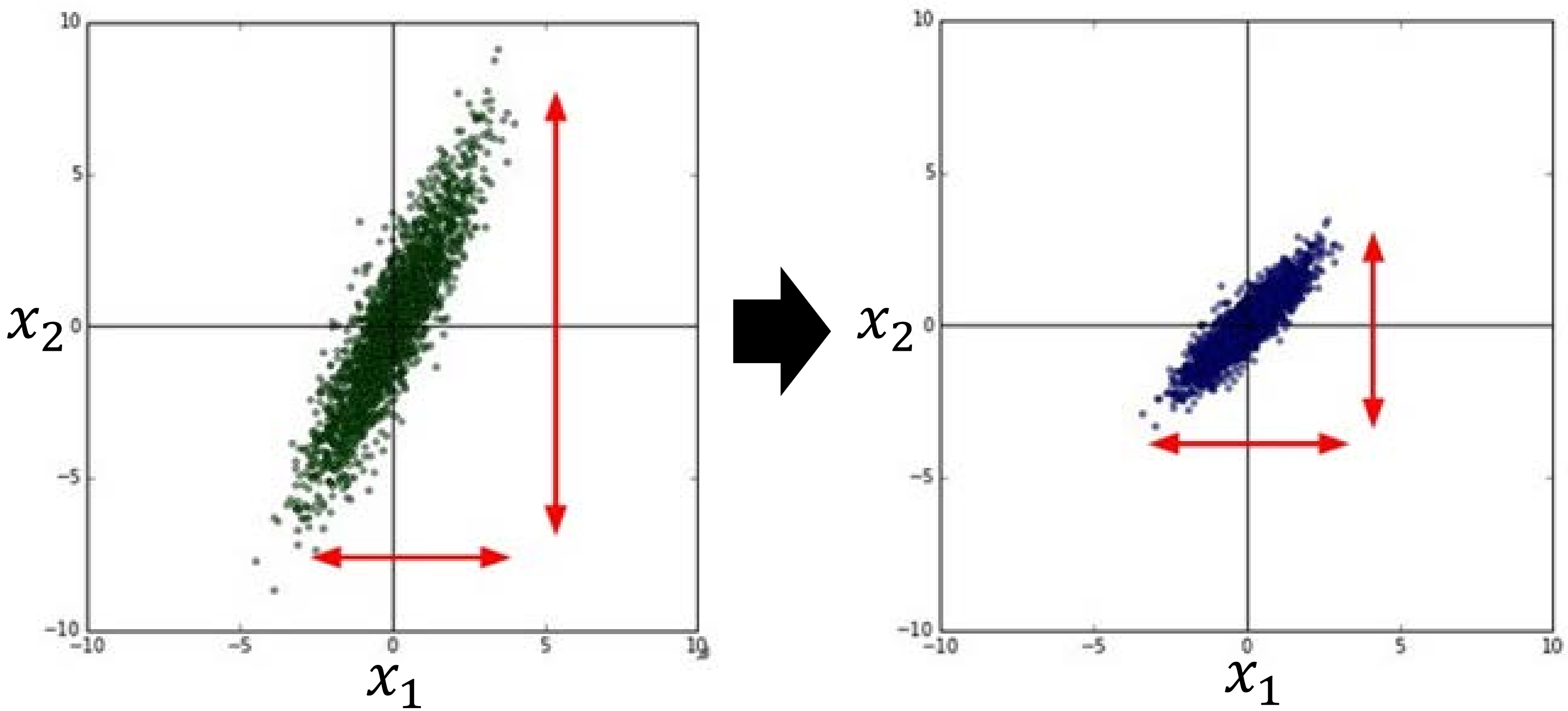
Gradient Descent

Tip 3: Feature Scaling

Feature Scaling

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

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

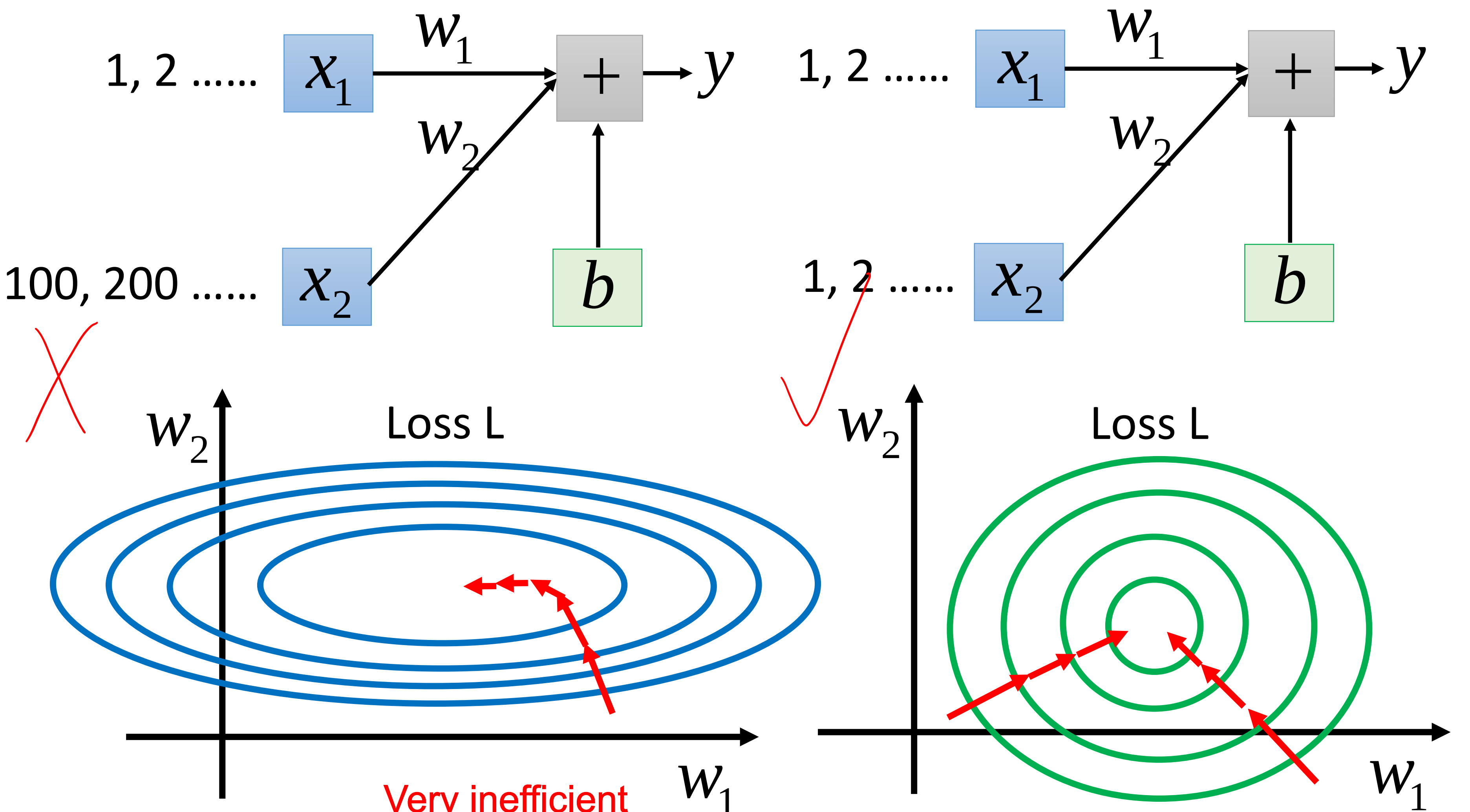


Make different features have the same scaling

impact

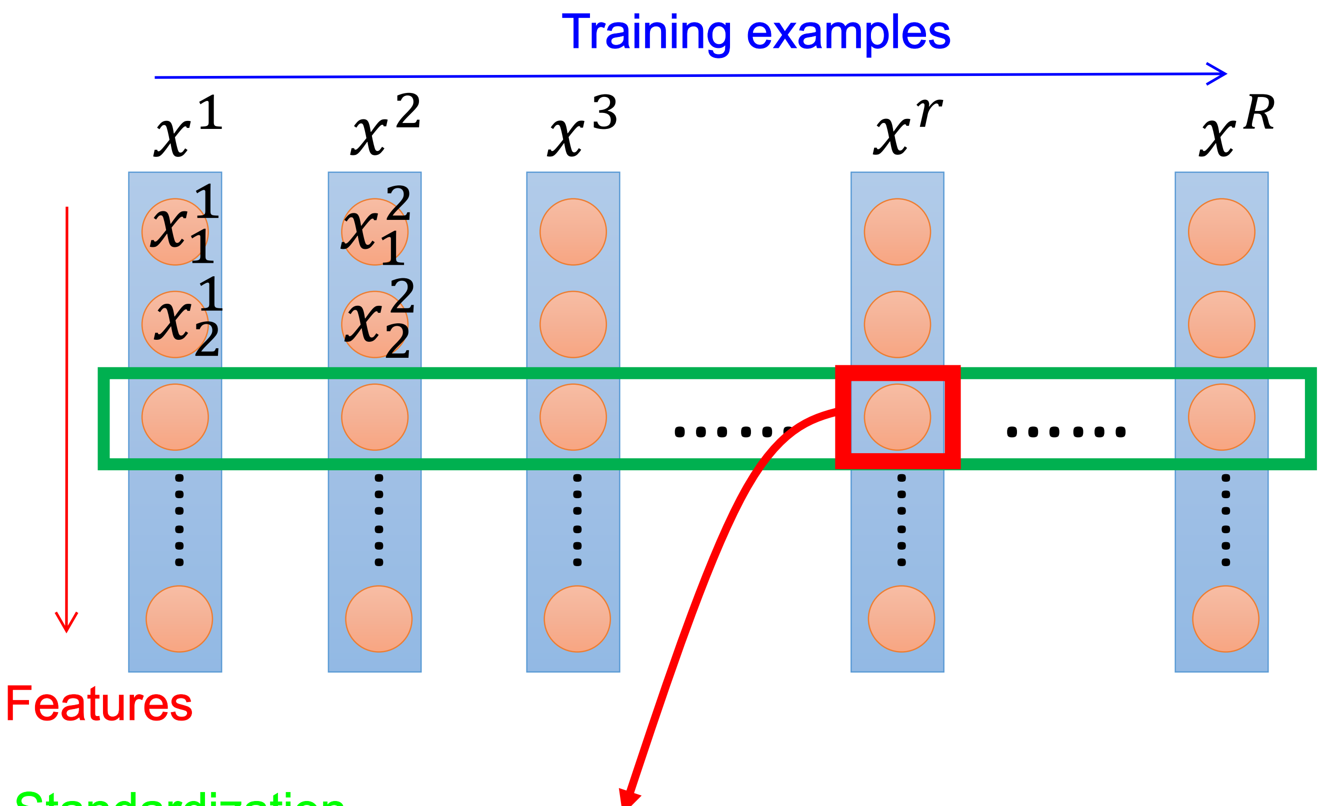
Feature Scaling

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



Feature Scaling

Normalization: rescale into [0, 1]
(In sklearn, it is MinMaxScaler instead of normalizer.)



For each dimension i:

mean: m_i

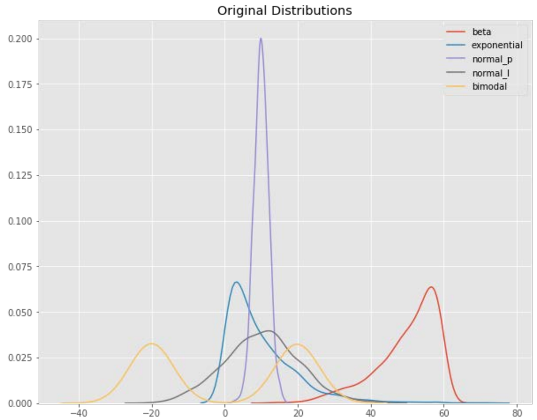
standard deviation: σ_i

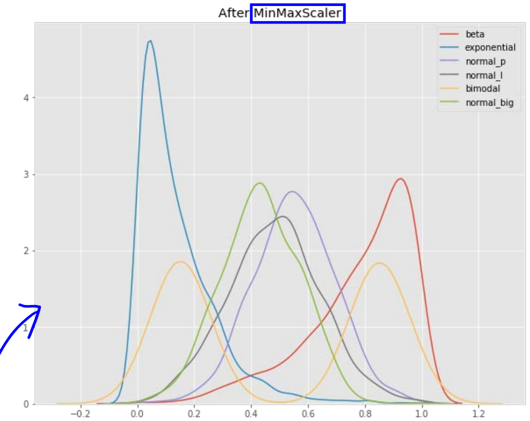
Standardization

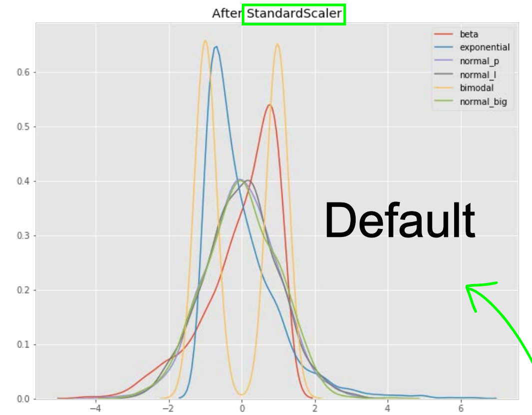
$$x_i^r \leftarrow \frac{x_i^r - m_i}{\sigma_i}$$

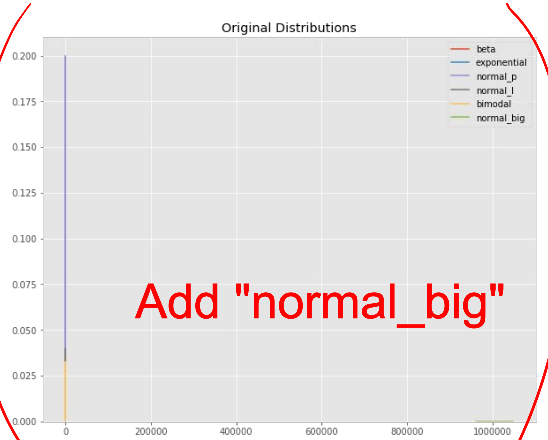
The means of all dimensions are 0, and the variances are all 1

(Mean removal + variance scaling)









An alternative standardization is scaling features to lie between a given minimum and maximum value, often between zero and one, or so that the maximum absolute value of each feature is scaled to unit size. This transformation is often used as an alternative to zero mean, unit variance scaling.

Standardization of a dataset is a common requirement for many machine learning estimators: they might behave badly if the individual features do not more or less look like standard normally distributed data (e.g. Gaussian with 0 mean and unit variance).

Preprocessing Type	Scikit-learn Function	Range	Mean	Distribution Characteristics	When Use	Definition	Notes
Scale	MinMaxScaler	0 to 1 default, can override	varies	Bounded	Use first unless have theoretical reason to need stronger medicine.	Add or substract a constant. Then multiply or divide by another constant. MinMaxScaler subtracts the mimimum value in the column and then divides by the difference between the original maximum and original minimum.	Preserves the shape of the original distribution. Doesn't reduce the importance of outliers. Least disruptive to the information in the original data. Default range for MinMaxScaler is 0 to 1.
Standardize	RobustScaler	varies	varies	Unbounded	Use if have outliers and don't want them to have much influence.	RobustScaler standardizes a feature by removing the median and dividing each feature by the interquartile range.	Outliers have less influence than with MinMaxScaler. Range is larger than MinMaxScaler or StandardScaler.
Standardize	StandardScaler	varies	0	Unbounded, Unit variance	When need to transform a feature so it is close to normally distributed.	StandardScaler standardizes a feature by removing the mean and dividing each value by the standard deviation.	Results in a distribution with a standard deviation equal to 1 (and variance equal to 1). If you have outliers in your feature (column), normalizing your data will scale most of the data to a small interval.
Normalize	Normalizer	varies	0	Unit norm	Rarely.	An observation (row) is normalized by applying l2 (Euclidian) normalization. If each element were squared and summed, the total would equal 1. Could also specify l1 (Manhatten) normalization.	Normalizes each sample observation (row), not the feature (column)!

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.

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

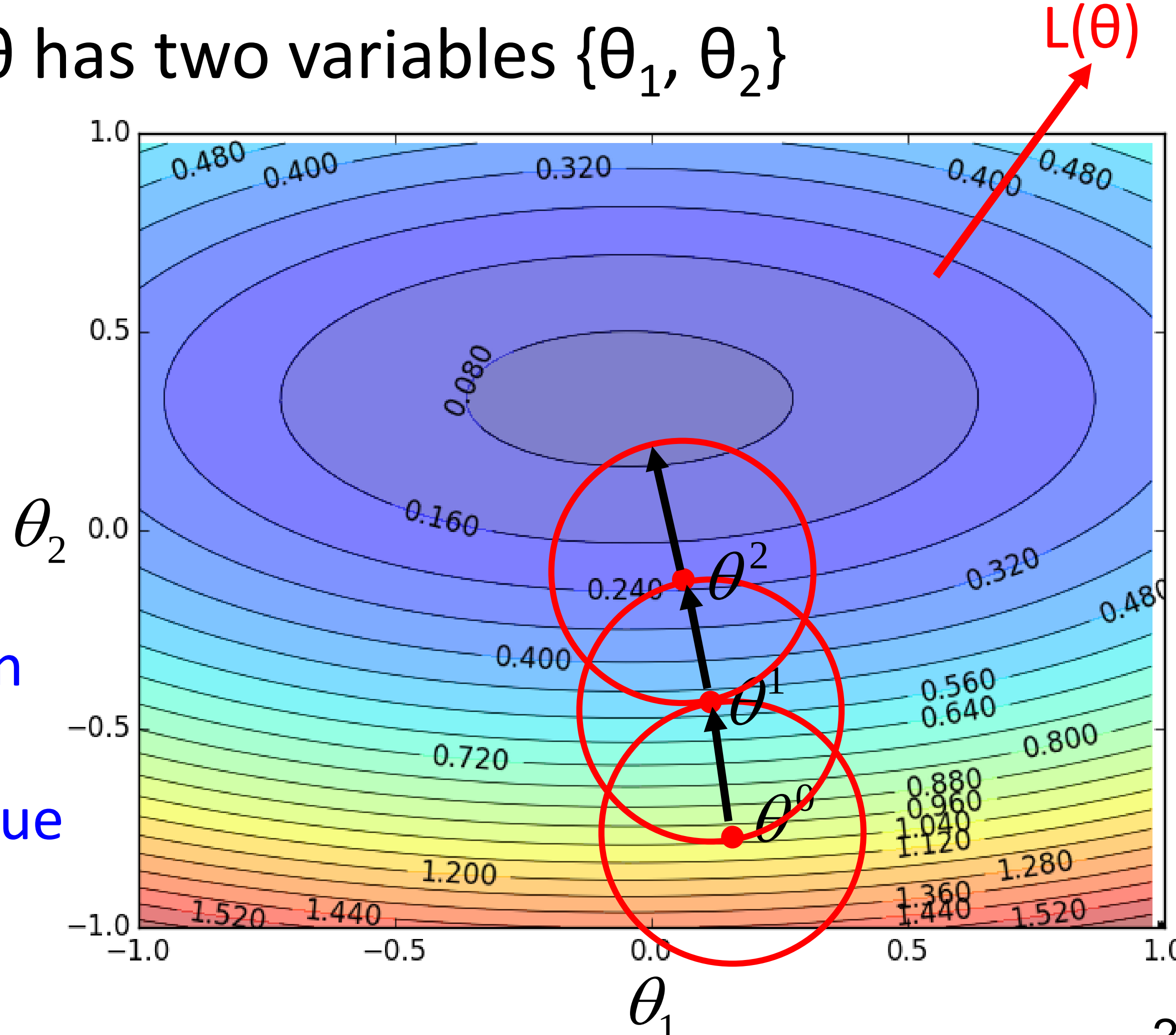
Is this statement correct? No

Warning of Math

The proof of gradient descent

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 $x = x_0$.

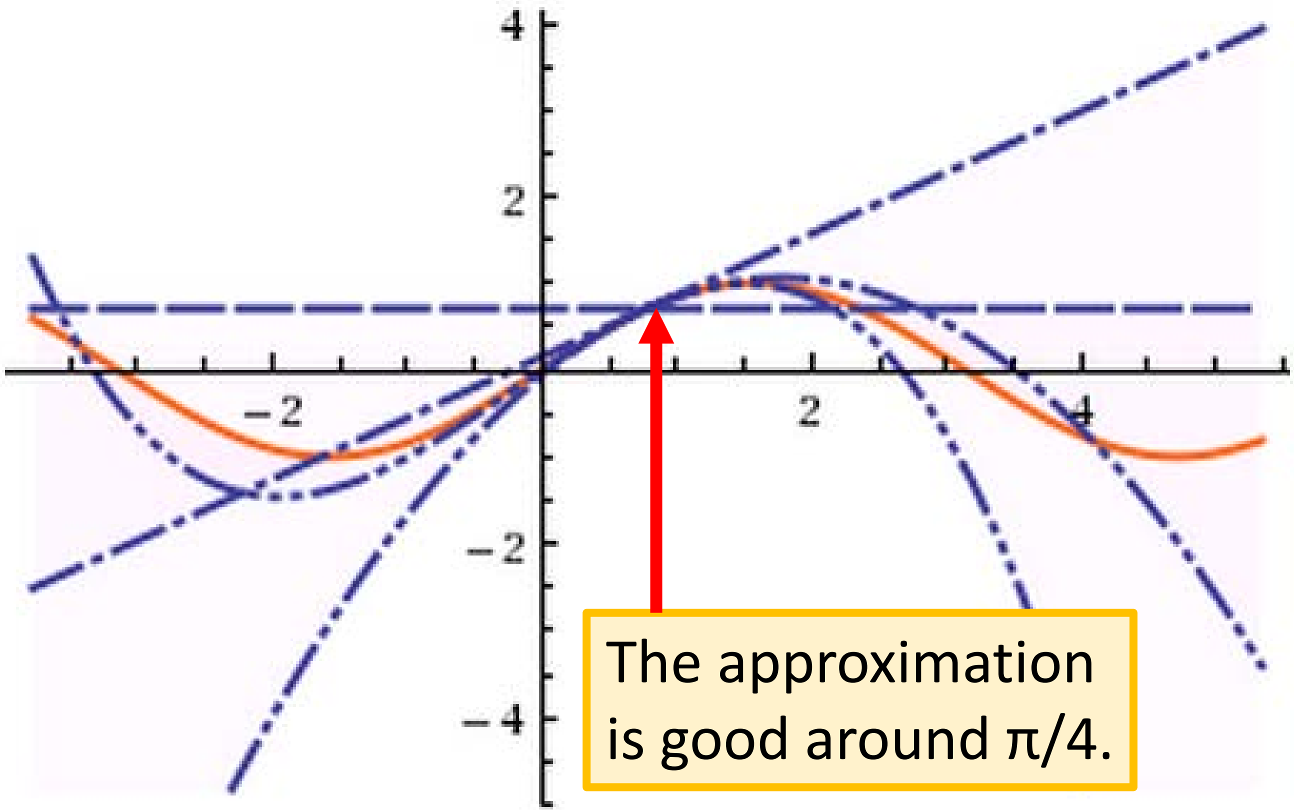
$$\mathrm{h}(\mathrm{x}) = \sum_{k=0}^{\infty} \frac{\mathrm{h}^{(k)}(x_0)}{k!}(x - x_0)^k$$

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

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 $(x\text{-}x_0)^2$ and $(y\text{-}y_0)^2$ + ……

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

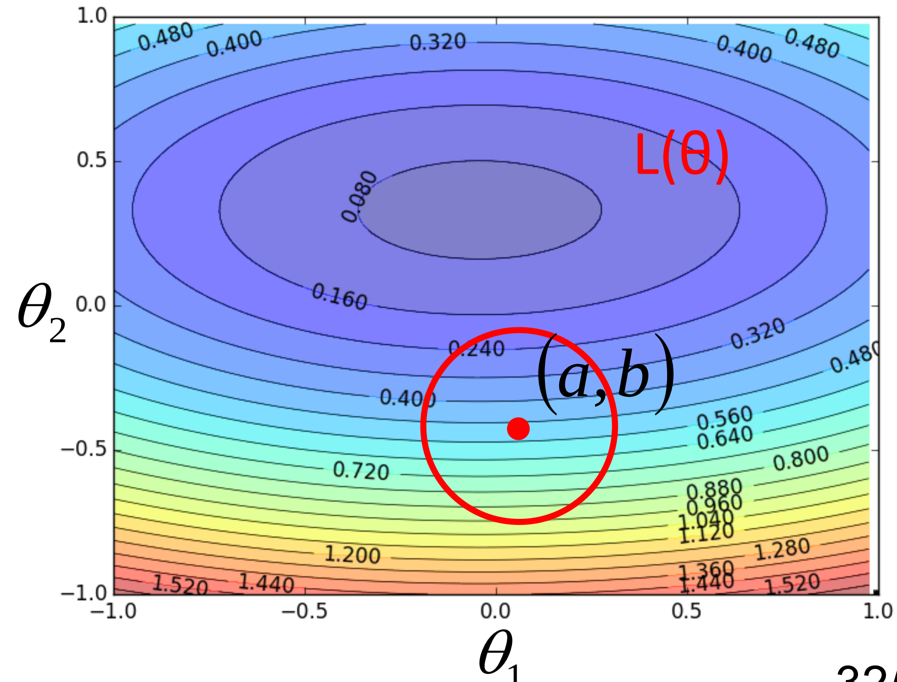
$$\mathrm{L}(\theta) \approx \mathrm{L}(a,b) + \frac{\partial \mathrm{L}(a,b)}{\partial \theta_1}(\theta_1 - a) + \frac{\partial \mathrm{L}(a,b)}{\partial \theta_2}(\theta_2 - b)$$

Constant

$$\boxed{\begin{gathered} s = \mathrm{L}(a,b) \\ u = \frac{\partial \mathrm{L}(a,b)}{\partial \theta_1}, v = \frac{\partial \mathrm{L}(a,b)}{\partial \theta_2} \end{gathered}}$$

$$\mathrm{L}(\theta)$$

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



Back to Formal Derivation

Based on Taylor Series:

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

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

constant

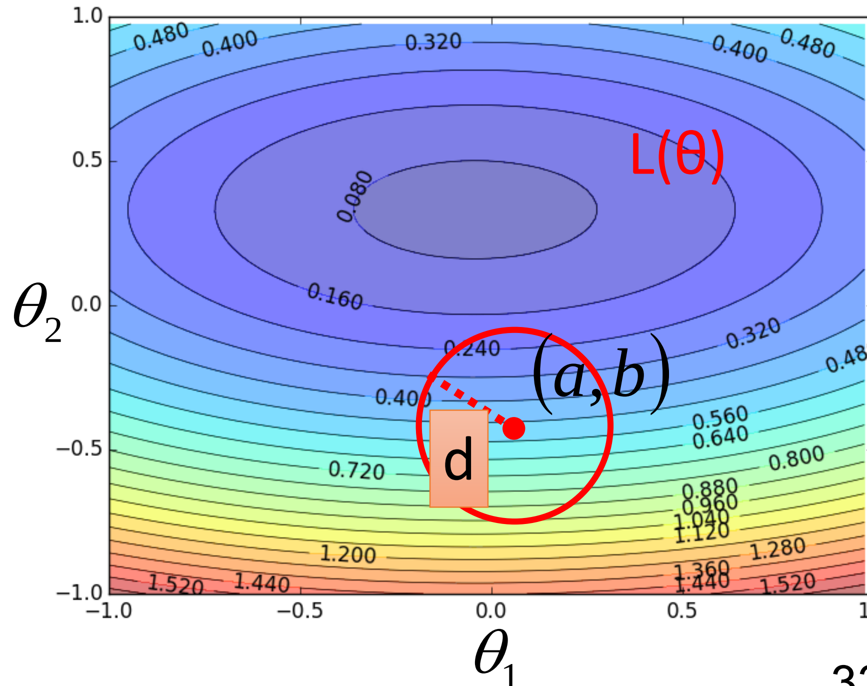
$$s = L(a, b)$$

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

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

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

Simple, right?



Gradient descent – two variables

Red Circle: (If the radius is small)

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

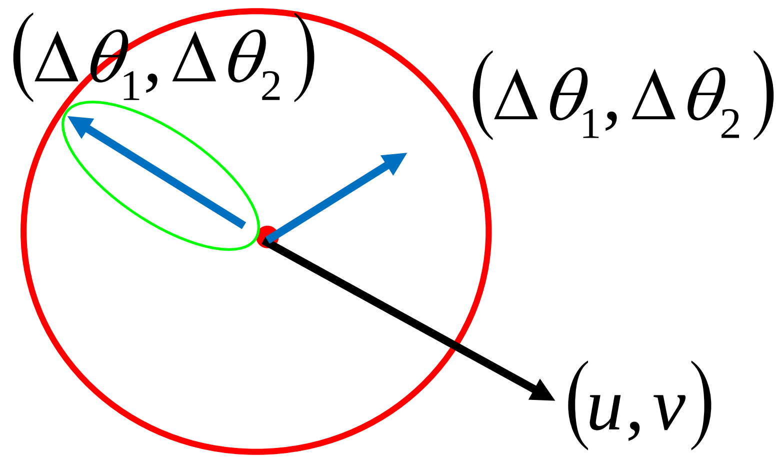
Inner product

$$= \begin{bmatrix} u \\ v \end{bmatrix} \cdot \begin{bmatrix} \Delta\theta_1 \\ \Delta\theta_2 \end{bmatrix} \Rightarrow$$

To find the smallest inner product, we need to find ($\Delta\theta_1$, $\Delta\theta_2$), which is longest and opposite direction of (u, v).

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)

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

Back to Formal Derivation

Based on Taylor Series:

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

constant

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

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

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

The assumption is the learning rate is very small.

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.

The computation cost is very large for deep learning.

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

Because in general, we will stop the update until the gradient is very small instead of equal to 0.