线性模型
1. DataSet 准备数据集
2. Model 设计模型
3. Training 训练模型
4. Inferring 推理结果

训练集训练结果误差很小
有可能噪声也学进去了
↓
希望模型有比较好的泛化能力

训练	测试
(x,y)	(x)

x,t 是 D(x,y) 联合分布

数据集是否满足真实的分布吗？

PyTorch Tutorial

训练后 对于没训练过的
图像也能识别正确

02. Linear Model

训练	开发dev	测试

用于评估模型性能，若性能好，
模型训练所有训练集数据

训练时  预测时 

差异较大 < 真实业务场景的图像分布
训练集中的分布

降低差异

若要保证模型上线，真正投入应用后有较高的性能

- Suppose that students would get ***y*** points in final exam, if they spent ***x*** hours in paper *PyTorch Tutorial*.

x (hours)	y (points)
1	2
2	4
3	6
4	?

Training

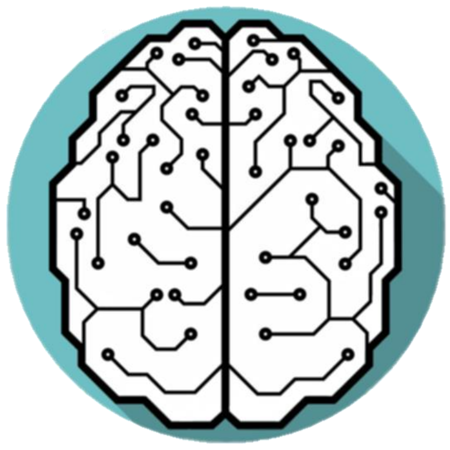
Inferring / Testing

- The question is what would be the grade if I study 4 hours?

Machine Learning

4 hours

Input - x

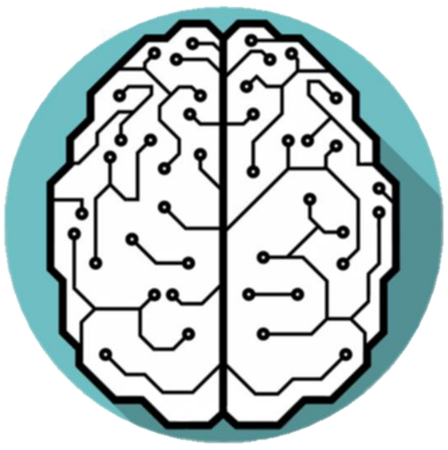


? points

Prediction - y

Dataset

x (hours)	y (points)
1	2
2	4
3	6
4	?

4 hours →  → **? points**

Prediction

x (hours)	y (points)
1	2
2	4
3	6
4	?

训练集：已知x,y→监督学习

Training Set

Test Set

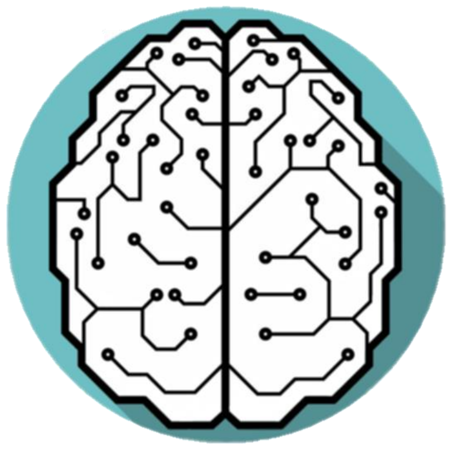
测试集：判定模型的泛化能力

Supervised Learning

Model design

- What would be the best model for the data?
- Linear model?

x (hours)	y (points)
1	2
2	4
3	6
4	?



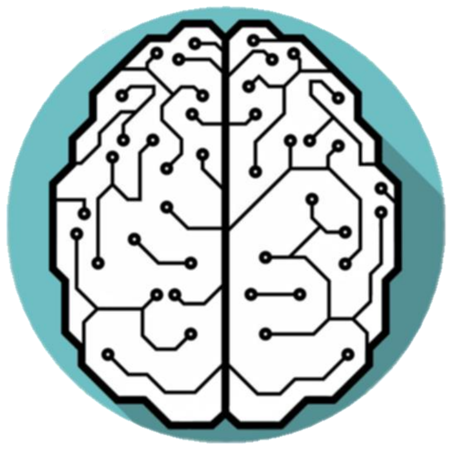
Linear Model

$$\hat{y} = x * \underline{\omega} + \underline{b}$$

Model design

- What would be the best model for the data?
- Linear model?

x (hours)	y (points)
1	2
2	4
3	6
4	?



Linear Model

$$\hat{y} = x * \omega \cancel{+b}$$

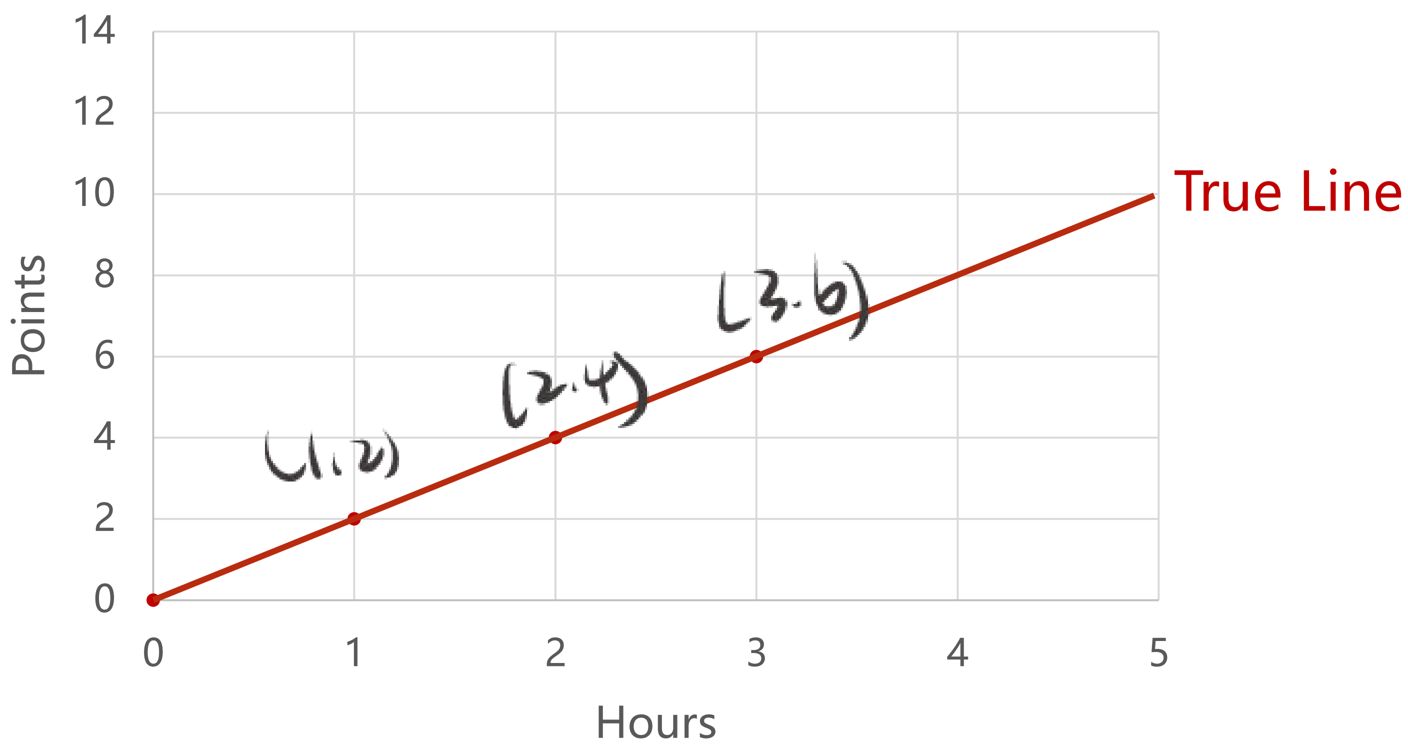
To simplify the model

预测值 y_hat

Linear Model

$$\hat{y} = x * \omega$$

x (hours)	y (points)
1	2
2	4
3	6



Linear Model

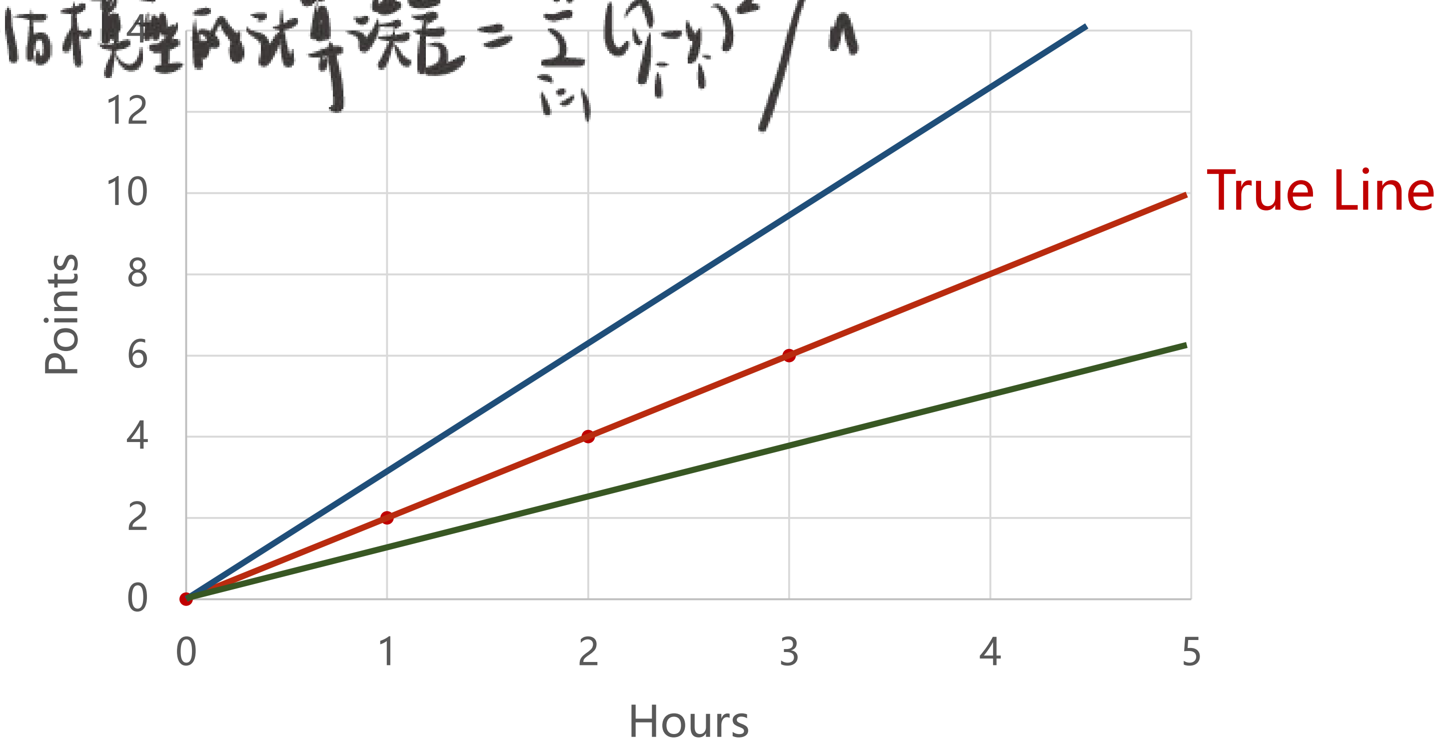
$$\hat{y} = x * \omega$$

x (hours)	y (points)
1	2
2	4
3	6

一开始取权重w = 随机数 random $\begin{cases} w\text{大了} \\ w\text{小了} \end{cases}$

The machine starts with **a random guess**, ω = random value

评估模型的计算误差 $= \sum_{i=1}^{n} (\hat{y}_i - y_i)^2 / n$



Compute Loss

Training Loss (Error)

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

x (Hours)	y (Points)	y_predict (w=3) $=xw$	Loss (w=3) $=(xw-y)^2$
1	2	3	1
2	4	6	4
3	6	9	9
			mean = 14/3

Compute Loss

Training Loss (Error)

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

x (Hours)	y (Points)	y_predict (w=4)	Loss (w=4)
1	2	4	4
2	4	8	16
3	6	12	36
			mean = 56/3

Compute Loss

Training Loss (Error)

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

x (Hours)	y (Points)	y_predict (w=0)	Loss (w=0)
1	2	0	4
2	4	0	16
3	6	0	36
			mean = 56/3

Compute Loss

Training Loss (Error)

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

x (Hours)	y (Points)	y_predict (w=1)	Loss (w=1)
1	2	1	1
2	4	2	4
3	6	3	9
			mean = 14/3

Compute Loss

Training Loss (Error)

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

x (Hours)	y (Points)	y_predict (w=2)	Loss (w=2)
1	2	2	0
2	4	4	0
3	6	6	0
			mean = 0

Loss function & Cost function

Training Loss (Error)

一个样本

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

Mean Square Error

所有样本

平均平方误差MSE

$$cost = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

Compute Cost

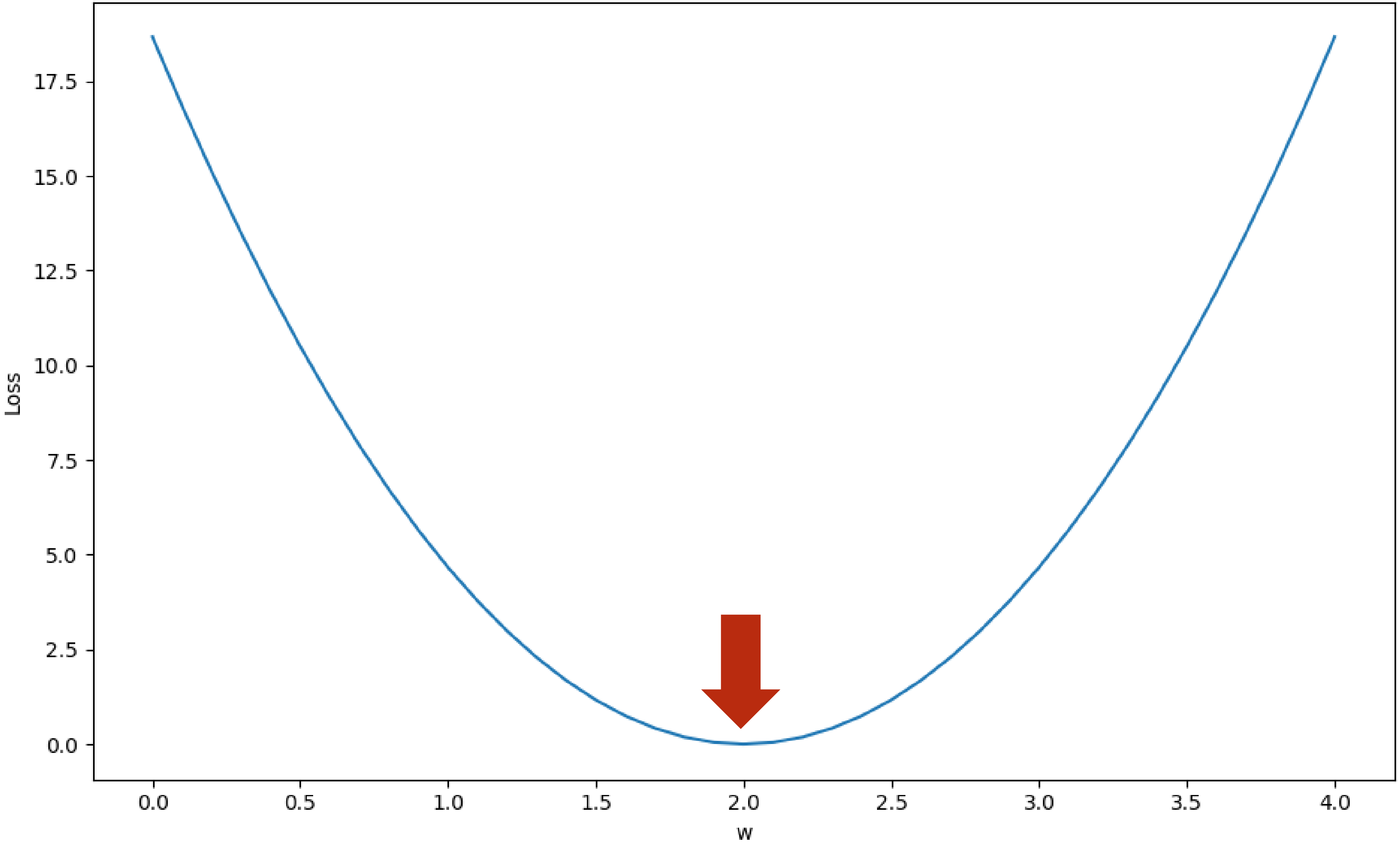
Mean Square Error

$$cost = \frac{1}{N}\sum_{n=1}^{N}(\hat{y}_n - y_n)^2$$

x (Hours)	Loss (w=0)	Loss (w=1)	Loss (w=2)	Loss (w=3)	Loss (w=4)
1	4	1	0	1	4
2	16	4	0	4	16
3	36	9	0	9	36
MSE	18.7	4.7	0	4.7	18.7

Linear Regression

It can be found that when $\omega = 2$, the cost will be minimal.



How to draw the graph

绘图包

```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```


```python
import numpy as np
import matplotlib.pyplot as plt
```


Import necessary library to draw the graph.

How to draw the graph

```
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

x y 训练集

```
x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]
```

Prepare the train set.

准备数据训练集

```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

前馈函数模型 $\hat{y} = x \cdot w$

```python
def forward(x):
    return x * w
```

Define the model:

Linear Model

$$\hat{y} = x * \omega$$

```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

损失函数

$loss = (\hat{y}-y)^2$

```python
def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)
```

Define the loss function:

Loss Function

$$loss = (\hat{y} - y)^2 = (x * \omega - y)^2$$

```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

```python
w_list = []
mse_list = []
```

权重

权重损失值

List *w_list* save the weights ω.

List *mse_list* save the cost values of each ω.

暂存到列表中 存放可能的权重值 有个范围 [0.0, 0.1, 0.2 ... 4.0]

How to draw the graph

```
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

```
for w in np.arange(0.0, 4.1, 0.1):
```

开始 结束 间隔

Compute cost value at [0.0, 0.1, 0.2, … , 4.0]

很多个可能的权重值 有个范围 [0.0, 0.1, 0.2 … 4.0]

How to draw the graph

(x=1, y=2) (x=2, y=4) (x=3, y=6)

```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

列表数据打包成元组，进行遍历

预测值 $\hat{y} = x \cdot w$

单点损失值 $loss = (\hat{y} - y)^2$

累加各点损失值 l_sum

```python
for x_val, y_val in zip(x_data, y_data):
    y_pred_val = forward(x_val)
    loss_val = loss(x_val, y_val)
    l_sum += loss_val
    print('\t', x_val, y_val, y_pred_val, loss_val)
```

For each sample in train set, the loss function values were computed.

ATTENTION:
Value of cost function is the sum of loss function.

计算三点平均损失值 l_sum/3 → 每个权重w对应一个权重损失值 MSE = l_sum/3

How to draw the graph

```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

```python
w_list.append(w)
mse_list.append(l_sum / 3)
```

Save ω and correspondence **MSE**.

保存w、MSE至列表中

画出w、MSE的一一对应曲线

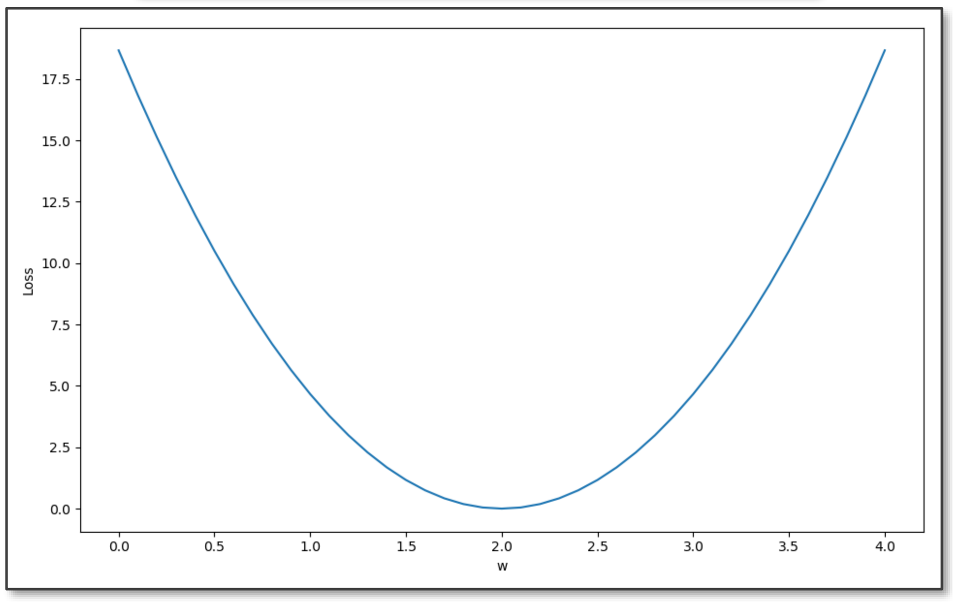
```python
import numpy as np
import matplotlib.pyplot as plt

x_data = [1.0, 2.0, 3.0]
y_data = [2.0, 4.0, 6.0]

def forward(x):
    return x * w

def loss(x, y):
    y_pred = forward(x)
    return (y_pred - y) * (y_pred - y)

w_list = []
mse_list = []
for w in np.arange(0.0, 4.1, 0.1):
    print('w=', w)
    l_sum = 0
    for x_val, y_val in zip(x_data, y_data):
        y_pred_val = forward(x_val)
        loss_val = loss(x_val, y_val)
        l_sum += loss_val
        print('\t', x_val, y_val, y_pred_val, loss_val)
    print('MSE=', l_sum / 3)
    w_list.append(w)
    mse_list.append(l_sum / 3)
```

Part of result

```
w= 0.0
	1.00	2.00	0.00	4.00
	2.00	4.00	0.00	16.00
	3.00	6.00	0.00	36.00
MSE= 18.666666666666668
w= 0.1
	1.00	2.00	0.10	3.61
	2.00	4.00	0.20	14.44
	3.00	6.00	0.30	32.49
MSE= 16.846666666666668
w= 0.2
	1.00	2.00	0.20	3.24
	2.00	4.00	0.40	12.96
	3.00	6.00	0.60	29.16
MSE= 15.120000000000003
w= 0.30000000000000004
	1.00	2.00	0.30	2.89
	2.00	4.00	0.60	11.56
	3.00	6.00	0.90	26.01
MSE= 13.486666666666665
w= 0.4
	1.00	2.00	0.40	2.56
	2.00	4.00	0.80	10.24
	3.00	6.00	1.20	23.04
MSE= 11.946666666666667
w= 0.5
	1.00	2.00	0.50	2.25
	2.00	4.00	1.00	9.00
	3.00	6.00	1.50	20.25
MSE= 10.5
```

Draw the graph

```python
plt.plot(w_list, mse_list)
plt.ylabel('Loss')
plt.xlabel('w')
plt.show()
```



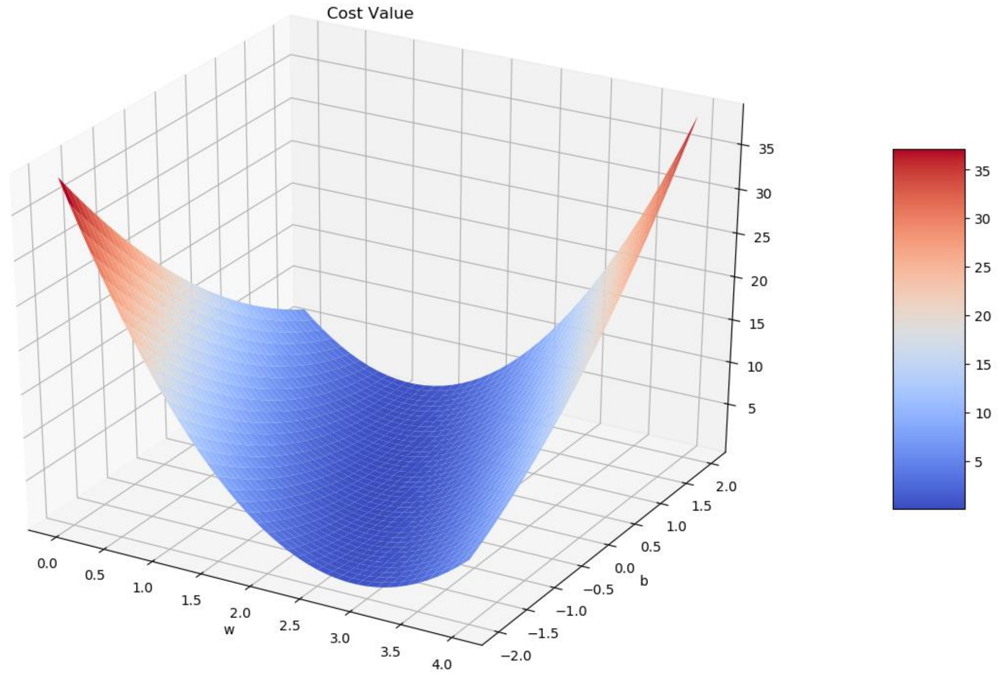
Exercise

- Try to use the model in right-side, and draw the cost graph.

Linear Model

$$\hat{y} = x * \omega + b$$

- Tips:
  - You can read the material of how to draw 3d graph. [link]
  - Function *np.meshgrid()* is very popular for drawing 3d graph, read the [docs] and utilize vectorization calculation.



PyTorch Tutorial

02. Linear Model