What is Kaggle Competition?

- Design assignments based on course content
- Each assignment will be a task related to machine learning
- You will be required to implement machine learning methods and apply the methods to data provided by the teaching assistant
- Based on your method, there will be a corresponding outcome (e.g., classification result)
- Your output results will be scored by the Kaggle platform

- For example:
  - Slides & Video

Regression

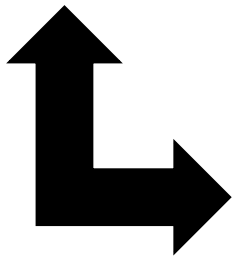
顏安孜

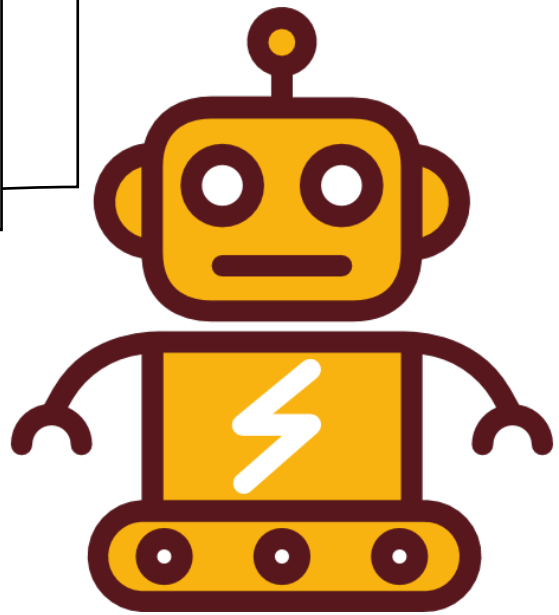
azyen@nycu.edu.tw

Some of the following slides are selected from the course material of Machine Learning by Prof. Hung-Yi Lee

What is Machine Learning?

Rules:
Please turn on the music ➔ ▶️
Please turn off the music ➔ ⏸️
…





Please don’t turn on the music.

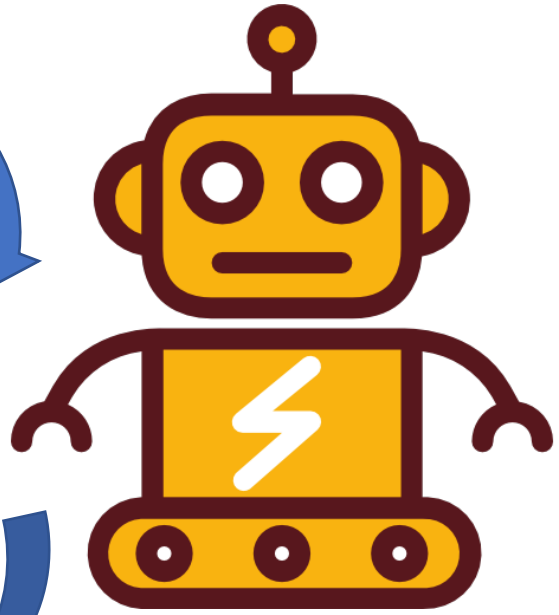
… ▶️

What is Machine Learning?

Learning from …

Machine Learning

機器學習



Input: Artificial Intelligence
Output:人工智慧

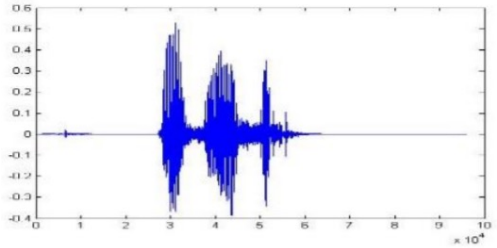
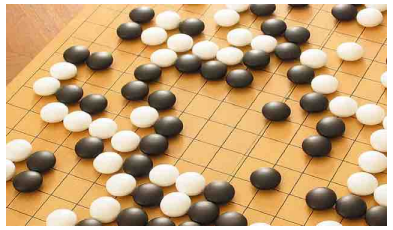
Input: Hello
Output:你好

Input: Good bye
Output: 再見

A large amount of training data

.
.
.

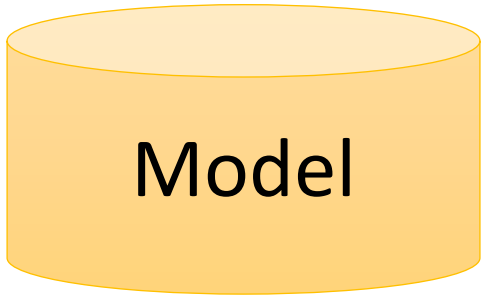
What is Machine Learning?

- Speech Recognition
  - f = () = “How are you”
- Image Recognition
  - f = () = “cat”
- Translation
  - f = (“Machine Learning”) = “機器學習”
- Playing Go
  - f = () = “5-5”

What is Model?

Input Output

- f = () = “cat”



A set of function
$f_1, f_2, \ldots$

- f_1 = () = “cat”
- f_1 = () = “dog”
- f_2 = () = “tiger”
- f_2 = () = “wolf”

Machine Learning Framework

Training

Model

A set of function

$f_1, f_2, \ldots$

• f_1 = ( ) = "cat"

• f_2 = ( ) = "tiger"

• f_1 = ( ) = "dog"

• f_2 = ( ) = "wolf"

Pick the "Best" Function

"f_1"

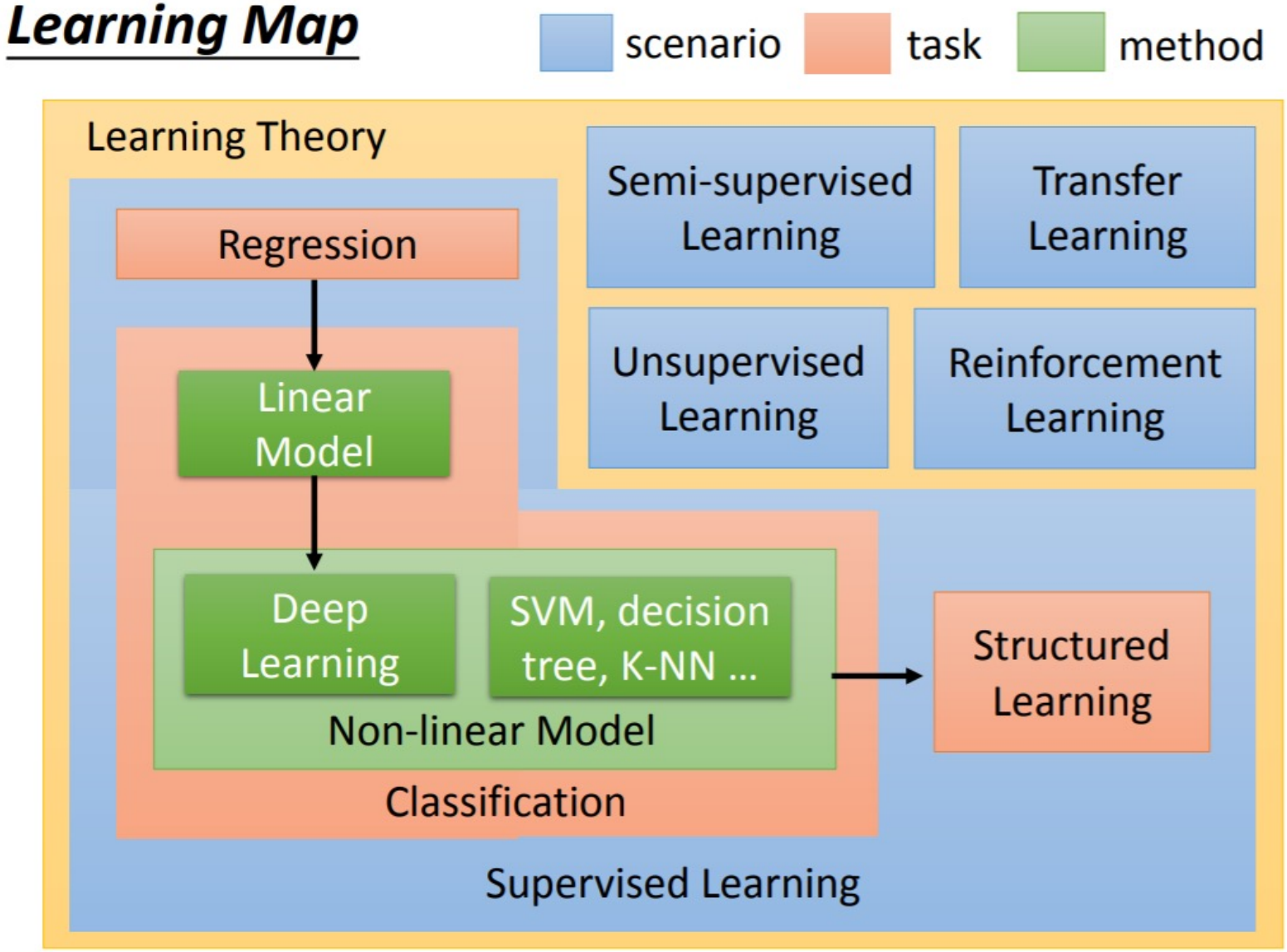
Training Data

Testing

"cat"

f^*

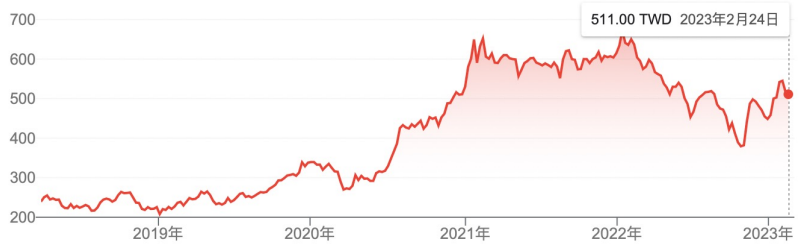
Three Steps of Machine Learning



- Reference: http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/introduction.pdf

Regression

- Stock Market Forecast

$f($ 

$) =$ closing price of 2330

- Self-driving Car

$f($  $) =$ steering angle

- Recommendation

$f($ User A Product B $) =$ the probability of buying Product B

Classification

- Binary Classification
  - Output: yes or no
- Multi-class Classification
  - Output: class1, class2, …, class*N*
- Multi-label Classification
  - Output: (class1, class2), (class1, class4, class5), …, $(\text{class3, class6})_m$
  - where *m* is the number of input

Regression

Application: Predicting PM2.5

- Predicting the PM2.5 for the next hour based on the current PM2.5

5/19 9:00 PM2.5

$$f(\ x_{\mathrm{PM2.5}}\ ) = \ y$$

5/19 10:00 PM2.5

Step 1: Model

$y = b + w \cdot x_{PM2.5}$

A set of function

Model

$f_1, f_2 \cdots$

w and b are parameters
(can be any value)

f_1: $y = 10.0 + 9.0 \cdot x_{PM2.5}$

f_2: $y = 9.8 + 9.2 \cdot x_{PM2.5}$

f_3: $y = -0.8 - 1.2 \cdot x_{PM2.5}$

…… infinite

5/19 9:00 PM2.5

$f($ 細懸浮微粒 $PM_{2.5}$ 11 小時濃度值(μg/m^3) $x) =$

5/19 10:00 PM2.5

細懸浮微粒 $PM_{2.5}$ 12 小時濃度值(μg/m^3) y

Linear model: $y = b + \sum w_i x_i$

x_i: $x_{PM2.5}, x_{co}, x_{so_2}, x_{o_3}$ …

feature

w_i: weight, b: bias

Step 2: Goodness of Function

$y = b + w \cdot x_{PM2.5}$

A set of function

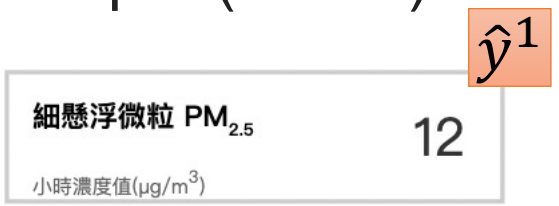
Model

$f_1, f_2 \cdots$

function input:

5/19 9:00 PM2.5

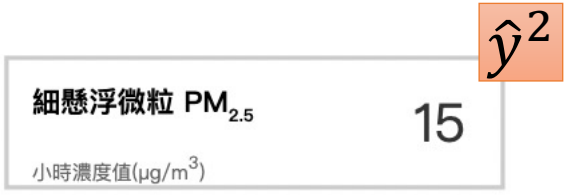
function output (scalar):



5/19 10:00 PM2.5

Training Data

5/18 12:00 PM2.5



5/18 13:00 PM2.5

Step 2: Goodness of Function

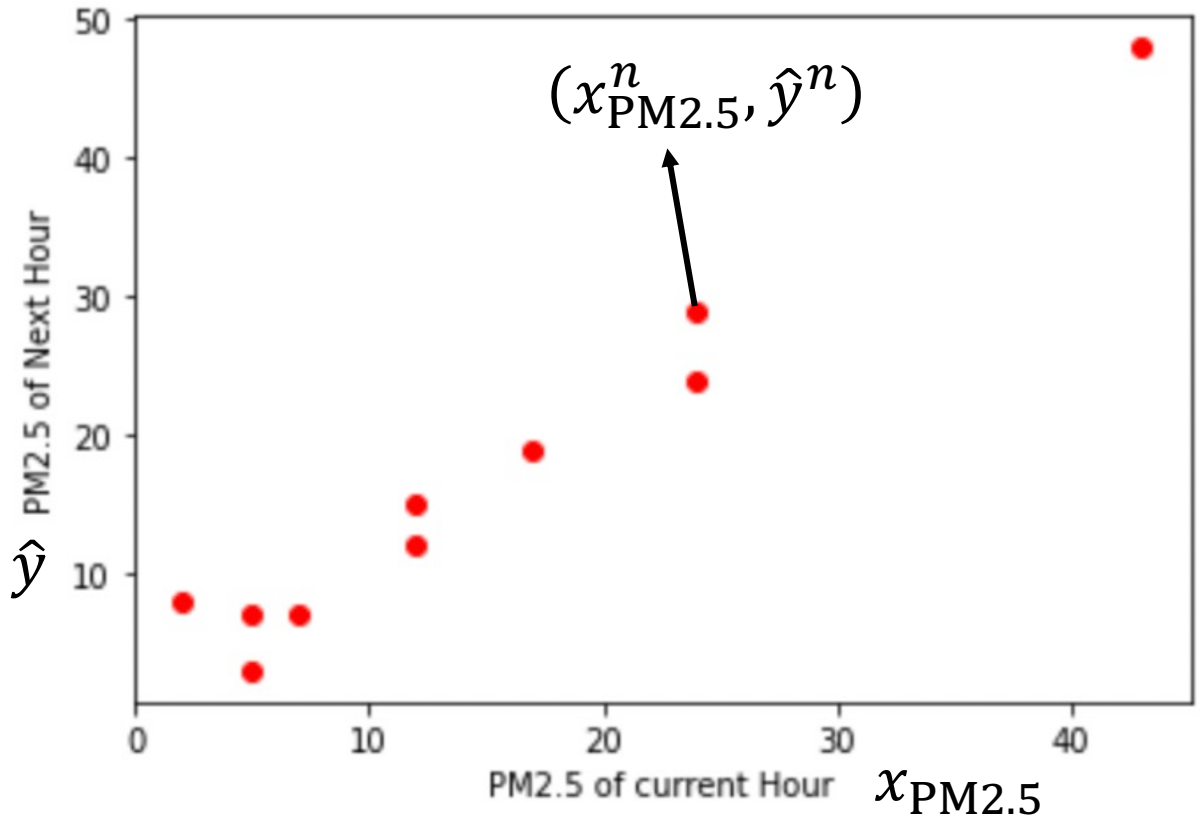
Training Data:
N PM2.5

$(x^1, \hat{y}^1)$

$(x^2, \hat{y}^2)$

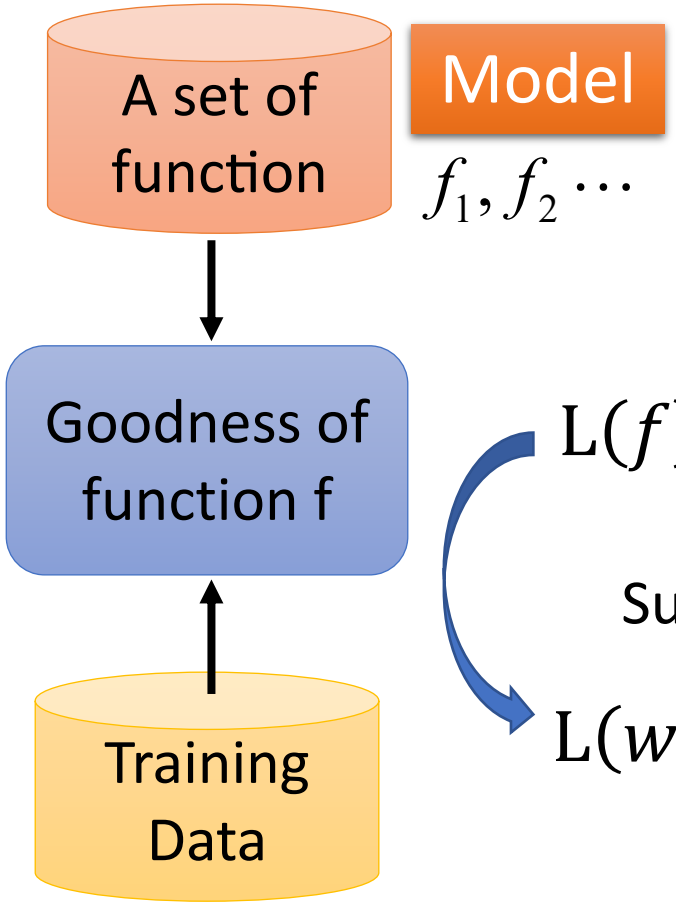
$\vdots$

$(x^N, \hat{y}^N)$



Step 2: Goodness of Function

$y = b + w \cdot x_{PM2.5}$



Loss function L:

Input: a function, output: how bad it is

Estimation error

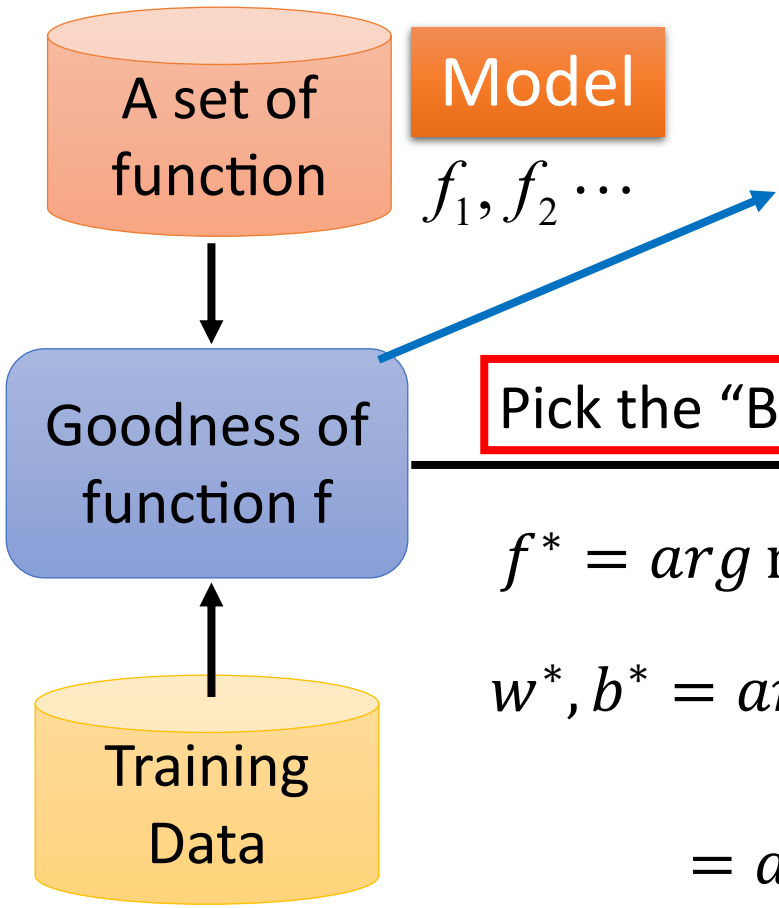
$$L(f) = \sum_{n=1}^{N} \left(\hat{y}^n - f\left(x_{PM2.5}^n\right)\right)^2$$

Sum over examples

Estimated y based on input function

$$L(w,b) = \sum_{n=1}^{N} \left(\hat{y}^n - \left(b + w \cdot x_{PM2.5}^n\right)\right)^2$$

Step 3: Best Function



$$L(w,b) = \sum_{n=1}^{N} \left(\hat{y}^n - \left(b + w \cdot x_{PM2.5}^n\right)\right)^2$$

Pick the "Best" Function

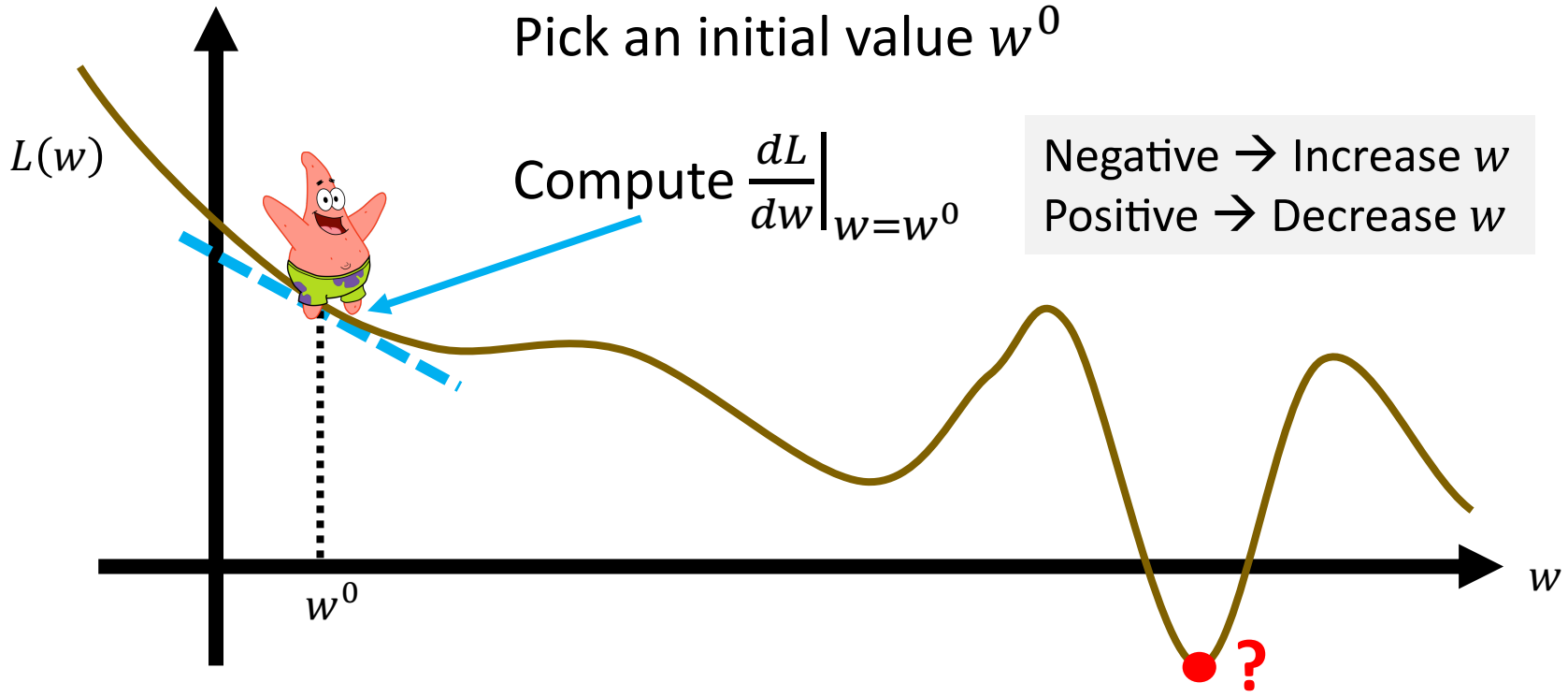
$$f^* = arg \min_f L(f)$$

$$w^*, b^* = arg \min_{w,b} L(w,b)$$

$$= arg \min_{w,b} \sum_{n=1}^{N} \left(\hat{y}^n - \left(b + w \cdot x_{\text{PM2.5}}^n\right)\right)^2$$

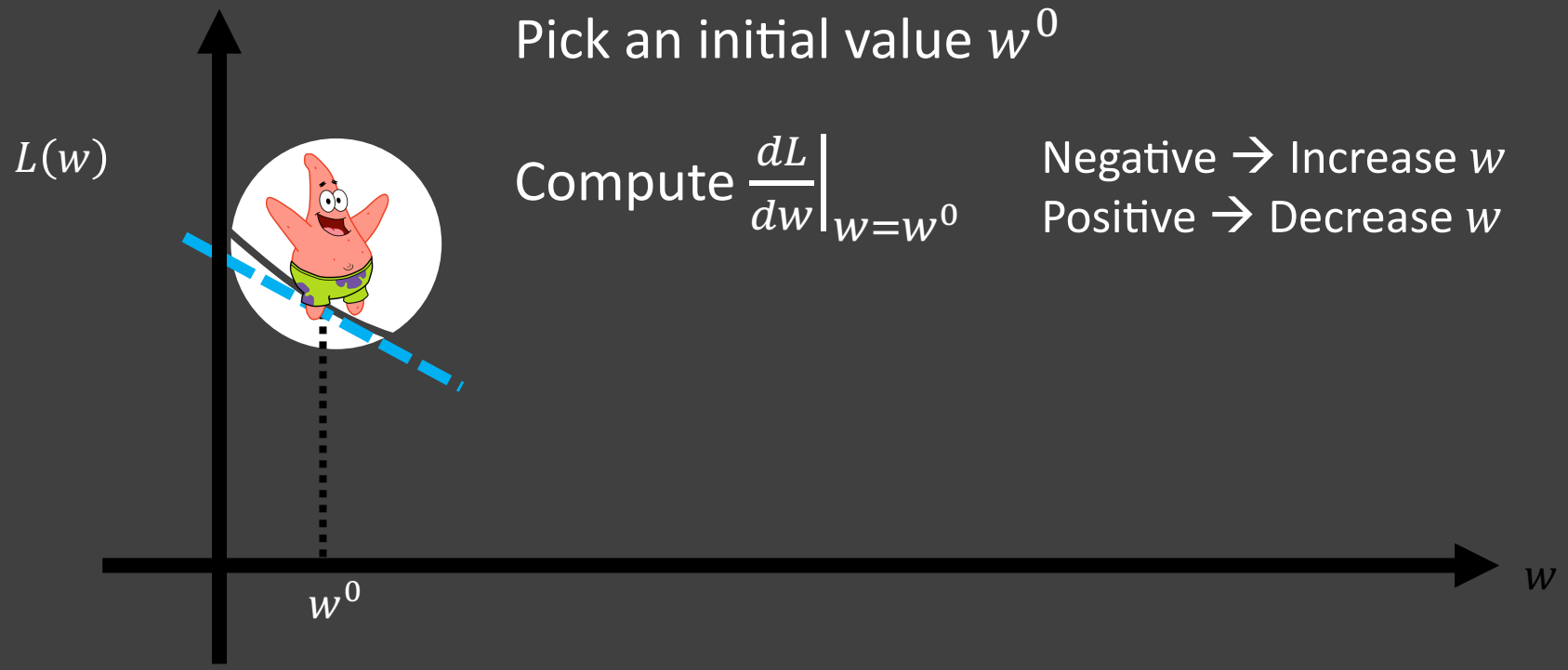
Gradient Descent

Step 3: Gradient Descent



Source of image: https://imgbin.com/png/ahH7zdnN/patrick-star-spongebob-squarepants-sandy-cheeks-starfish-png

Step 3: Gradient Descent



Pick an initial value w^0

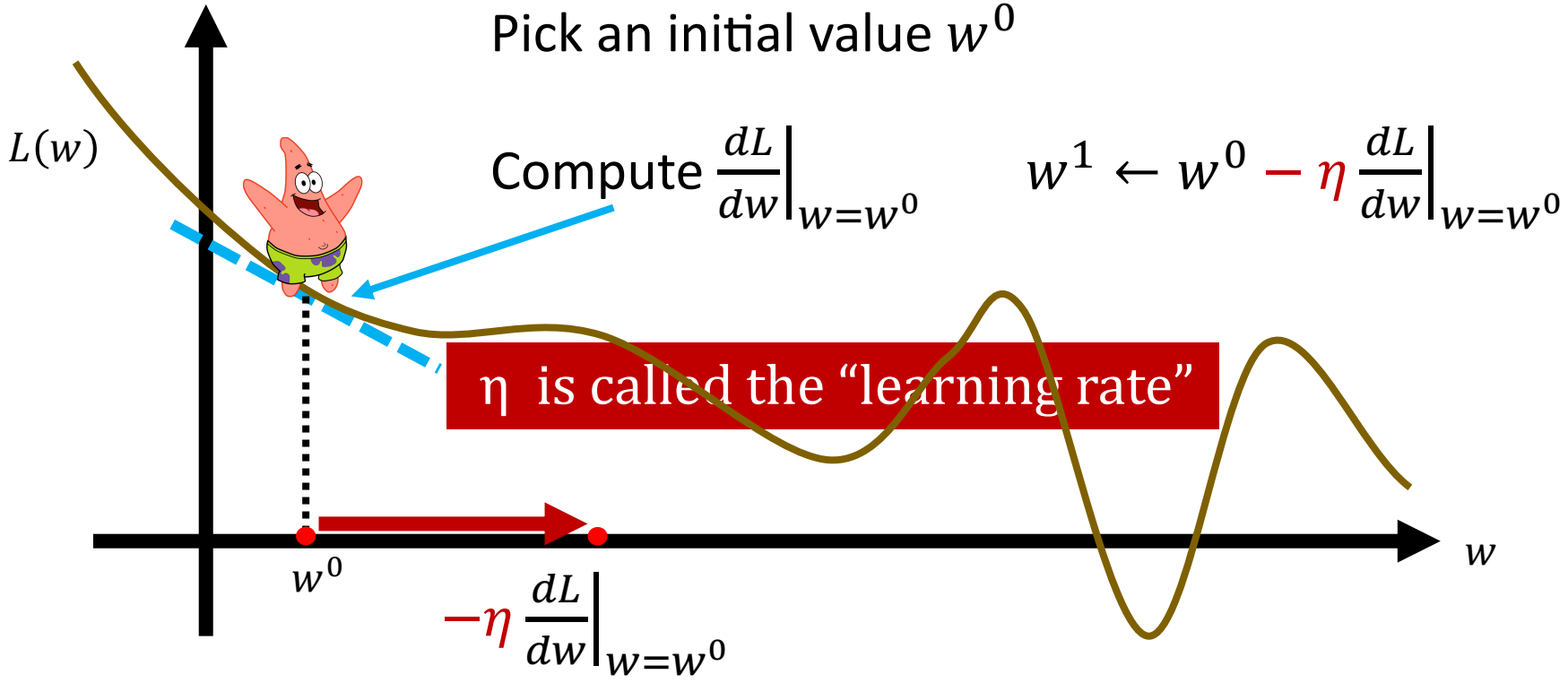
Compute $\left.\frac{dL}{dw}\right|_{w=w^0}$

Negative → Increase w

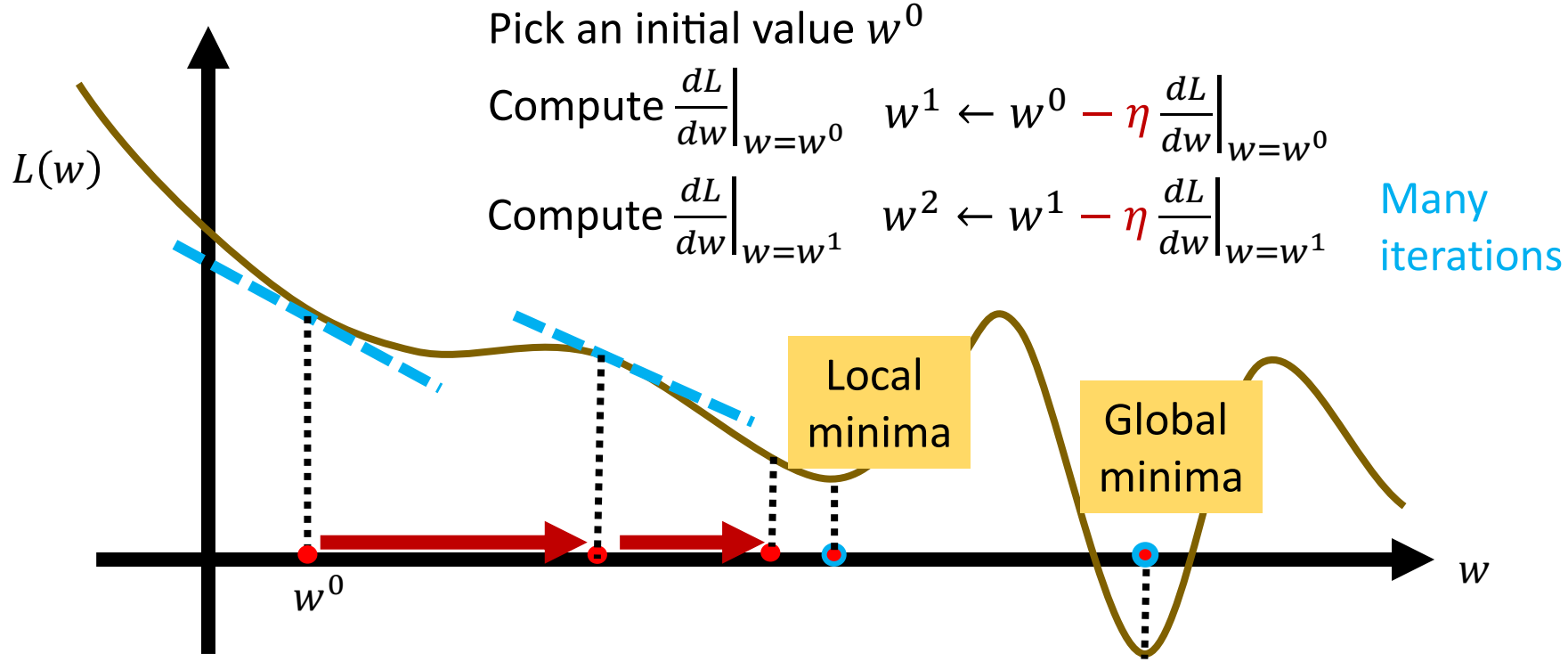
Positive → Decrease w

🙅‍♂️🙅‍♀️ Imagine there are fog of war

Step 3: Gradient Descent



Step 3: Gradient Descent



Step 3: Gradient Descent

$$\begin{bmatrix} \frac{\partial L}{\partial w} \\ \frac{\partial L}{\partial b} \end{bmatrix}$$ gradient

- How about two parameters? $w^*, b^* = arg \min_{w,b} L(w,b)$

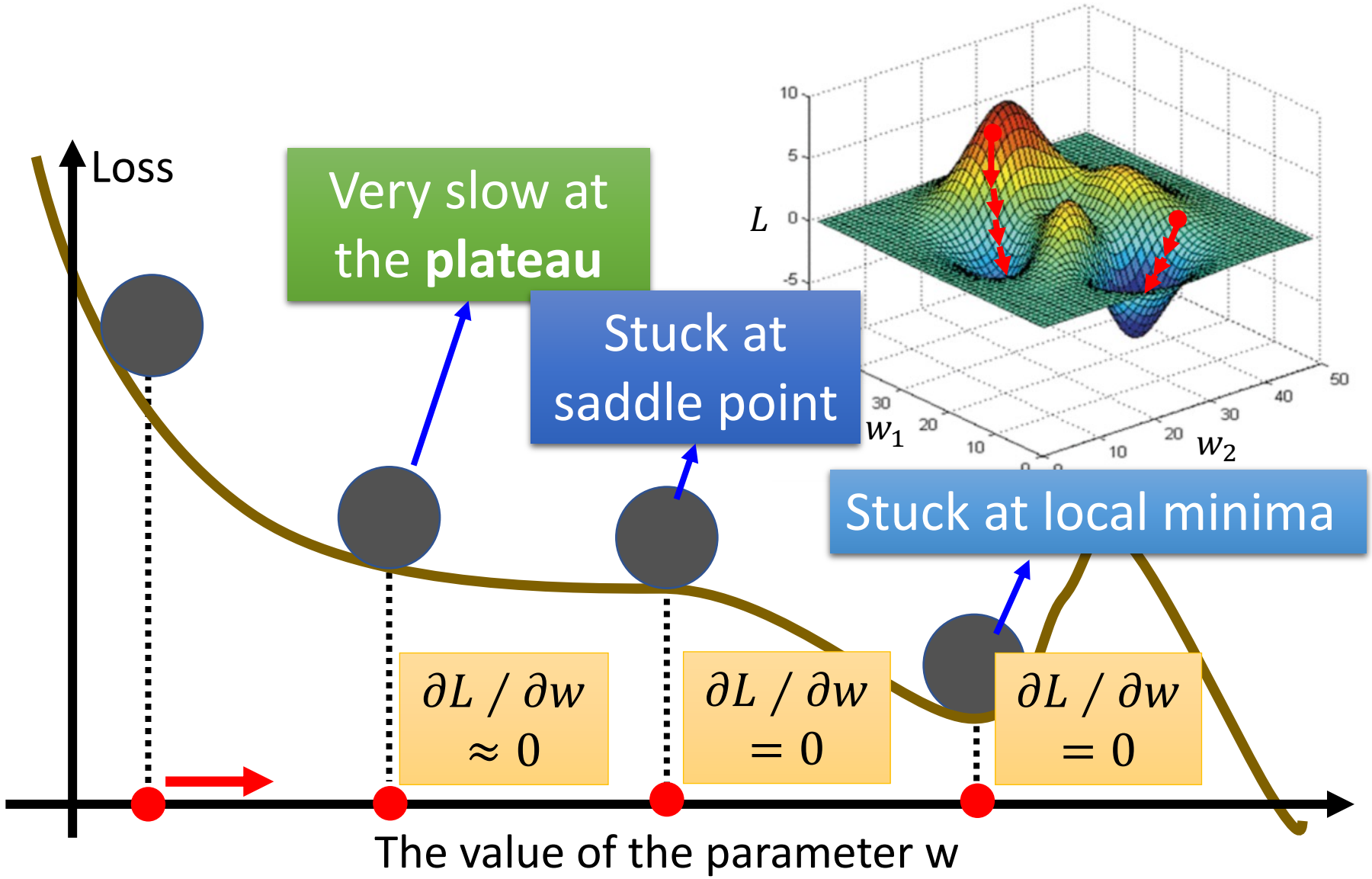
1. (Randomly) Pick an initial value w^0, b^0
2. Compute $\frac{\partial L}{\partial w}|_{w=w^0,b=b^0}$, $\frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$

$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

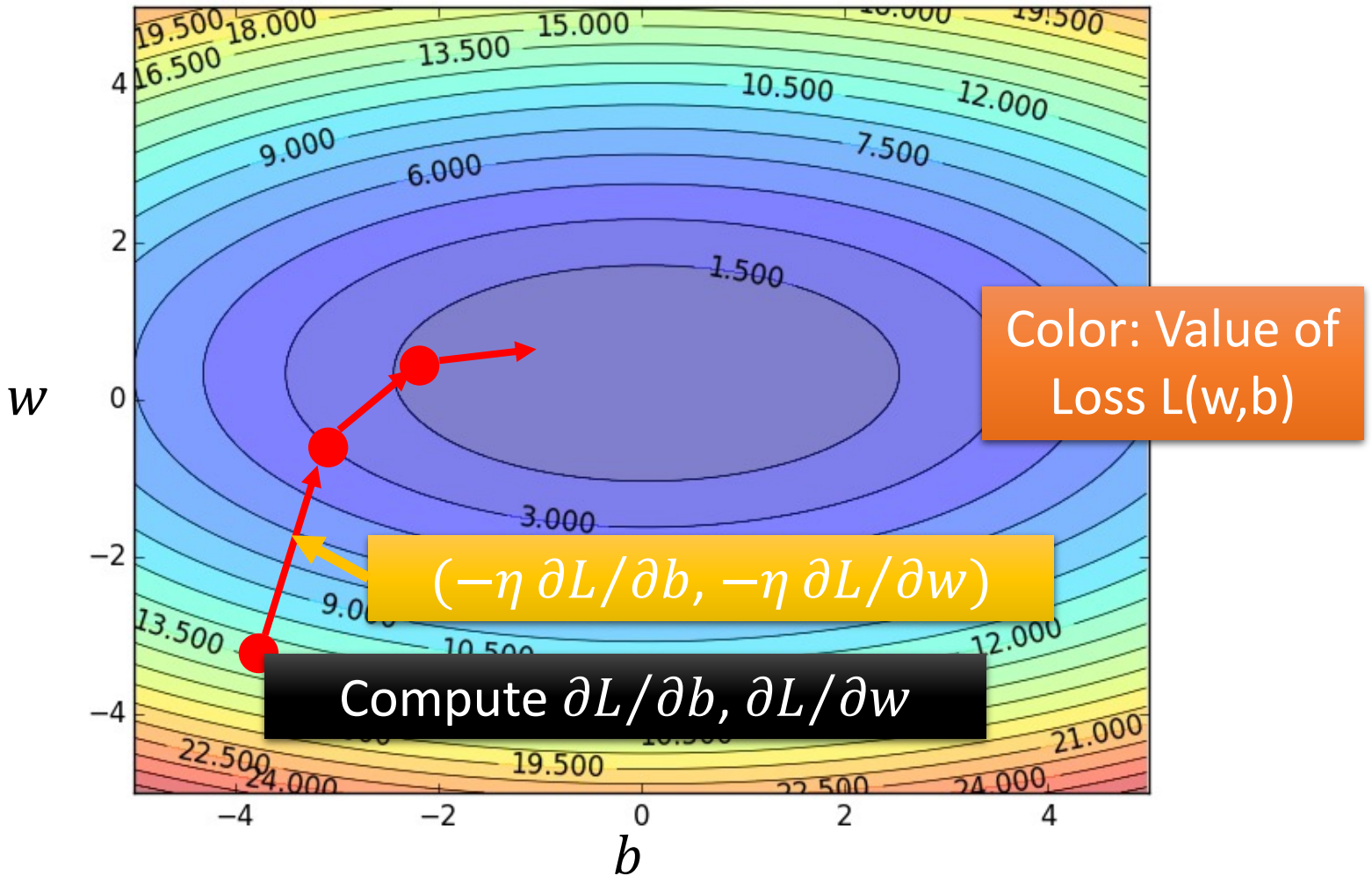
3. Compute $\frac{\partial L}{\partial w}|_{w=w^1,b=b^1}$, $\frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$

$$w^2 \leftarrow w^1 - \eta \frac{\partial L}{\partial w}|_{w=w^1,b=b^1} \qquad b^2 \leftarrow b^1 - \eta \frac{\partial L}{\partial b}|_{w=w^1,b=b^1}$$

Step 3: Gradient Descent



Step 3: Gradient Descent



Step 3: Gradient Descent

- Formulation of $\partial L/\partial w$ and $\partial L/\partial b$

$$L(w,b) = \sum_{n=1}^{N} \left(\hat{y}^n - (b + w \cdot x^n)\right)^2$$

$$\frac{\partial L}{\partial w} = \sum_{n=1}^{N} 2\left(\hat{y}^n - (b + w \cdot x^n)\right)(-x^n)$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{N} 2\left(\hat{y}^n - (b + w \cdot x^n)\right)(-1)$$

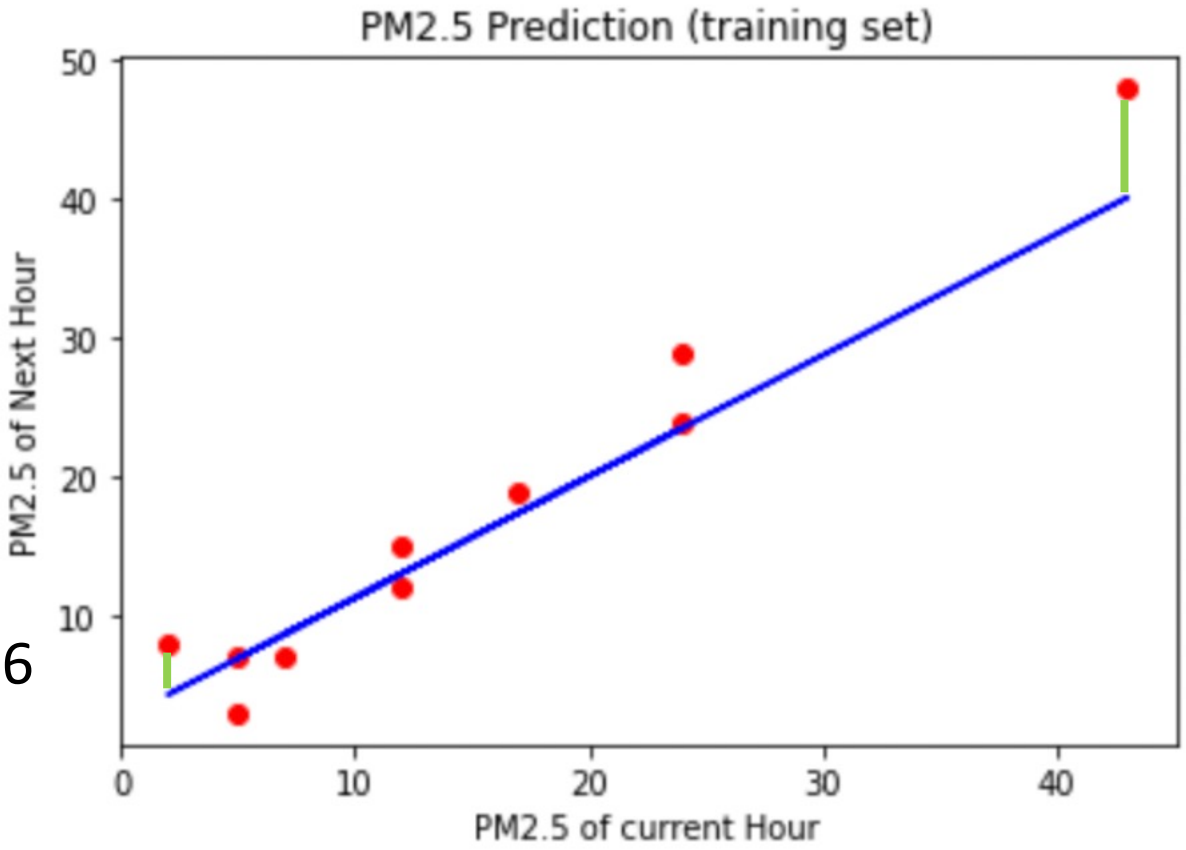
How's the results?

$y = b + w \cdot x_{PM2.5}$

b = 2.57

w = 0.8743

Training Data



Performance on Training Data

$$= \sqrt{\frac{\sum_{n=1}^{N}(\hat{y}^n - yn)^2}{N}} = 3.6126$$

root-mean-square error

How's the results? - Generalization

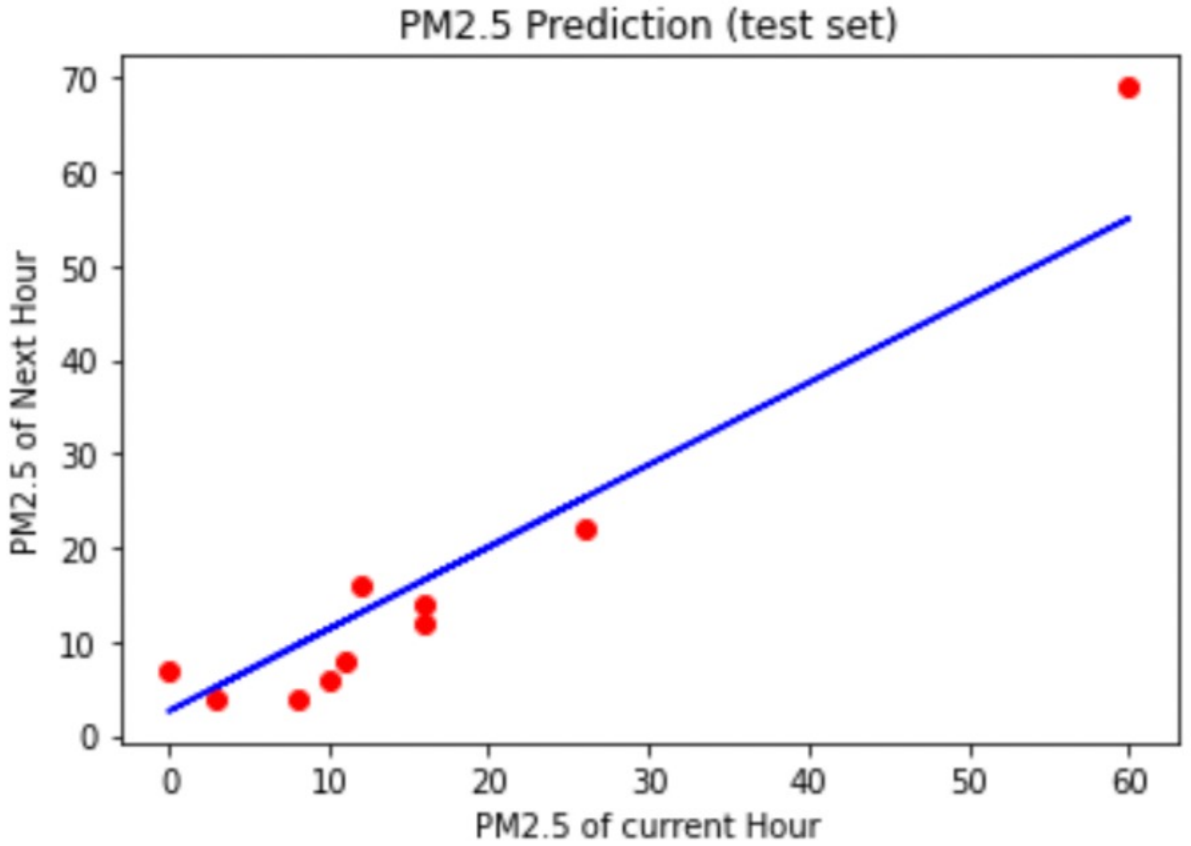
What we really care about is the error on new data (test data)

$y = b + w \cdot x_{PM2.5}$

b = 2.57

w = 0.8743

Test data



Performance on Test Data

$$= \sqrt{\frac{\sum_{n=1}^{N}(\hat{y}^n - yn)^2}{N}} = 5.8294$$

> Training Data (3.6126)

Model Selection

- Polynomial Regression

$$y = b + w \cdot x_{PM2.5}$$

$$y = b + w_1 \cdot x_{PM2.5} + w_2 \cdot (x_{PM2.5})^2$$

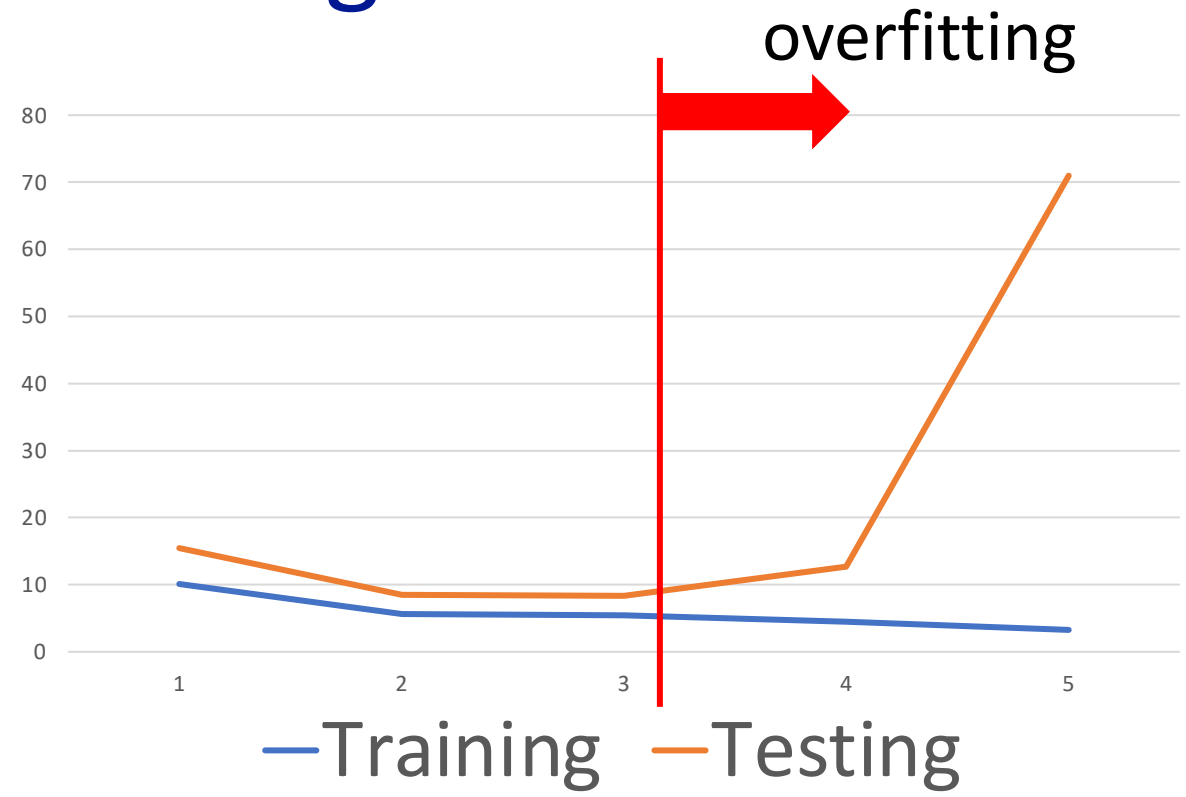
$$y = b + w_1 \cdot x_{PM2.5} + w_2 \cdot (x_{PM2.5})^2 + w_3 \cdot (x_{PM2.5})^3$$

$$y = b + w_1 \cdot x_{PM2.5} + w_2 \cdot (x_{PM2.5})^2 + w_3 \cdot (x_{PM2.5})^3 + w_4 \cdot (x_{PM2.5})^4$$

$$y = b + w_1 \cdot x_{PM2.5} + w_2 \cdot (x_{PM2.5})^2 + w_3 \cdot (x_{PM2.5})^3 + w_4 \cdot (x_{PM2.5})^4 + w_5 \cdot (x_{PM2.5})^5$$

Be careful of overfitting!!!

Overfitting

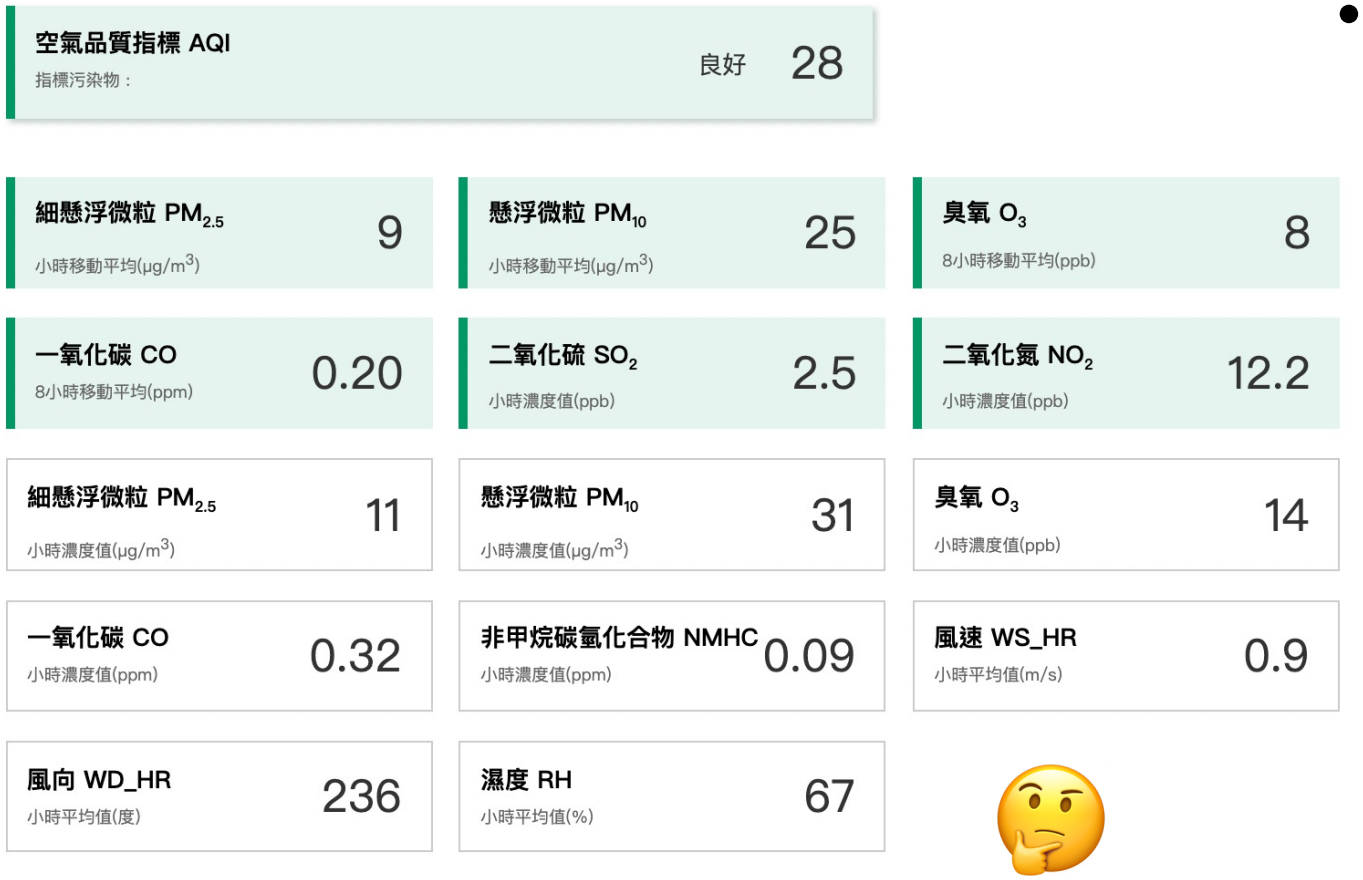


	Training	Testing
1	10.11	15.43
2	5.62	8.49
3	5.48	8.35
4	4.51	12.69
5	3.27	70.98

A more complex model does not always lead to better performance on ***testing data***.

This is ***Overfitting***. → Select suitable model

Are there any other hidden factors?



- Add more features

$$y = b + w_1 \cdot x_{\text{PM2.5}} + w_2 \cdot (x_{\text{PM2.5}})^2 + w_3 \cdot x_{co} + w_3 \cdot (x_{co})^2 + w_4 \cdot x_{so_2} + w_4 \cdot (x_{so_2})^2 + w_5 \cdot x_{o_3} + w_6 \cdot (x_{o_3})^2$$

Be careful of overfitting!!!

Back to step 2: Regularization

$$y = b + \sum w_i x_i$$

$$L = \sum_n \left(\hat{y}^n - \left(b + \sum w_i x_i \right) \right)^2 \quad \boxed{+\lambda \sum (w_i)^2}$$

The functions with smaller w_i are better

- Smaller w_i means … smoother

$$y = b + \sum w_i x_i$$

$$y + \sum w_i \Delta x_i = b + \sum w_i (x_i + \Delta x_i)$$

- We believe smoother function is more likely to be correct

Do you have to apply regularization on bias?

Regularization

smoother

λ	Training	Testing
0	5.48	8.35
1	6.79	8.19
10	7.01	7.53
100	7.43	8.12
1000	8.54	8.99
10000	9.25	9.51

How smooth?

Select λ obtaining the best model

- Training error: largerλ, considering the training error less
- We prefer smooth function, but don't be too smooth.

Regression Implementation (scikit-learn)

Steps

1. Collect data
2. Preprocess data
3. Prepare input and output based on your task
4. Split data into training set and test set
5. Train model by training set
6. Save model
7. Predict on test set
8. Evaluate performance on test set

pip install sklearn

Remaining Steps

4. Split collected data into training set and test set
5. Train model by training set
6. Save model
7. Predict on test set
8. Evaluate performance on test set

```
1  X_train.shape
```

```
(1694, 1)
```

```
1  Y_train.shape
```

```
(1694,)
```

```
1  import os
2  from sklearn.model_selection import train_test_split
3  from sklearn.linear_model import LinearRegression
4  import pickle
5  from sklearn import metrics
6
7  X = X.reshape(X.shape[0], 1)
8
9  # step4: Split collected data into training set and test set
10 X_train, X_test, Y_train, Y_test = \
11             train_test_split(X, Y, test_size=0.4, random_state=101)
12
13 # step5: Train model by training set
14 lr = LinearRegression()
15 lr.fit(X_train, Y_train)
16
17 # step6: save trained model
18 with open('lr_model.pkl', 'wb') as f:
19     pickle.dump(lr, f)
20
21 # step7: Predict on test set
22 y_pred = lr.predict(X_test)
23
24 # step8: Evaluate performance on test set
25 rmse = np.sqrt(metrics.mean_squared_error(Y_test, y_pred))
26 print(rmse)
```

```
5.593930245760409
```

https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LinearRegression.html

https://en.wikipedia.org/wiki/Root-mean-square_deviation

View Results

```
output = pd.DataFrame({'input': X_test.reshape(X_test.shape[0],), \
                       'predict': y_pred, 'ground_truth': Y_test})
output.head(20)
```

	input	predict	ground_truth
0	39.0	37.381889	27.0
1	10.0	11.171589	8.0
2	27.0	26.536248	21.0
3	6.0	7.556376	5.0
4	12.0	12.979196	19.0
5	36.0	34.670479	33.0
6	24.0	23.824837	18.0
7	25.0	24.728641	15.0
8	17.0	17.498213	14.0
9	15.0	15.690606	13.0
10	13.0	13.883000	14.0
11	24.0	23.824837	25.0
12	15.0	15.690606	16.0
13	30.0	29.247658	29.0
14	26.0	25.632444	29.0

```
lr.coef_
```

```
array([0.90380344])
```

```
lr.intercept_
```

```
2.1335548953225434
```

$$y = 0.90380344x + 2.1335548953225434$$

Polynomial Regression

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.PolynomialFeatures.html

```
1  import numpy as np
2  import pandas as pd
3  from sklearn.model_selection import train_test_split
4  from sklearn.linear_model import LinearRegression
5  from sklearn.preprocessing import PolynomialFeatures
6
7  X = X.reshape(X.shape[0], 1)
8
9  X_train, X_test, Y_train, Y_test = \
10         train_test_split(X, Y, test_size=0.4, random_state=101)
11
12 polynomial_features = PolynomialFeatures(degree = 4)
13 X_trans_train = polynomial_features.fit_transform(X_train)
14 X_trans_test = polynomial_features.fit_transform(X_test)
15
16
17 lr2 = LinearRegression()
18 lr2.fit(X_trans_train, Y_train)
19
20 with open('lr2_model.pkl', 'wb') as f:
21     pickle.dump(lr2, f)
```

$$y = b + w_1 \cdot x_{PM2.5} + w_2 \cdot (x_{PM2.5})^2 + w_3 \cdot (x_{PM2.5})^3 + w_4 \cdot (x_{PM2.5})^4$$

```
1  lr2.coef_
```

```
array([ 0.00000000e+00,  1.12819023e+00, -1.64953586e-02,  4.19417119e-04,
       -3.19230000e-06])
```

```
1  lr.intercept_
```

```
2.1335548953225434
```

```
1  X_train.shape
```

```
(1694, 1)
```

```
1  X_trans_train.shape
```

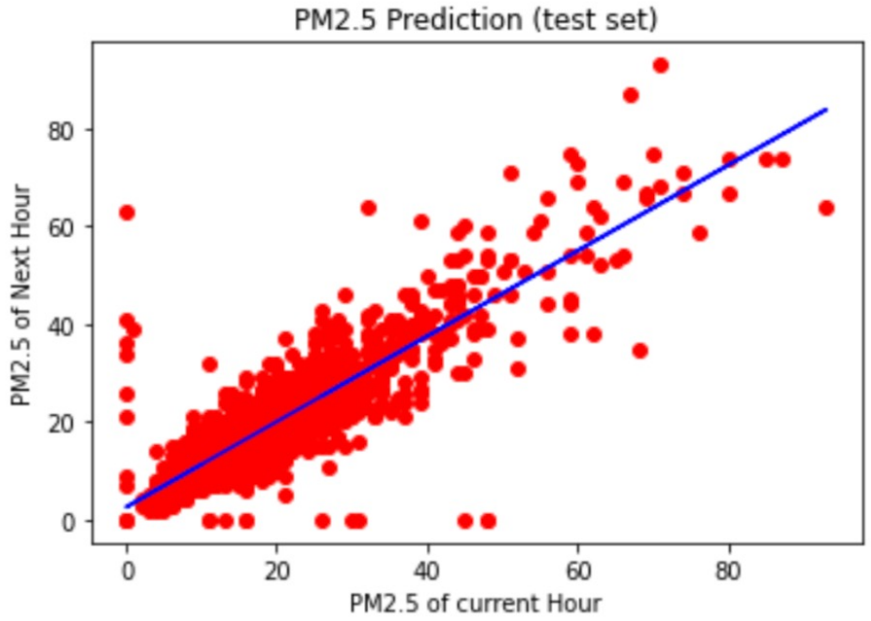
```
(1694, 5)
```

View Results

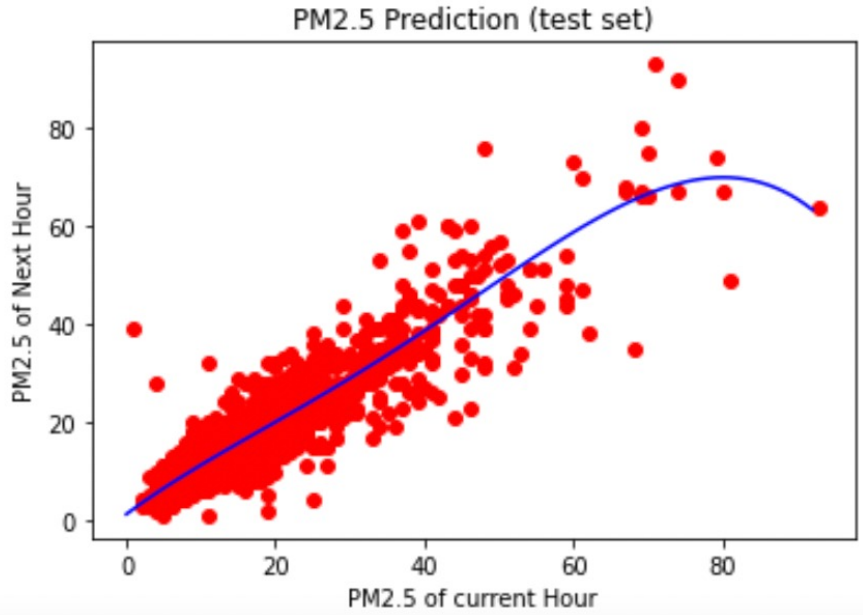
Just to make the line look better

```
1  import matplotlib.pyplot as plt
2
3  _x = np.array([e for e in range(int(max(X_test)[0]))])
4  polynomial_features = PolynomialFeatures(degree = 4)
5  _X_trans_test = polynomial_features.fit_transform(_x.reshape(_x.shape[0], 1))
6
7  plt.scatter(X_test, Y_test, color = 'red')
8  plt.plot(_x, lr2.predict(_X_trans_test), color = 'blue')
9  plt.title('PM2.5 Prediction (test set)')
10 plt.xlabel("PM2.5 of current Hour")
11 plt.ylabel("PM2.5 of Next Hour")
12 plt.show()
```

```
1  plt.scatter(X_test, Y_test, color = 'red')
2  plt.plot(X_test, y_pred, color = 'blue')
3  plt.title('PM2.5 Prediction (test set)')
4  plt.xlabel("PM2.5 of current Hour")
5  plt.ylabel("PM2.5 of Next Hour")
6  plt.show()
```

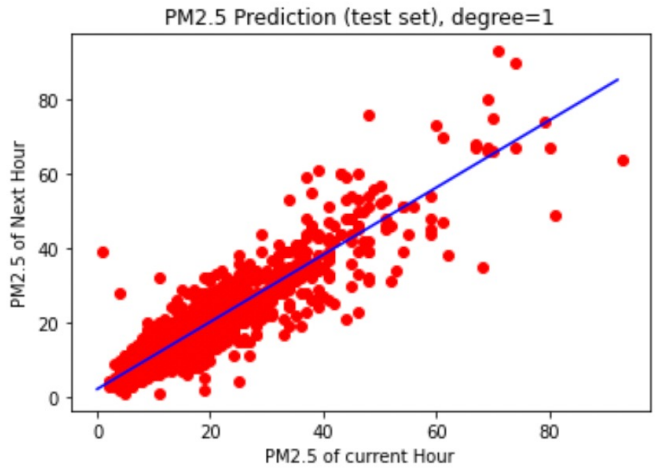


Degree = 1

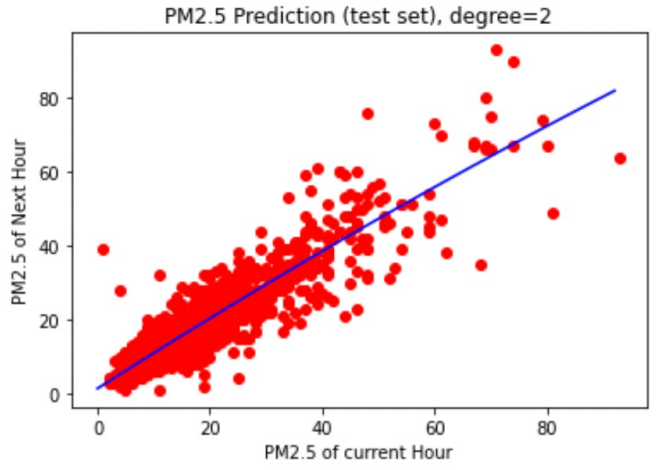


Degree = 4

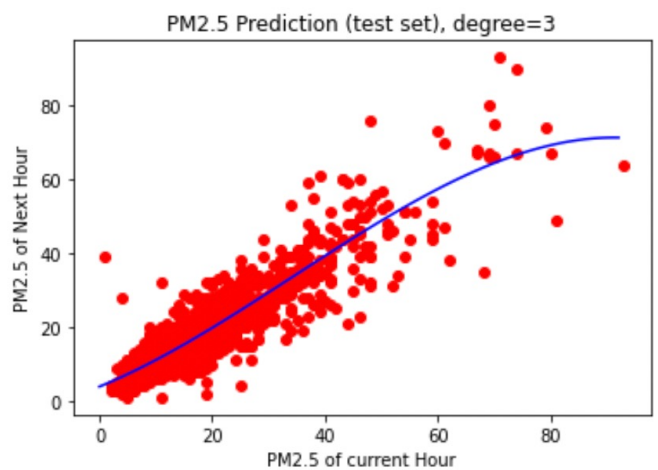
RMSE: 5.5939



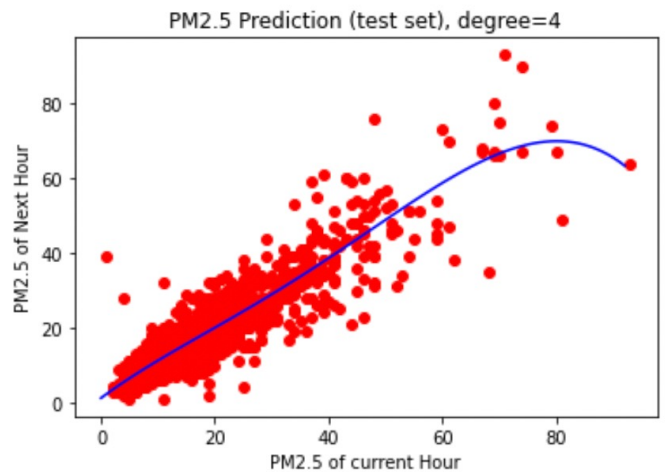
RMSE: 5.5972



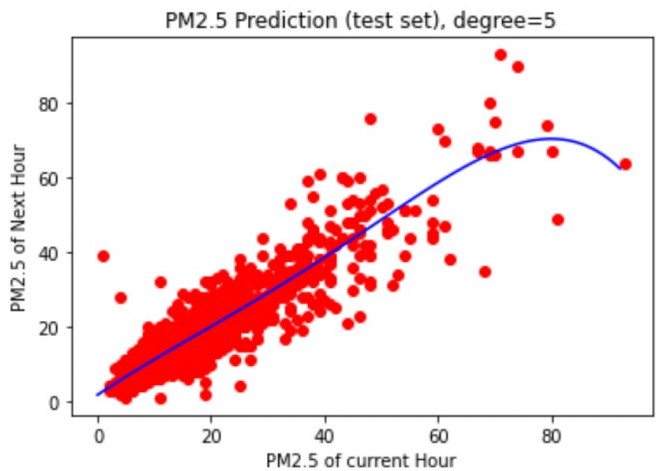
RMSE: 5.5512



RMSE: 5.5555

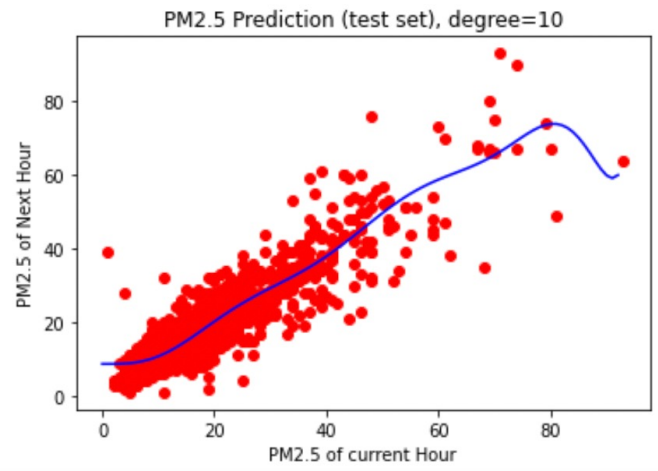


RMSE: 5.5510



...

RMSE: 5.6099



Implement gradient descent by NumPy

$$\frac{\partial L}{\partial w} = \sum_{n=1}^{10} 2\left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)\left(-x_{cp}^n\right)$$

$$\frac{\partial L}{\partial b} = \sum_{n=1}^{10} 2\left(\hat{y}^n - \left(b + w \cdot x_{cp}^n\right)\right)(-1)$$

```
import numpy as np
Load features and labels as numpy arrays
    e.g., X = np.array(X)
Initial bias (b), weight (w), learning rate (lr), and iteration
for i in range(iteration):
    b_grad, w_grad = 0.0, 0.0
    for j in range(length of X):
        b_grad = b_grad – 2.0*(y[j] – b – w * X[j]) * 1
        w_grad = w_grad – 2.0*(y[j] – b – w * X[j]) * X[j]
    b = b – lr * b_grad
    w = w – lr * w_grad
```

You can also use dot product to change this process:
y = np.dot(x,w) + b

Reference

- Lecture Slides from Machine Learning by Prof. Hung-Yi Lee
  - https://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/Regression.pdf