



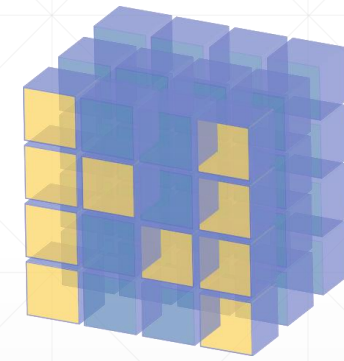
# 回归问题实战

---

主讲人：龙良曲

## Find $w', b'$

- $loss = \sum_i (w * x_i + b - y_i)^2$
- $w' = w - lr * \frac{\partial loss}{\partial w}$
- $b' = b - lr * \frac{\partial loss}{\partial b}$
- $w' * x + b' \rightarrow y$



NumPy

---

# Step1. Compute Loss

- $loss = \sum_i (w * x_i + b - y_i)^2$

- $w' = w - lr * \frac{\partial loss}{\partial w}$

- $b' = b - lr * \frac{\partial loss}{\partial b}$

- $w' * x + b' \rightarrow y$

---

# Step1. Compute Loss

```
● ● ●  
  
# y = wx + b  
def compute_error_for_line_given_points(b, w, points):  
    totalError = 0  
    for i in range(0, len(points)):  
        x = points[i, 0]  
        y = points[i, 1]  
        # computer mean-squared-error  
        totalError += (y - (w * x + b)) ** 2  
    # average loss for each point  
    return totalError / float(len(points))
```

---

## Step2.Compute Gradient and update

- $loss = \sum_i (w * x_i + b - y_i)^2$
  - $w' = w - lr * \frac{\partial loss}{\partial w}$
  - $b' = b - lr * \frac{\partial loss}{\partial b}$
-

## Step2.Compute Gradient and update



```
def step_gradient(b_current, w_current, points, learningRate):
    b_gradient = 0
    w_gradient = 0
    N = float(len(points))
    for i in range(0, len(points)):
        x = points[i, 0]
        y = points[i, 1]
        # grad_b = 2(wx+b-y)
        b_gradient += (2/N) * ((w_current * x + b_current) - y)
        # grad_w = 2(wx+b-y)*x
        w_gradient += (2/N) * x * ((w_current * x + b_current) -
y) # update w'
    new_b = b_current - (learningRate * b_gradient)
    new_w = w_current - (learningRate * w_gradient)
    return [new_b, new_w]
```

## Step3.Set $w=w'$ and loop

- $loss = \sum_i (w * x_i + b - y_i)^2$
  - $w' = w - lr * \frac{\partial loss}{\partial w}$
  - $b' = b - lr * \frac{\partial loss}{\partial b}$
  - $w \leftarrow w'$
  - $b \leftarrow b'$
-

## Step3.Set $w=w'$ and loop

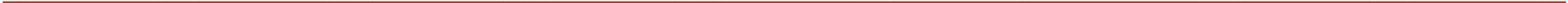


```
def gradient_descent_runner(points, starting_b, starting_w, learning_rate, num_iterations):  
    b = starting_b  
    w = starting_w  
    # update for several times  
    for i in range(num_iterations):  
        b, w = step_gradient(b, w, np.array(points), learning_rate)  
    return [b, w]
```

---



**JUST  
DO  
IT.**

The text "JUST DO IT." is rendered in a bold, black, sans-serif font. The letters are filled with a dark camouflage pattern. The text is surrounded by a dynamic splash of water, with droplets and ripples extending outwards, particularly around the "DO" and "IT." parts. The entire graphic is centered on a light gray background with a subtle, repeating diamond-shaped grid pattern.

# 下一课时

---

Discrete Problem  
Prediction

**Thank You.**

---