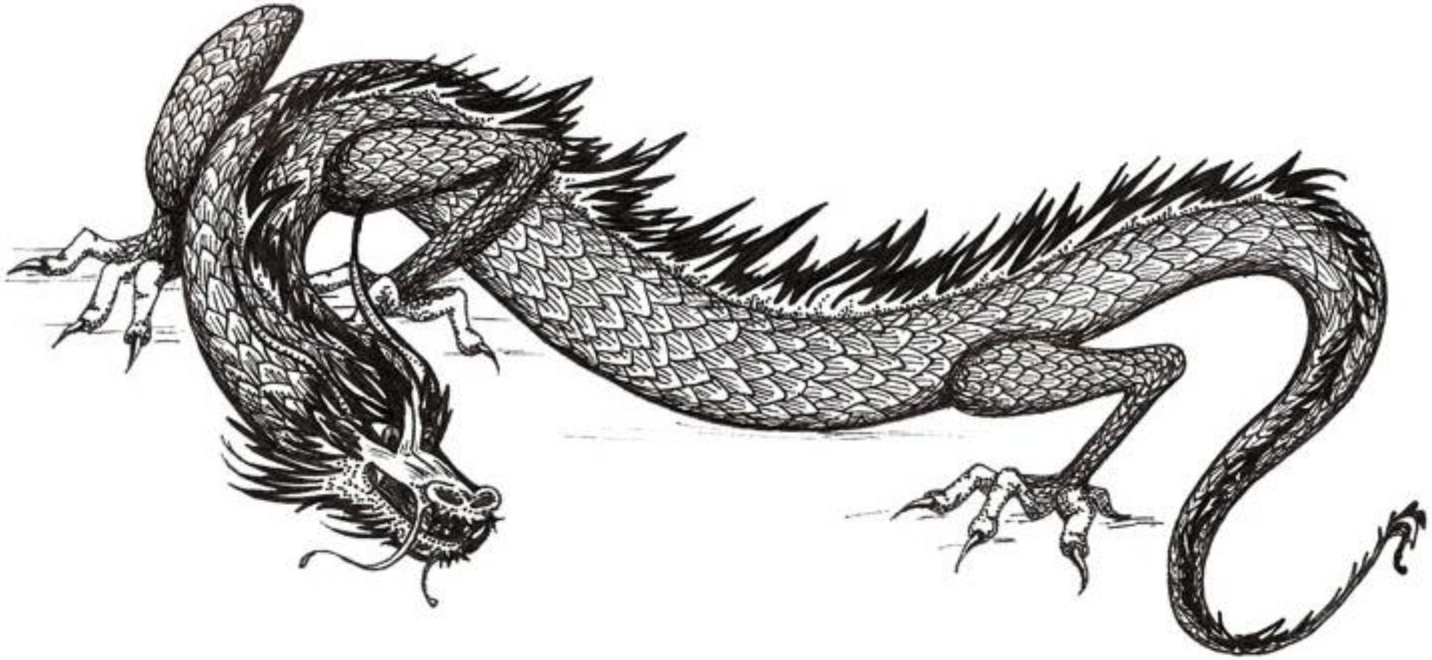


Gradient Descent

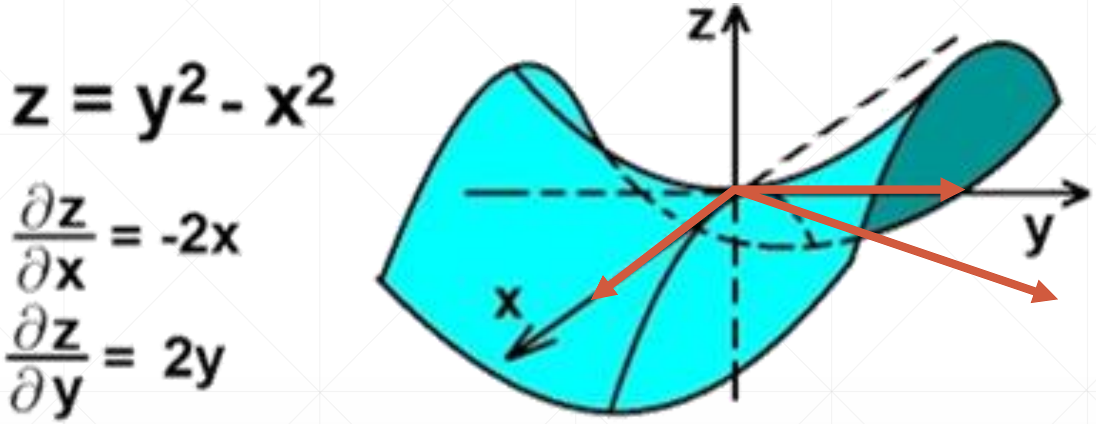
主讲：龙良曲

Outline

- What’s Gradient
- What does it mean
- How to Search
- AutoGrad

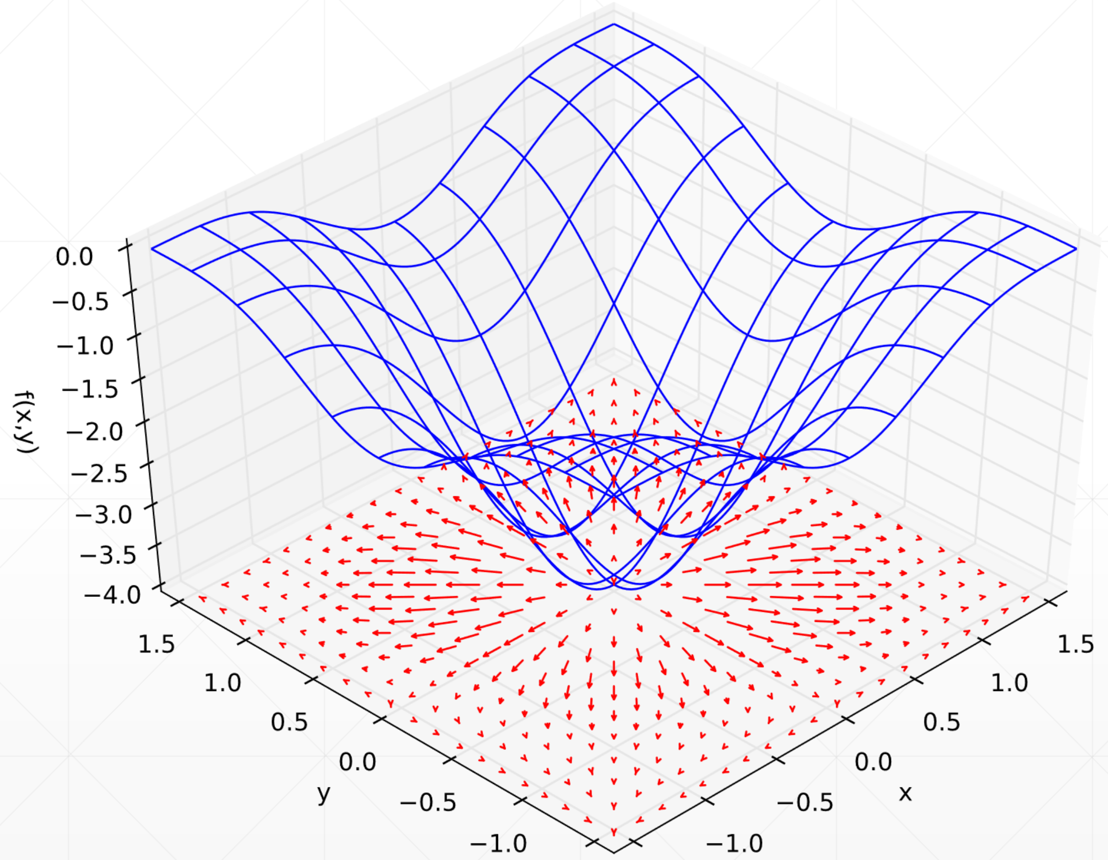
What's Gradient?

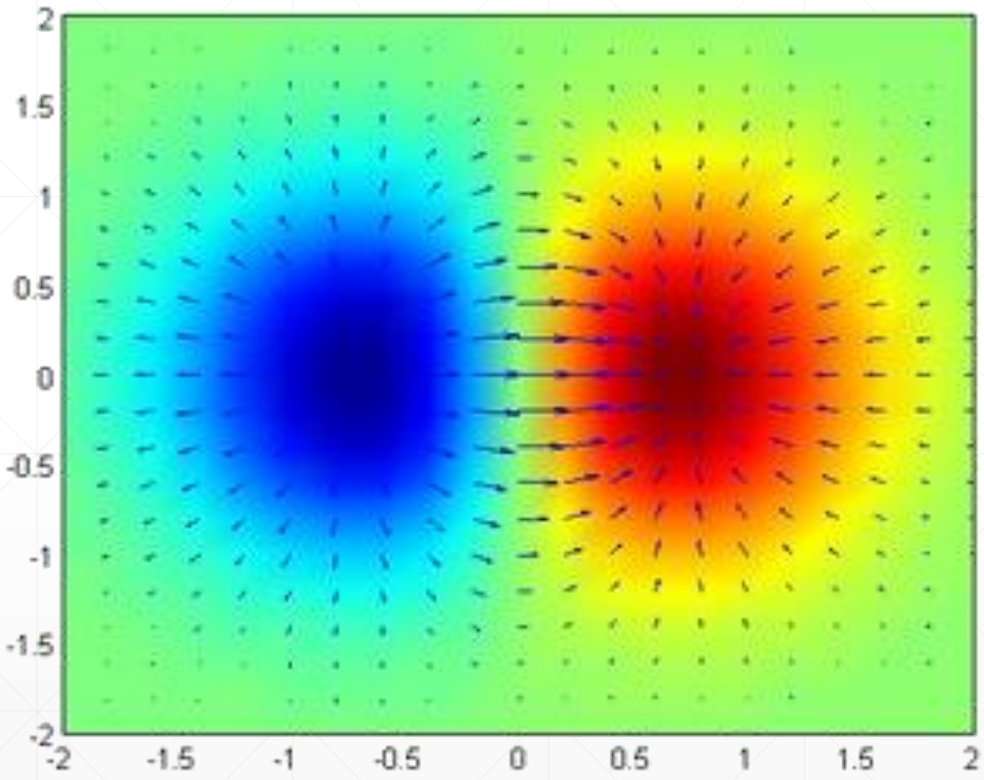
- 导数, derivative
- 偏微分, partial derivative
- 梯度, gradient



$$\nabla f = \left(\frac{\partial f}{\partial x_1}; \frac{\partial f}{\partial x_2}; \ldots; \frac{\partial f}{\partial x_n} \right)$$

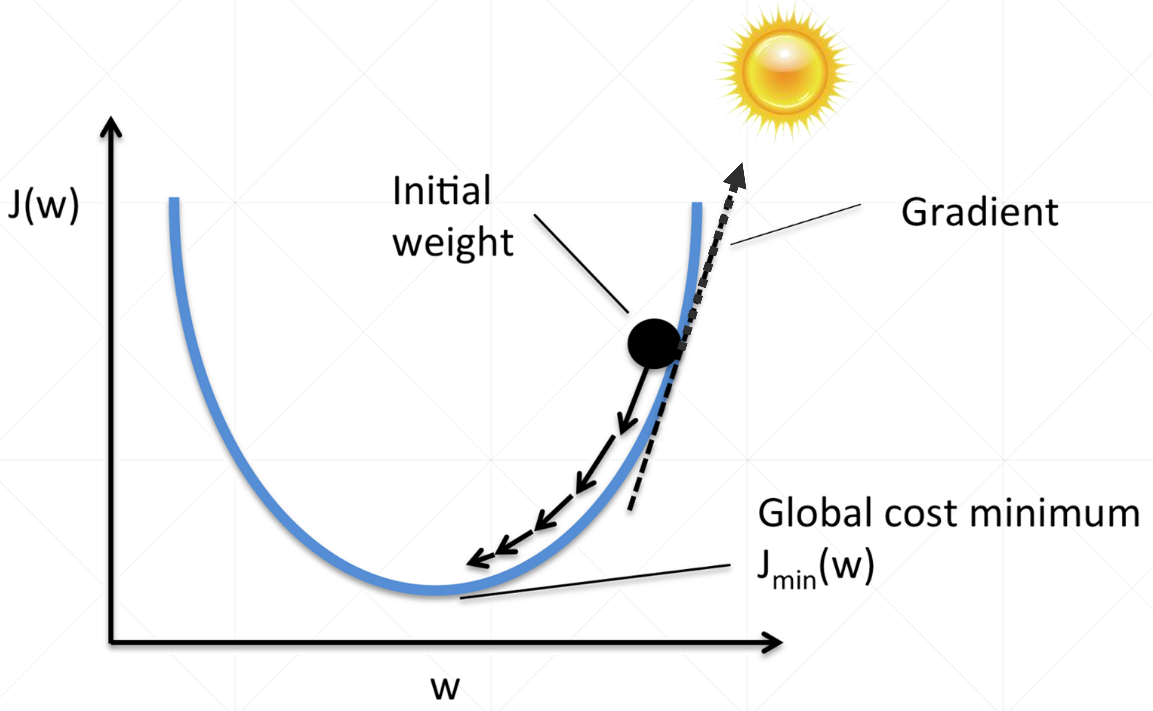
What does it mean?





How to search?

- $\nabla f(\theta) \rightarrow larger\ value$



- Search for minima:
  - $lr\ \alpha\ \eta$

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)\,.$$

For instance

$$\theta_{t+1} = \theta_t - \alpha_t \nabla f(\theta_t)\,.$$

Function:

$$J(\theta_1, \theta_2) = {\theta_1}^2 + {\theta_2}^2$$

Objective:

$$\min_{\theta_1, \theta_2} J(\theta_1, \theta_2)$$

Update rules:

$$\theta_1 := \theta_1 - \alpha \frac{d}{d\theta_1} J(\theta_1, \theta_2)$$

$$\theta_2 := \theta_2 - \alpha \frac{d}{d\theta_2} J(\theta_1, \theta_2)$$

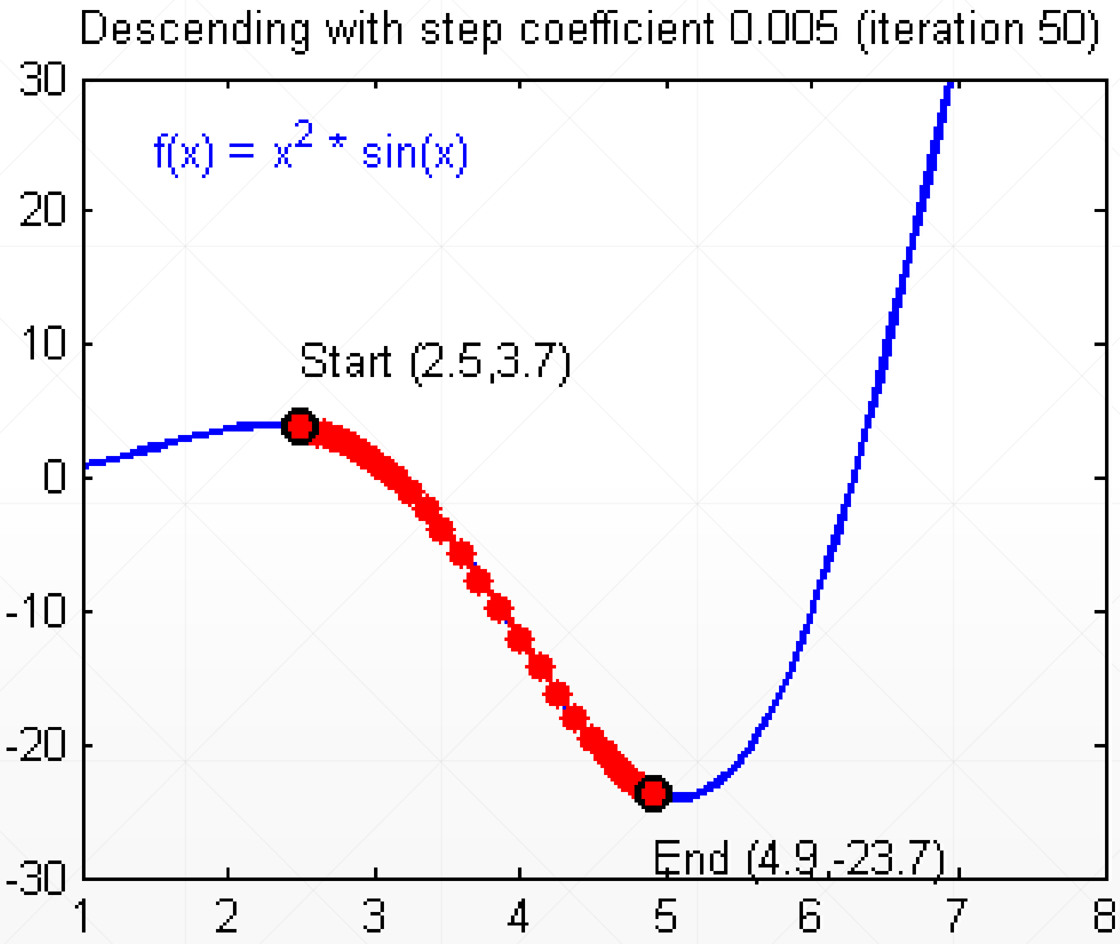
Derivatives:

$$\frac{d}{d\theta_1} J(\theta_1, \theta_2) = \frac{d}{d\theta_1} {\theta_1}^2 + \frac{d}{d\theta_1} {\theta_2}^2 = 2\theta_1$$

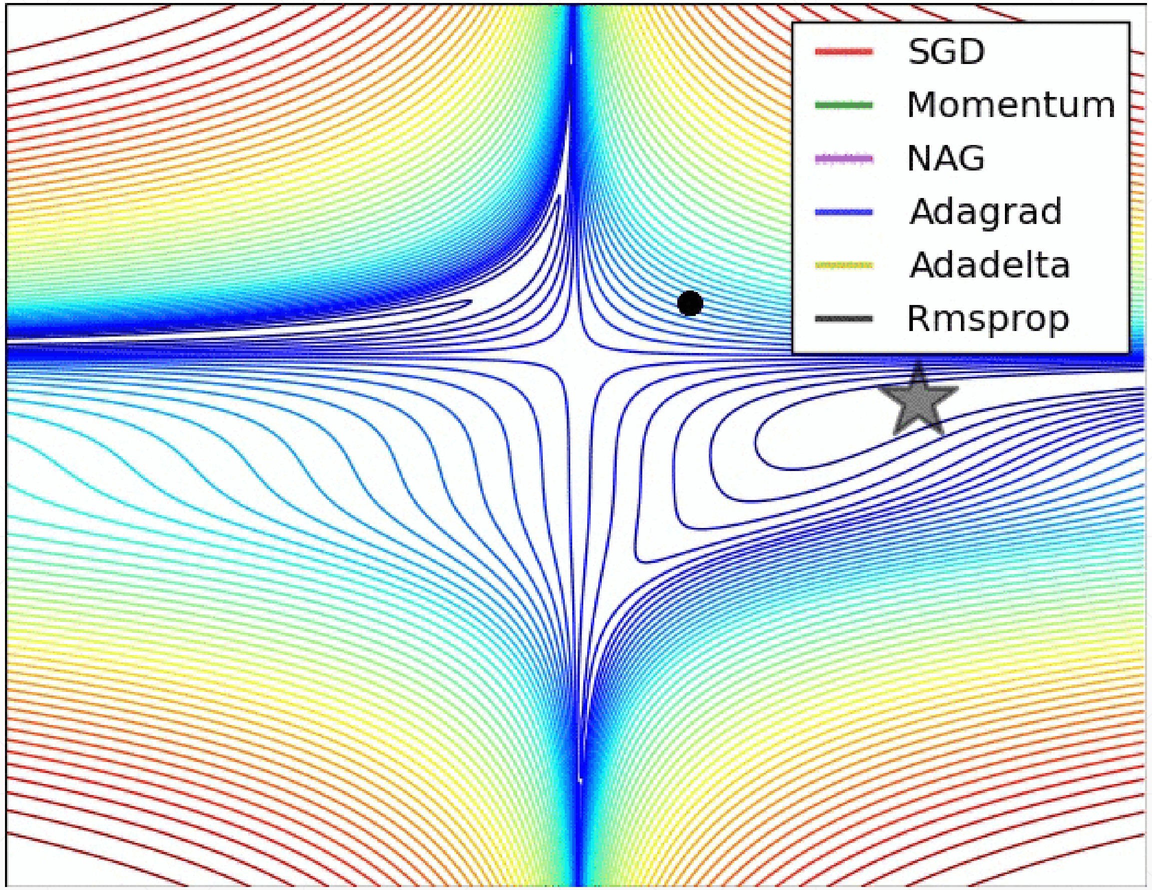
$$\frac{d}{d\theta_2} J(\theta_1, \theta_2) = \frac{d}{d\theta_2} {\theta_1}^2 + \frac{d}{d\theta_2} {\theta_2}^2 = 2\theta_2$$

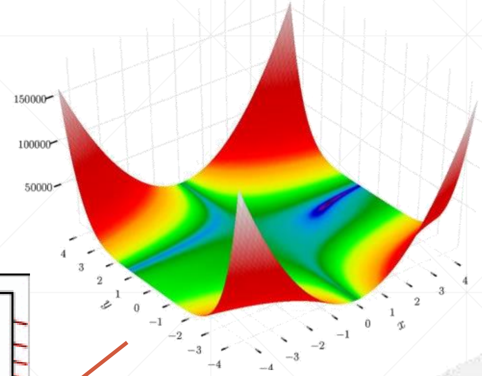
http://mccormickml.com/2014/03/04/gradient-descent-derivation/

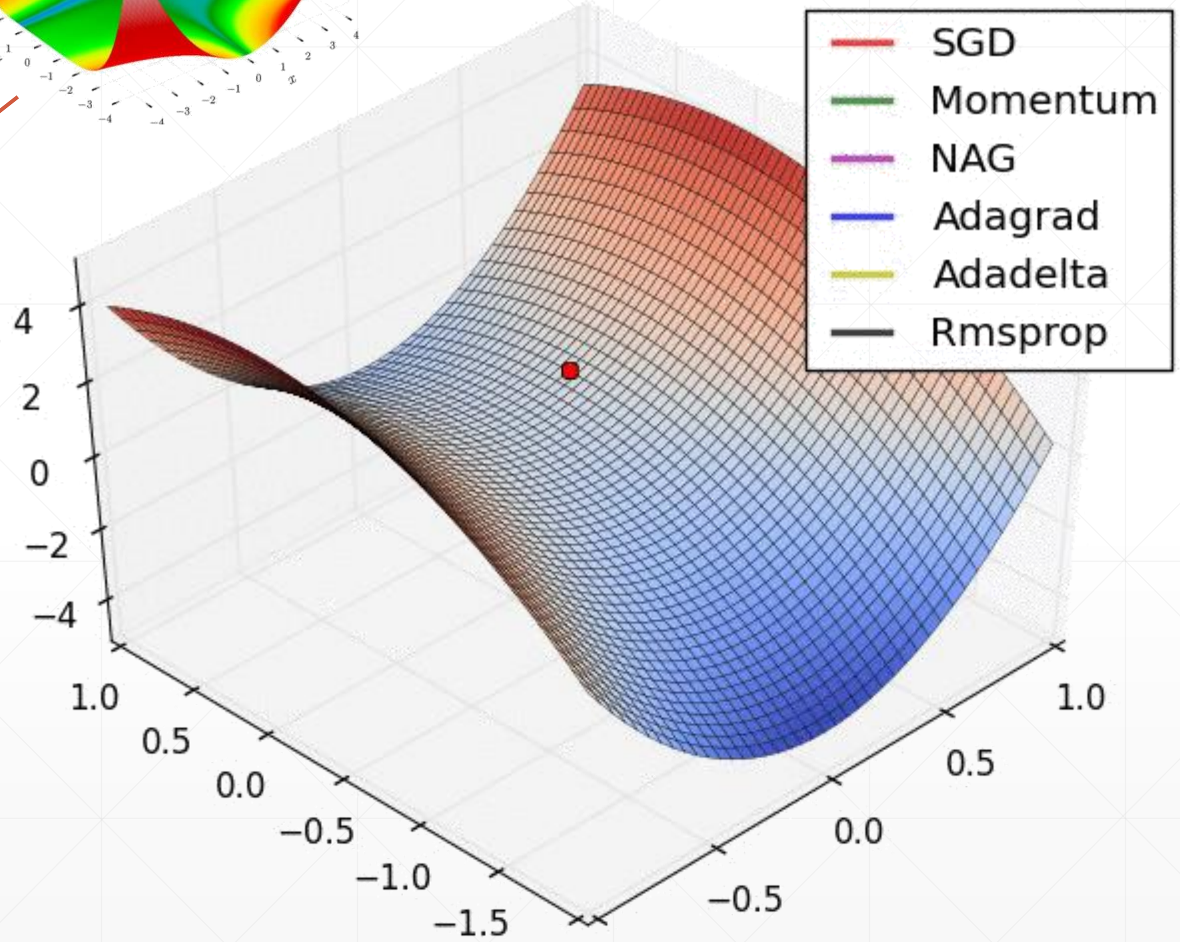
Learning Process-1



Learning Process-2







http://ruder.io/optimizing-gradient-descent/

AutoGrad

- With Tf.GradientTape() as tape:
  - Build computation graph
  - $loss = f_\theta(x)$

- [w_grad] = tape.gradient(loss, [w])

GradientTape

```
In [3]: w=tf.constant(1.)
In [4]: x=tf.constant(2.)
In [5]: y=x*w

In [8]: with tf.GradientTape() as tape:
   ...:     tape.watch([w])
   ...:     y2=x*w
In [11]: grad1=tape.gradient(y,[w])
Out[12]: [None]

In [18]: with tf.GradientTape() as tape:
    ...:     tape.watch([w])
    ...:     y2=x*w
In [19]: grad2=tape.gradient(y2,[w])
Out[16]: [<tf.Tensor: id=8, shape=(), dtype=float32, numpy=2.0>]
```

Persistent GradientTape

```
In [3]: w=tf.constant(1.)
In [4]: x=tf.constant(2.)
In [5]: y=x*w

In [18]: with tf.GradientTape() as tape:
    ...:     tape.watch([w])
    ...:     y2=x*w
In [19]: grad2=tape.gradient(y2,[w])
Out[16]: [<tf.Tensor: id=8, shape=(), dtype=float32, numpy=2.0>]

In [19]: grad2=tape.gradient(y2,[w])
RuntimeError: GradientTape.gradient can only be called once on non-persistent
tapes.

In [18]: with tf.GradientTape(persistent=True) as tape:
    ...:     tape.watch([w])
      ...
```

2^{nd}-order

- $y = xw + b$

- $\frac{\partial y}{\partial w} = x$
- $\frac{\partial^2 y}{\partial w^2} = \frac{\partial y'}{\partial w} = \frac{\partial x}{\partial w} = None$

2nd-order

```
with tf.GradientTape() as t1:

  with tf.GradientTape() as t2:
    y = x * w + b
  dy_dw, dy_db = t2.gradient(y, [w, b])

d2y_dw2 = t1.gradient(dy_dw, w)
```

JUST
DO
IT.

下一课时

选看：**反向传播算法推导**

必看：优化方法

Thank You.