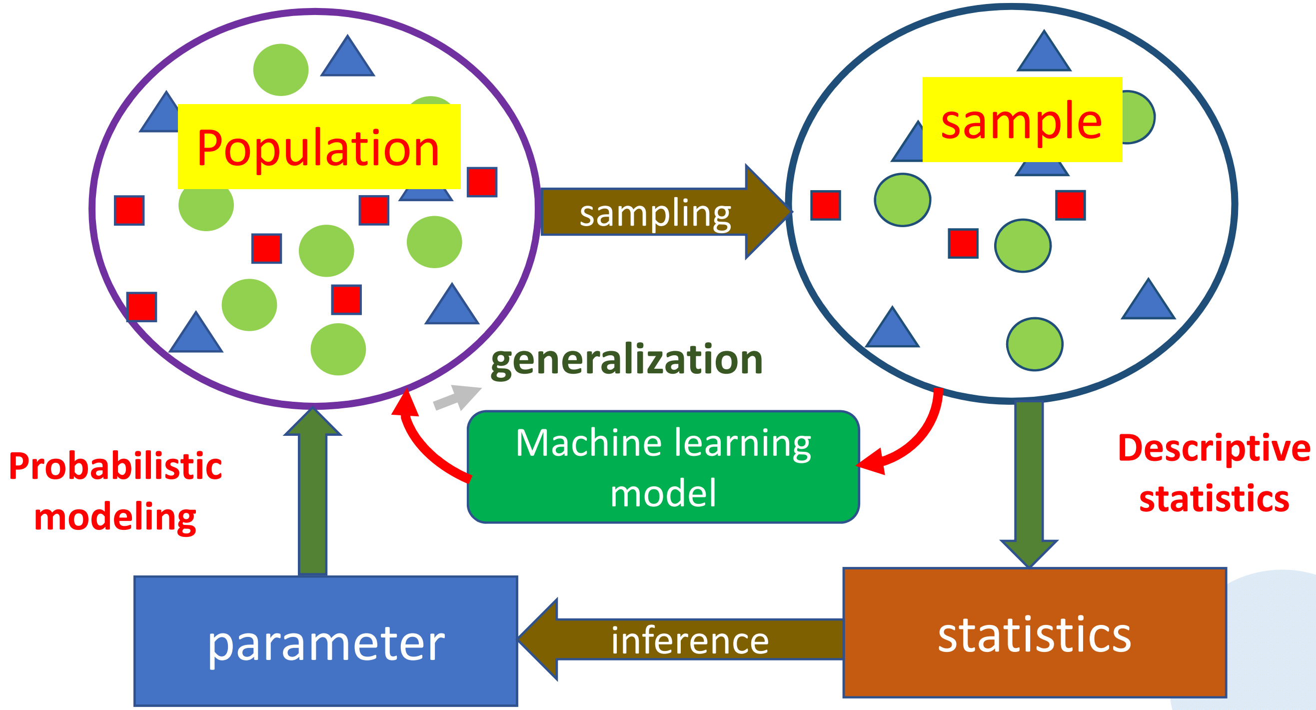
Machine Learning Introduction

Yun-Nan Chang

NATIONAL SUN YAT-SEN UNIVERSITY

1 Machine Learning

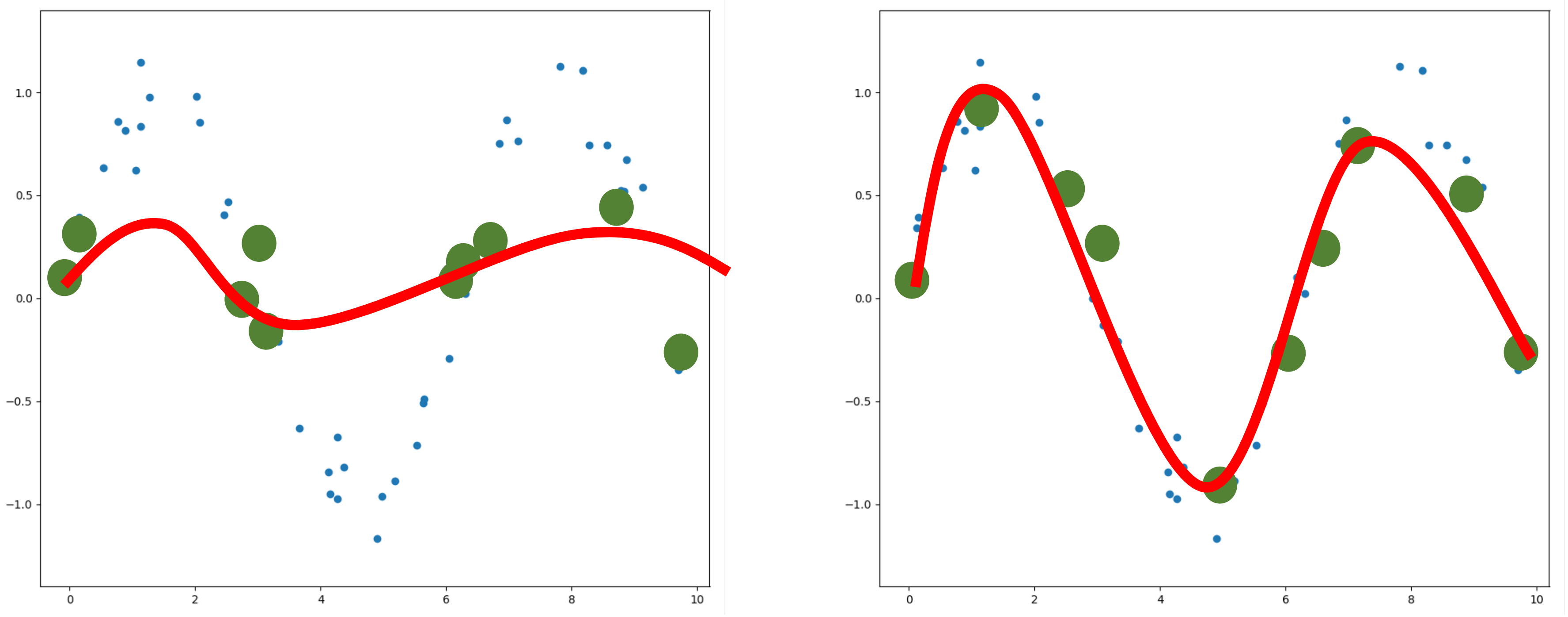
Sampling & Learning



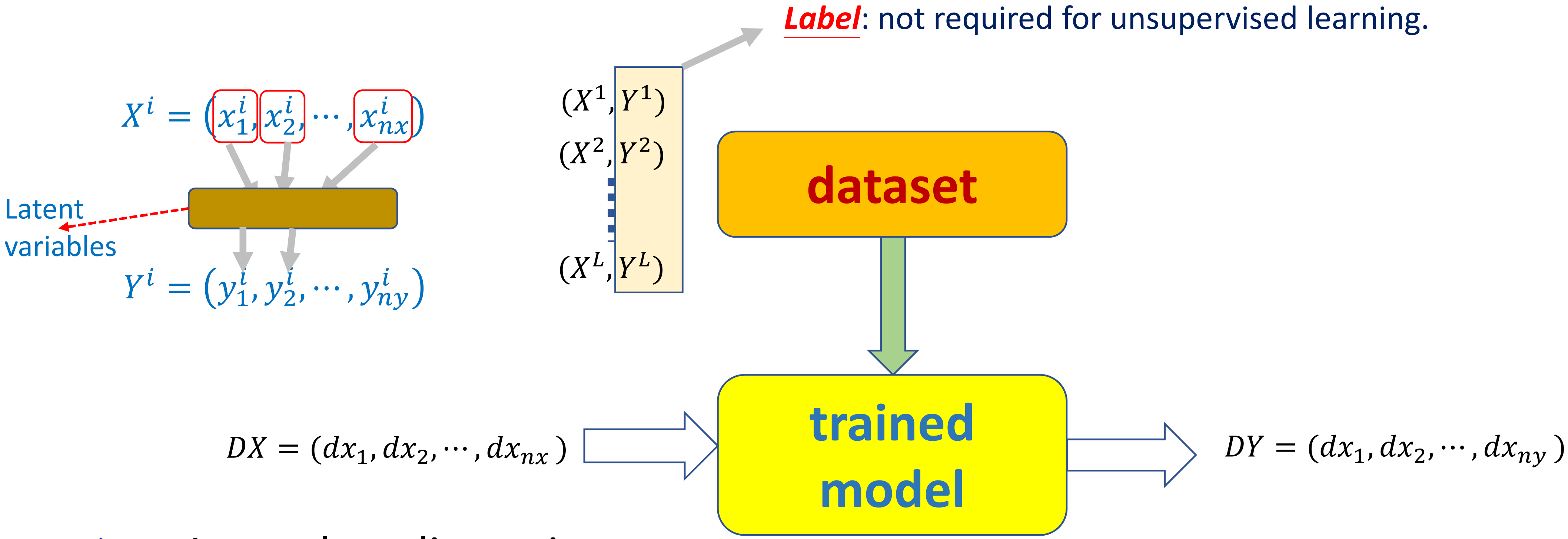
Overfitting vs Underfitting

Sampling

◈Same dataset but different results.



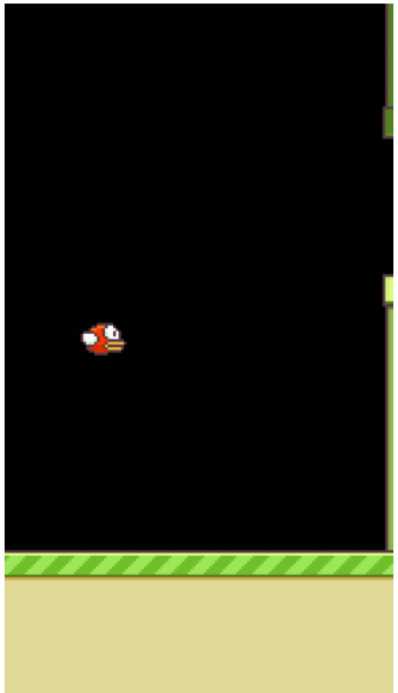
Machine learning

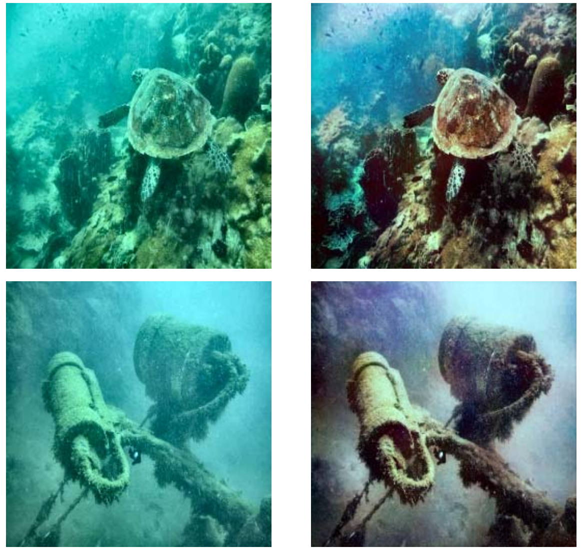


- ***nx***: Input data dimension
- ***ny***: output data dimension.
  - ***ny=1*** for binary classification, regression.

Machine Learning Application

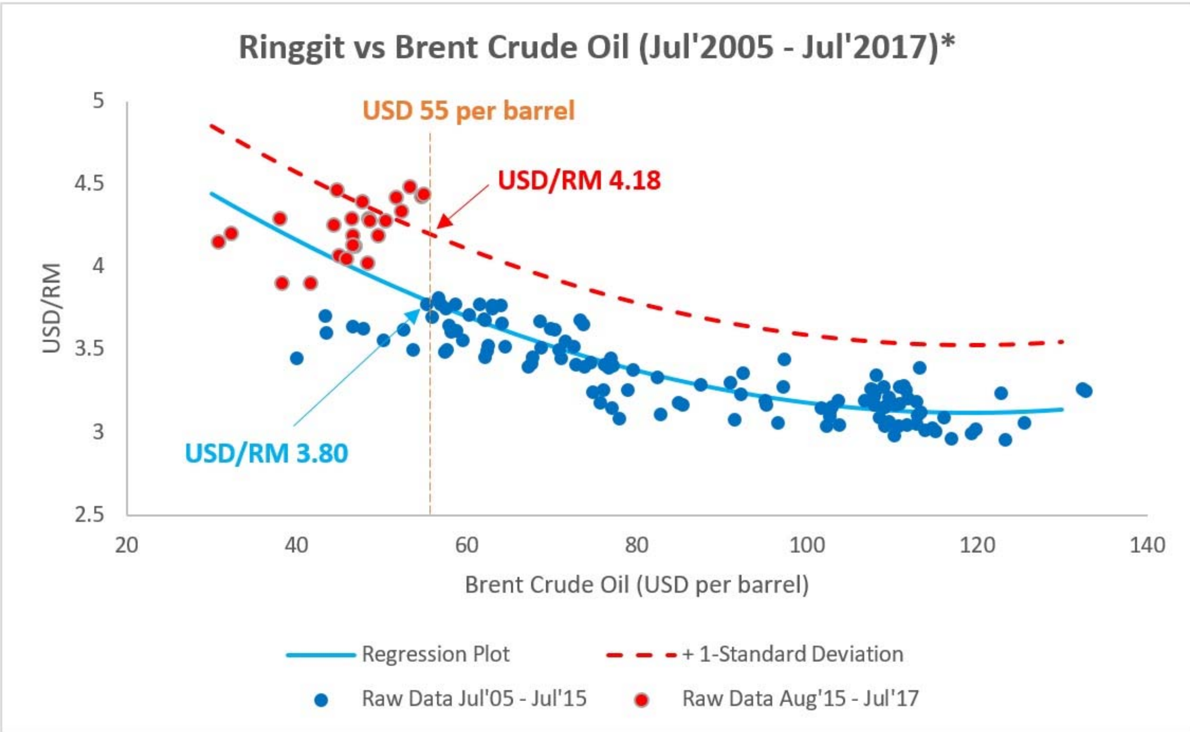
- Image recognition
- Create Picture
- Chatbot
- AlphaGo
- Play video game





Regression

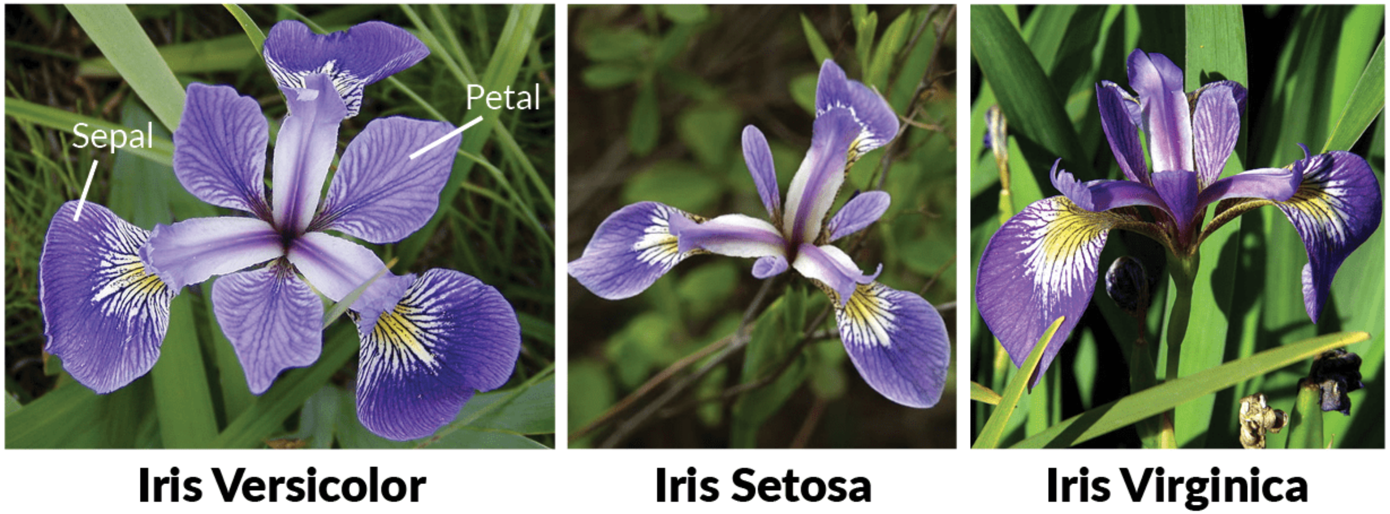
- Oil price forecast



** The data range was chosen from July 2005 onwards because the Ringgit was unpegged after July 2005.*

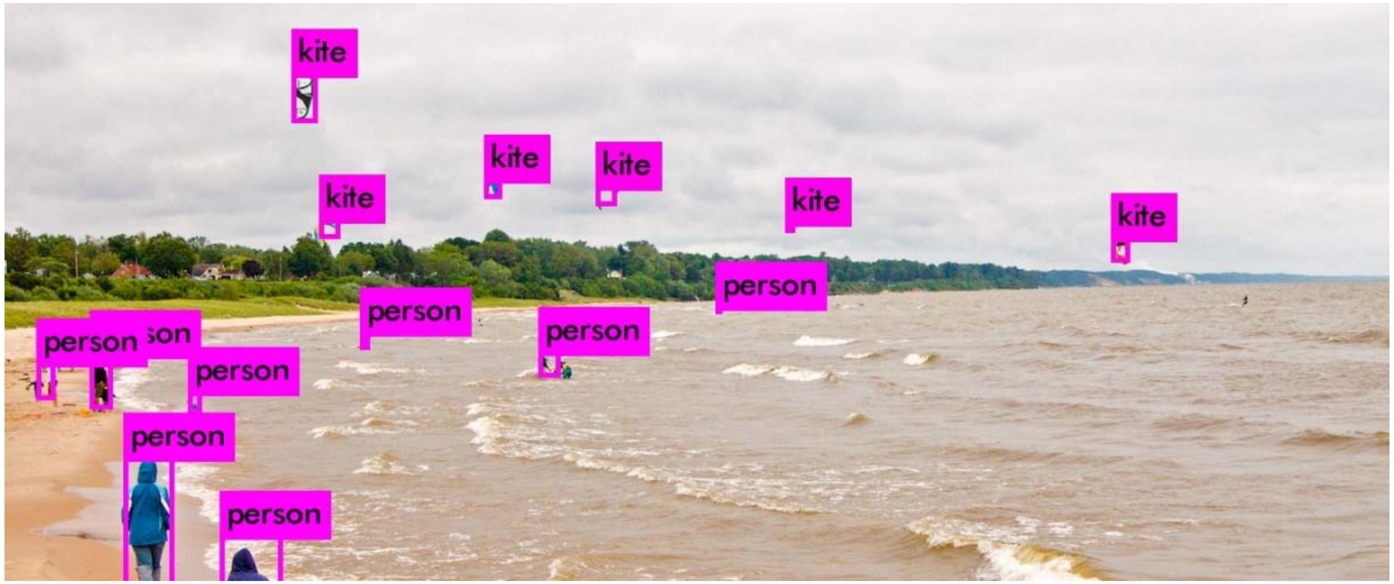
Classification

- Iris flowers classification



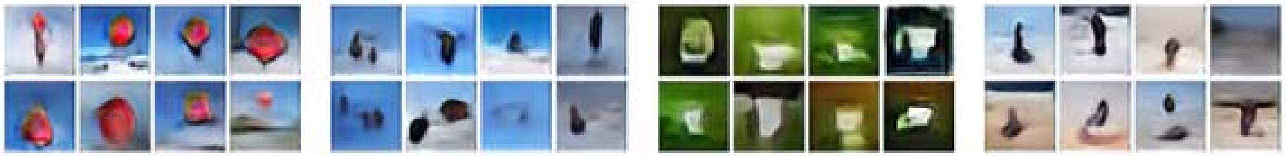
Pattern recognition

- Image recognition



Generator

◆Create Picture



A stop sign is flying in blue skies.

A herd of elephants flying in the blue skies.

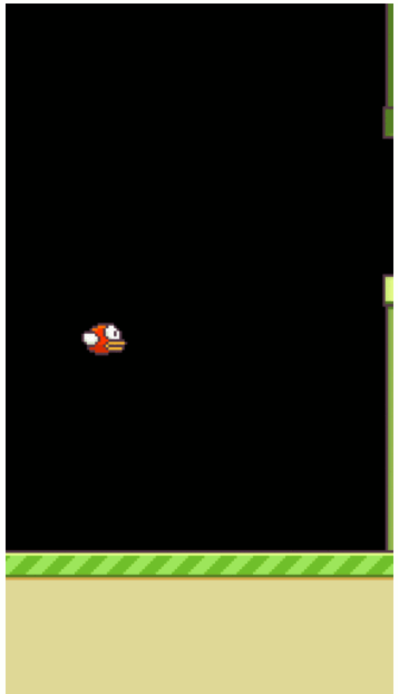
A toilet seat sits open in the grass field.

A person skiing on sand clad vast desert.

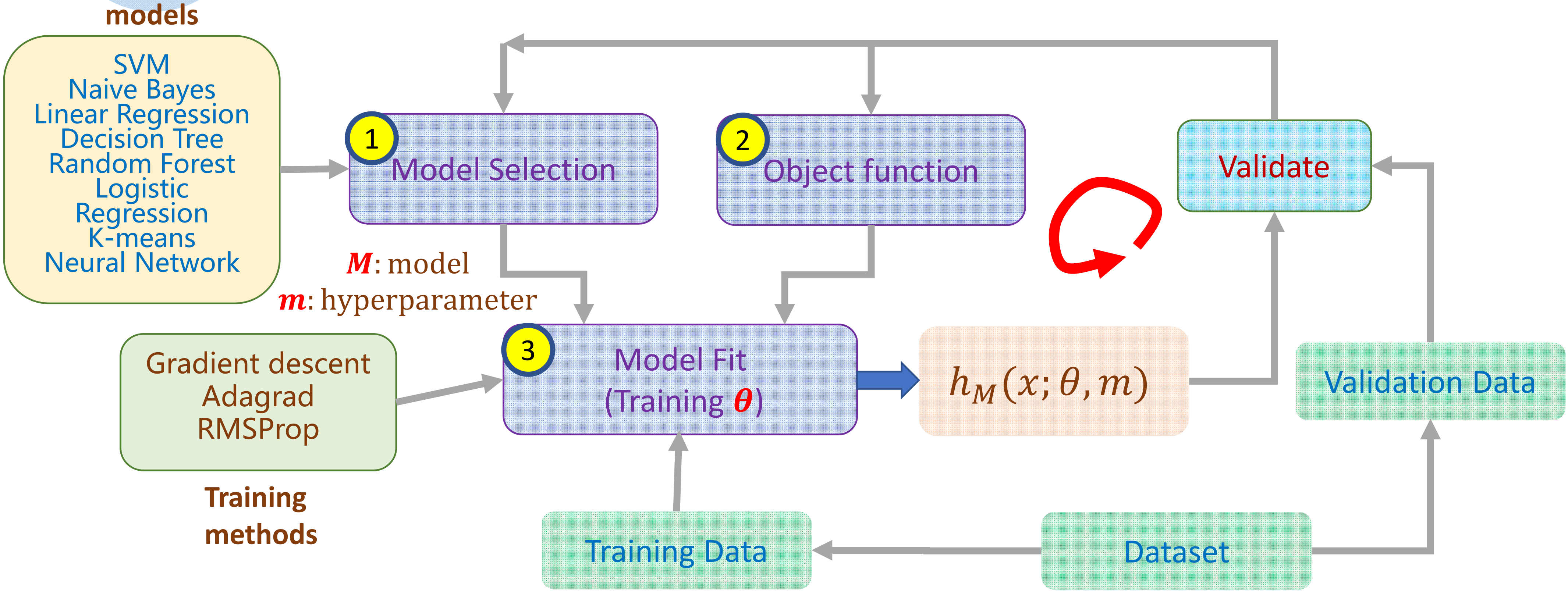
Figure 1: Examples of generated images based on captions that describe novel scene compositions that are highly unlikely to occur in real life. The captions describe a common object doing unusual things or set in a strange location.

Play Game

- AlphaGo
- Play video game



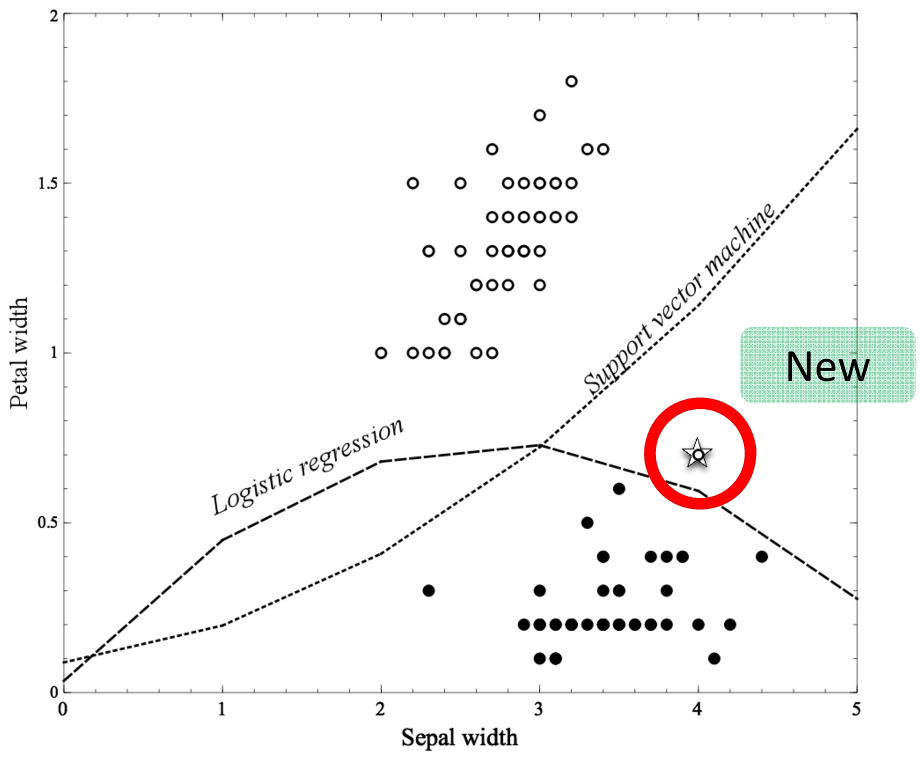
Machine Learning Process



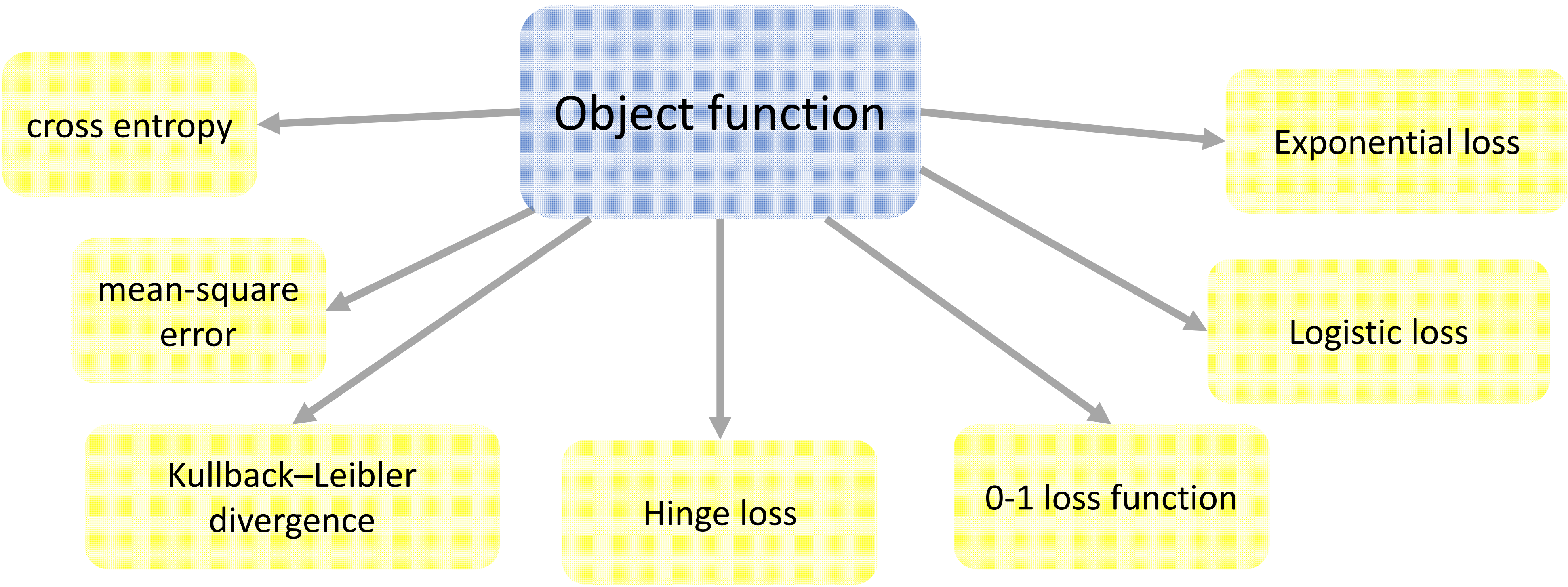
Model selection

- Different model we will get different result.

- In SVM, new data is classified into Black group.
- But if you use Logistic regression, it is classified into While group



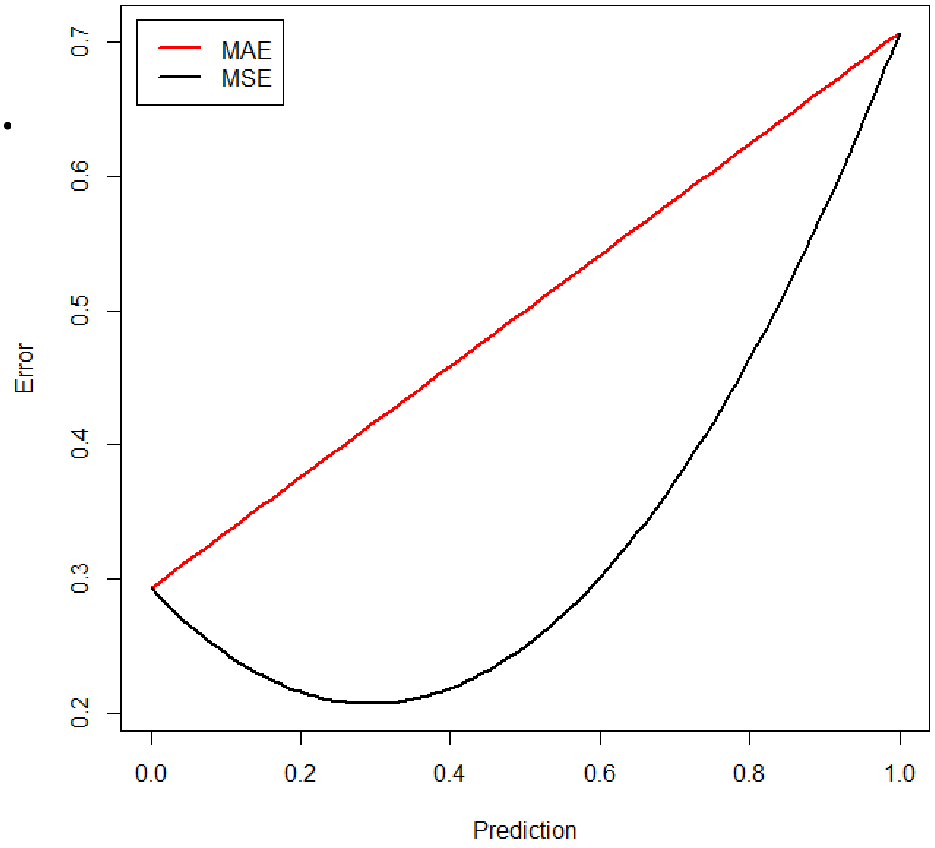
Object function



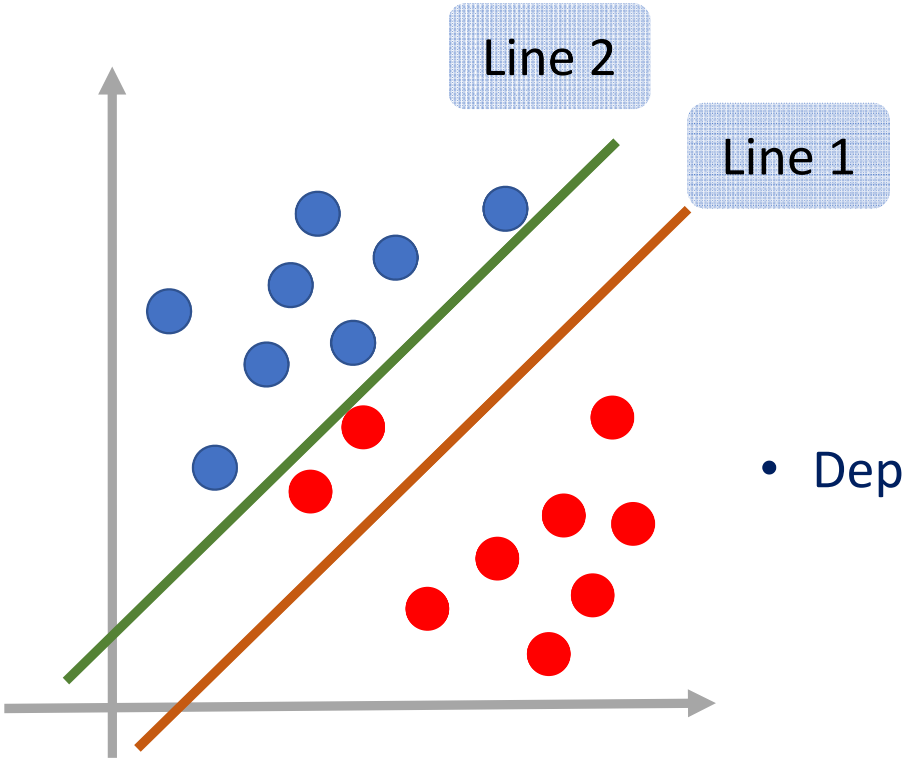
Object function

- Use different object function in same model will get different result.

- Use Mean Absolute Error and Mean Square Error in Logistic Regression



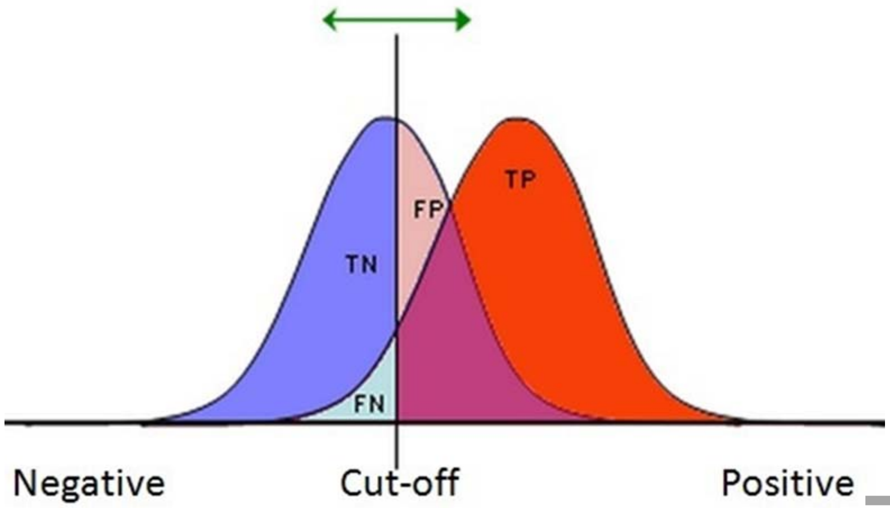
Which separation line for classification is better?

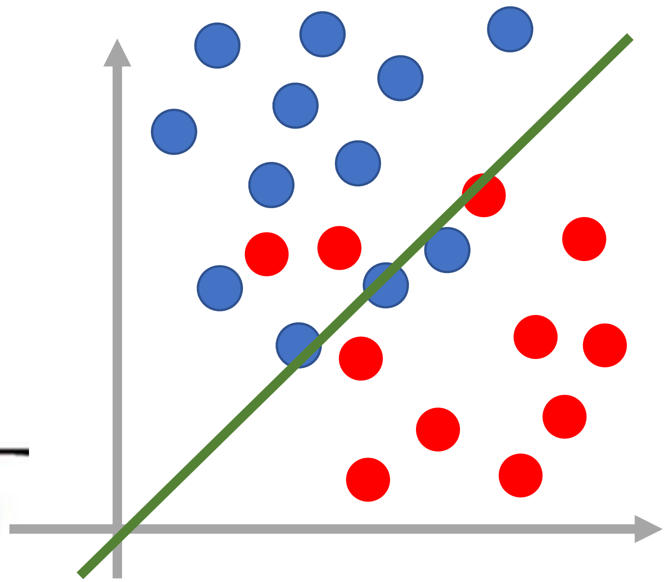


- Depend on your defined **goodness**

Cut-off point for binary classification

- The selection of cut-off will affect decision/prediction outcome
  - Actual positive: TP+FN   Actual negative: TN+FP.





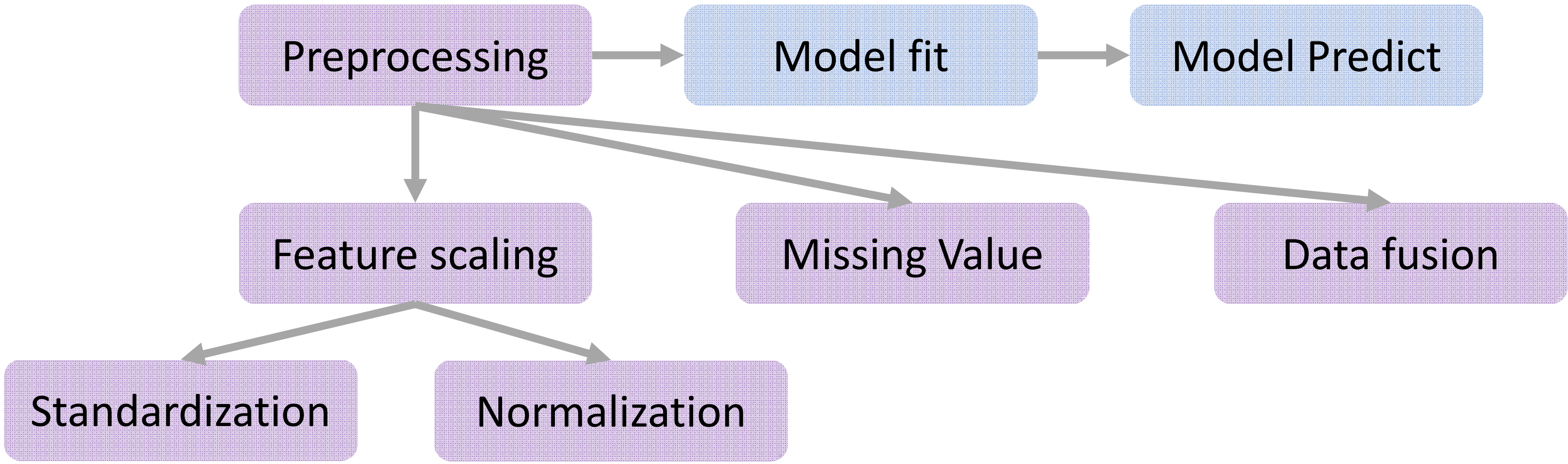
	Actual Yes	Actual No
Predict Yes	TP	FP
Predict No	FN	TN

Confusion Matrix

	Actual Yes	Actual No
Predict Yes	TP (True Positive)	FP (False Positive)
Predict No	FN (False Negative)	TN (True Negative)

Accuracy	$\frac{TP + TN}{Total}$
Sensitivity (Recall)	$\frac{TP}{TP + FN}$
Precision	$\frac{TP}{TP + FP}$
Specificity	$\frac{TN}{TN + FP}$

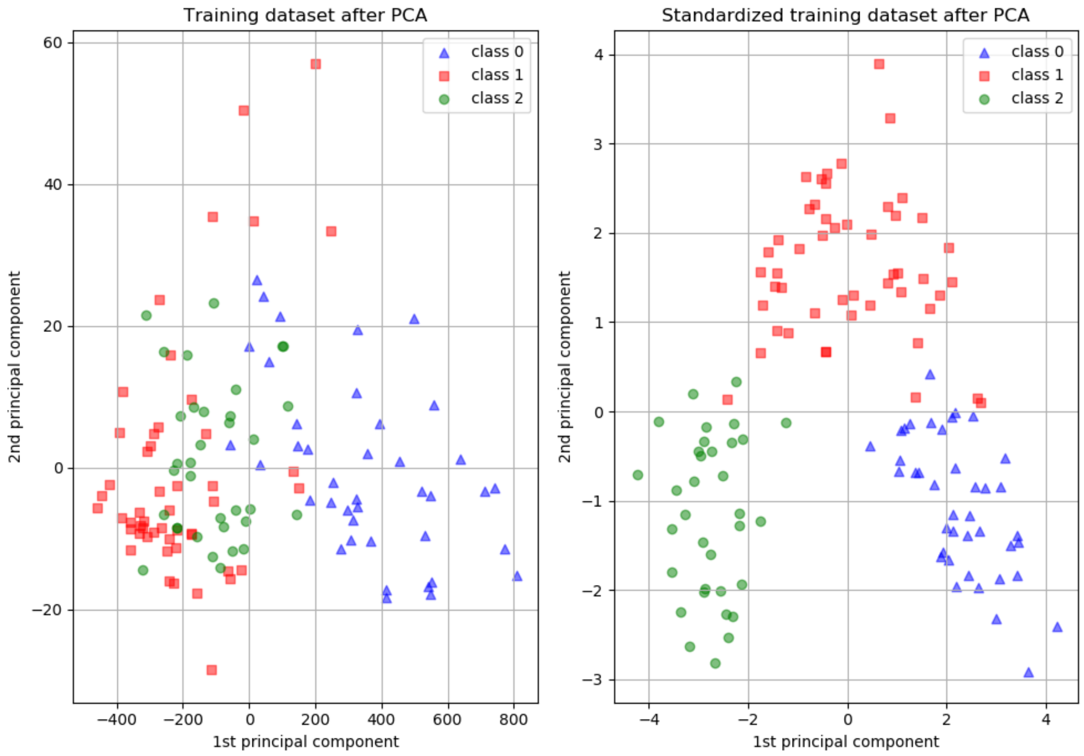
Preprocessing



Preprocessing

Standardization

◈Standardization



Preprocessing

Missing Value

	A	B	C	D
0	1	3	5	8
1	NaN	2	5	2
2	4	7	NaN	2
3	4	2	5	1

Drop

	A	B	C	D
0	1	3	5	8
1	NaN	2	5	2
2	4	7	NaN	2
3	4	2	5	1

Drop Specified column
(If A is NaN)

	A	B	C	D
0	1	3	5	8
1	NaN	2	5	2
2	4	7	NaN	2
3	4	2	5	1

Preprocessing

Missing Value

	A	B	C	D
0	1	3	5	8
1	NaN	2	5	2
2	4	7	NaN	2
3	4	2	5	1

Fill Average

	A	B	C	D
0	1	3	5	8
1	3	2	5	2
2	4	7	5	2
3	4	2	5	1

Fill Mode

	A	B	C	D
0	1	3	5	8
1	4	2	5	2
2	4	7	5	2
3	4	2	5	1

Preprocessing in scikit-learn

Standardization

```
from sklearn import preprocessing
Standard_X=preprocessing.StandardScaler.fit_transform(X)
```

Normalization

```
from sklearn import preprocessing
Normalized_X = preprocessing.normalize(X, norm='l2’)
```

Preprocessing in scikit-learn

MinMaxScaler

```
from sklearn import preprocessing
Scalar_X = preprocessing.MinMaxScaler().fit_transform(X)
```

Missing Value

```
from sklearn.preprocessing import Imputer
imp = Imputer(missing_values='NaN', strategy='mean', axis=0)
imp_X = imp.fit(X)
```

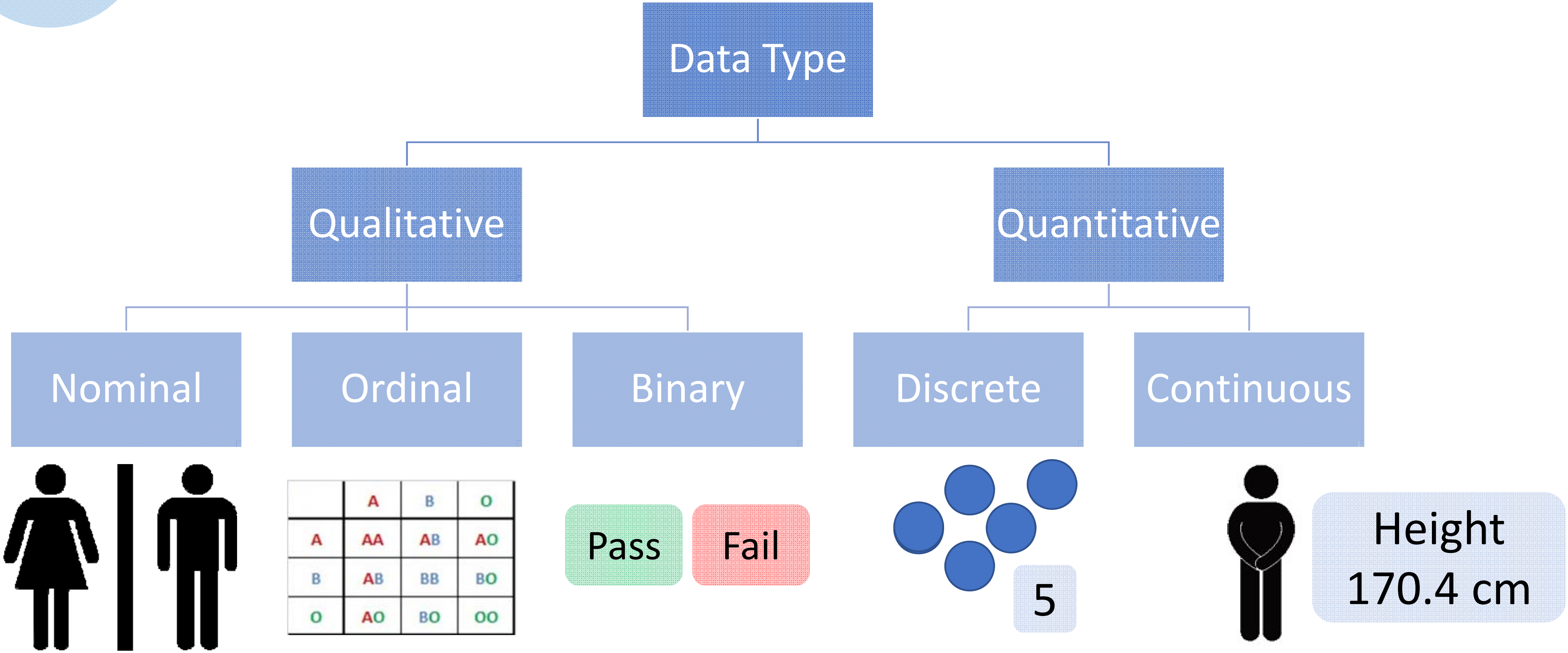
Model Selection

◈ ***Cross-validation*** method can be applied for limited data, which allows a proportion ***(S −1)/S*** of the available data to be used for training.

Model complexity

- We should choose the model with suitable complexity for the size of the given dataset.
  - High complexity -> more freedom -> low bias with potential high variance
  - Low complexity -> less freedom -> low variance with potential high bias
- Should we use all the features in the given dataset?

Type of input variables/feature



One-hot encoding

Fruit
Apple
Orange
Banana
Orange

Encode

Apple	Orange	Banana
1	0	0
0	1	0
0	0	1
0	1	0

Some well-known popular model (M)

- Regression
- SVM (Support Vector Machine)
- KNN (K-Nearest Neighbors)
- Logistic Regression
- Decision Tree
- K-Means
- Random Forest
- Naive Bayes
- Dimensional Reduction Algorithms
- Gradient Boosting Algorithms
- Convolutional Neural Network
- Deep learning

2

Over-fitting
Under-fitting

Polynomial Curve Fitting

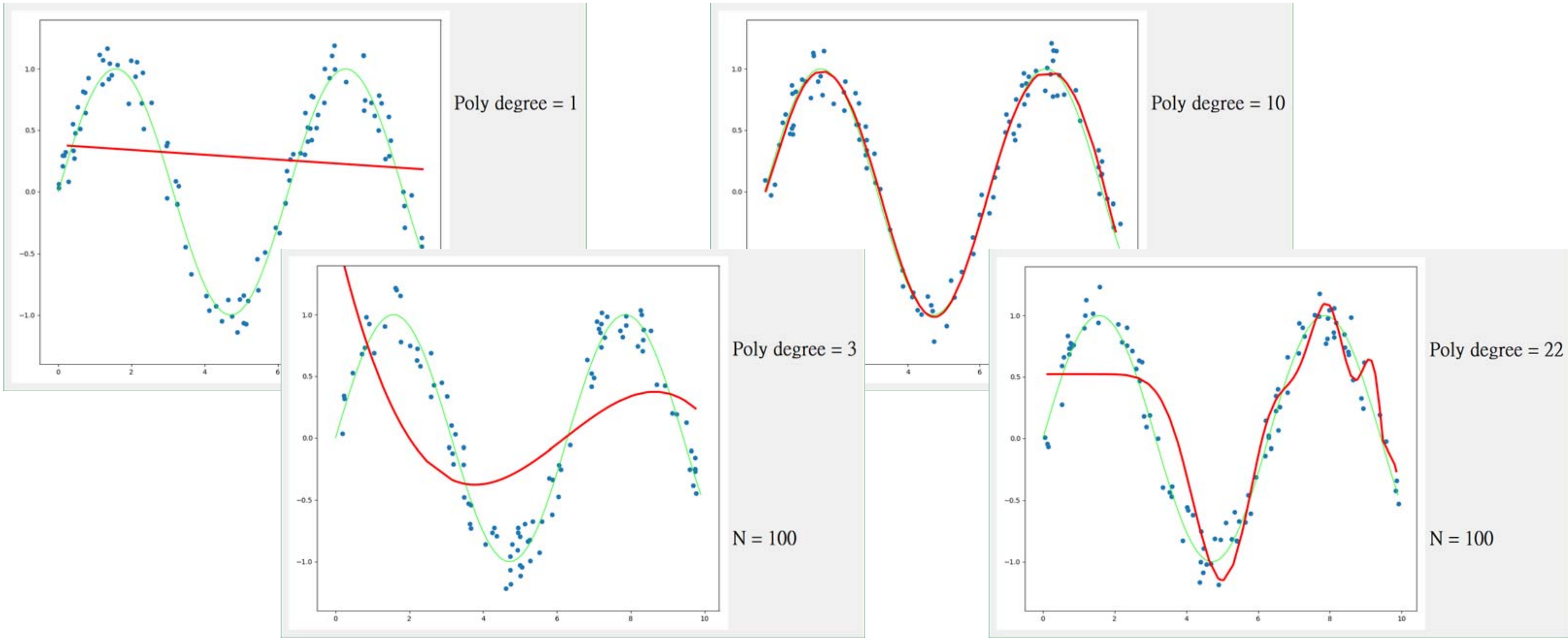
- Model :

$$y(\mathrm{x}, \mathrm{w}) = w_0 + w_1 x + w_2 x^2 + \cdots w_M x^M$$

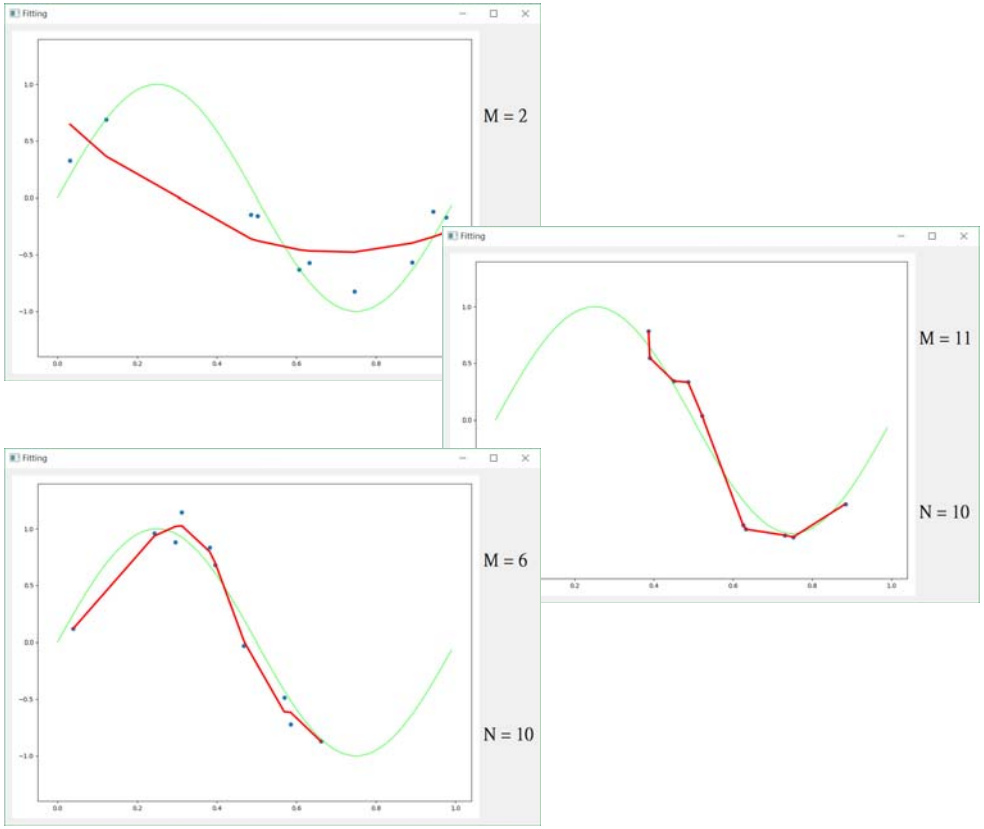
- Error function:

$$E(w) = \frac{1}{2} \sum_{n=1}^{N} \{y(x_n, w) - t_n\}^2$$

Polynomial Curve Fitting

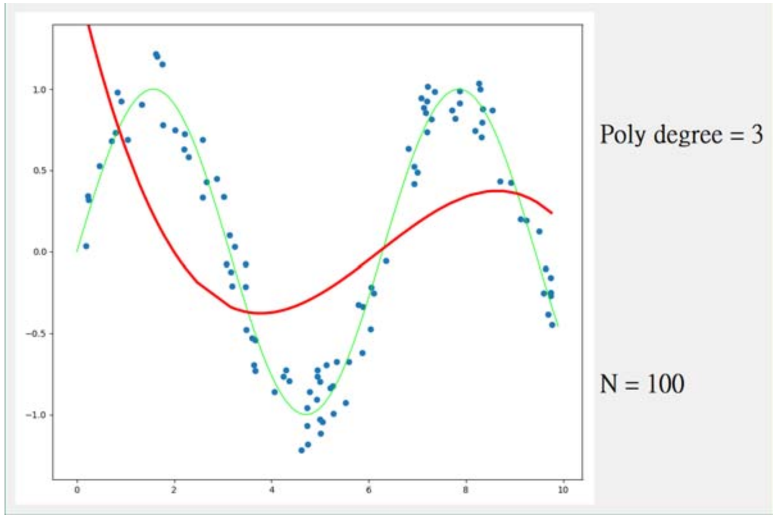


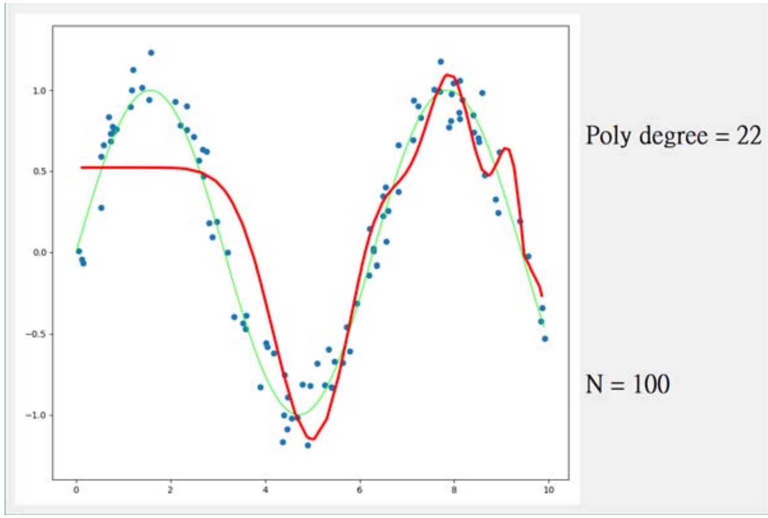
Weight



Weight	M = 2	M = 6	M = 11
w_o	0	0	2.67850417
w_1	-3.48815	100.8412	-776755.1085
w_2	2.46723	-1046.57	4502951.55
w_3		5108.034	-14066940.4
w_4		-12573.3	24327608.55
w_5		14976.32	-18752447.23
w_6		-6864.09	-6136446.715
w_7			20332026.69
w_8			-2160559.429
w_9			-20347291.69
w_{10}			17932831.65
w_{11}			-4913022.544

Over-fitting and Under-fitting

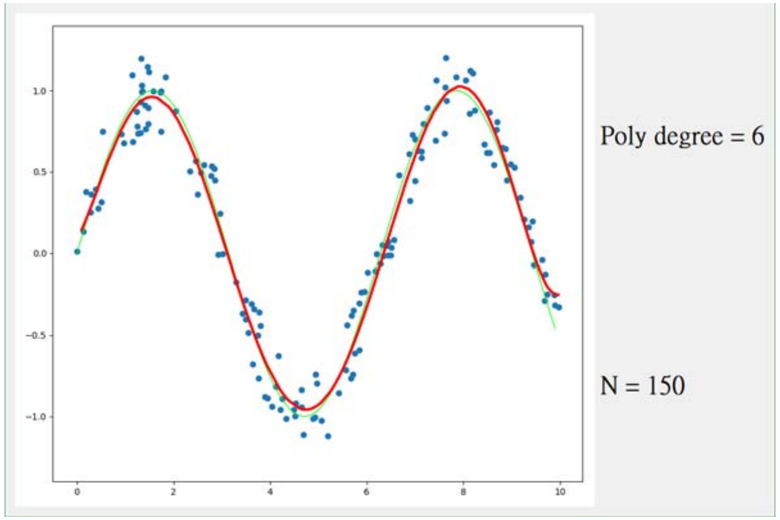


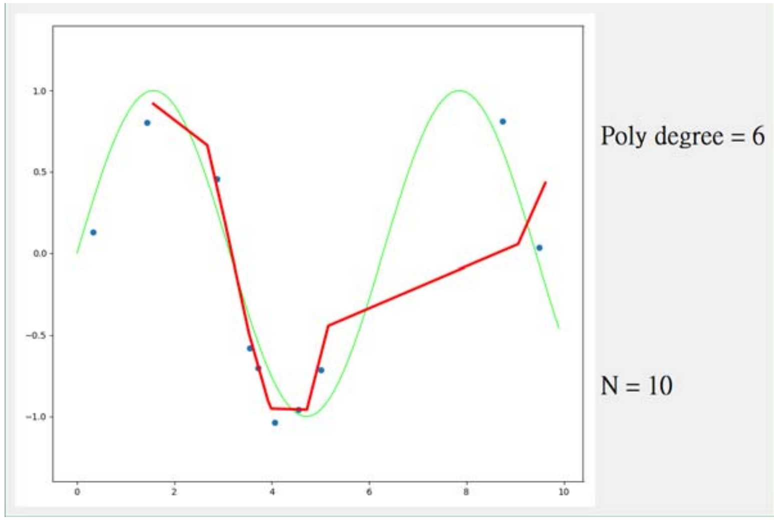


Small degree -> underfitting
Less variance

Large degree -> overfitting
Less bias

More data More accurate





More data-> Accurate

Less data -> Inaccurate

Regularization for the control of overfitting

- The coefficient governs the relative importance of the regularization term
- Ridge regression, L2 regularization

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \alpha\|w\|^2$$

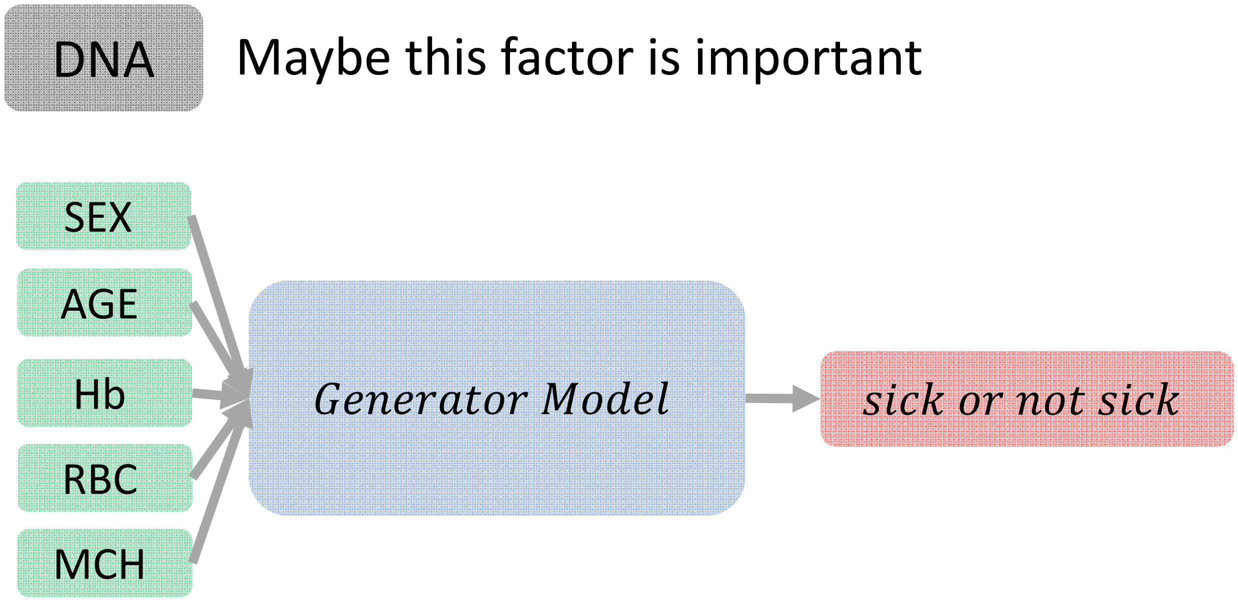
$$\|w\|^2 \equiv w^T w = w_0^2 + w_1^2 + \ldots + w_M^2$$

- Lasso regression, L1 regularization

$$E(w) = \frac{1}{2}\sum_{n=1}^{N}\{y(x_n, w) - t_n\}^2 + \alpha\|w\|^1$$

Implicit factors

◆Some data we don't get, but important.



Implicit factors

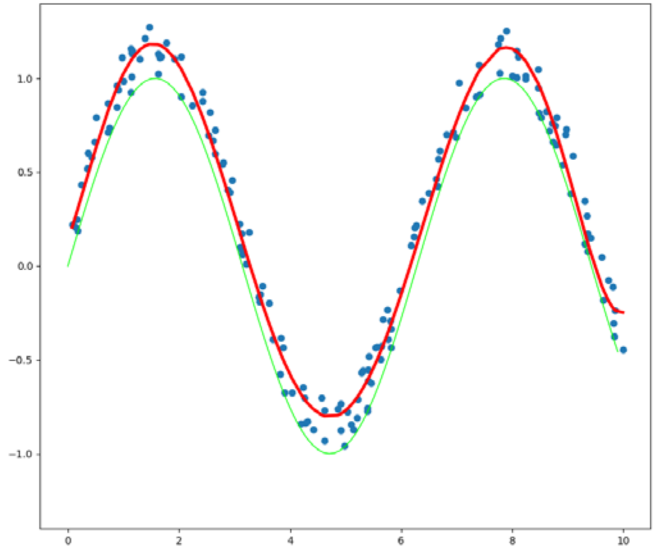
- Some data we don't get, but important.

$$f(x) = \sin(x)$$

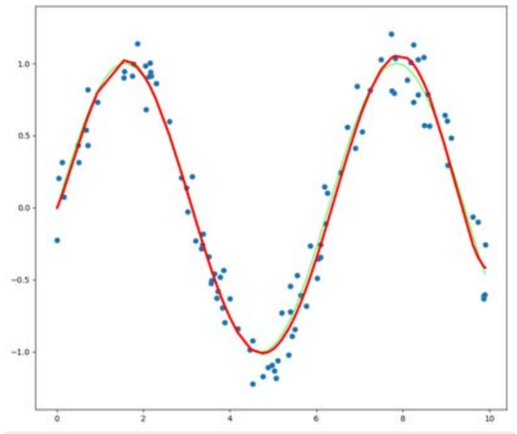
- If we don't get x_2, we can't get correct model

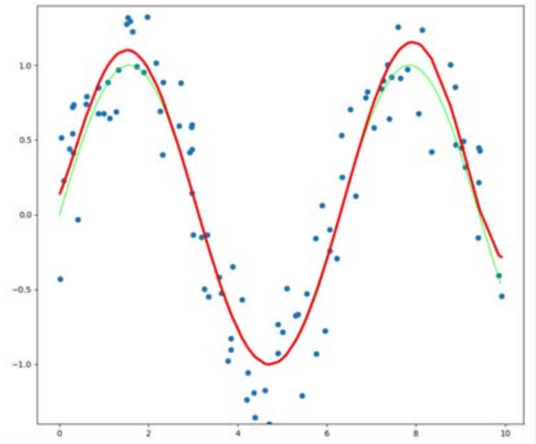
$$f(x) = \sin(x_1) + 0.3 * x_2$$

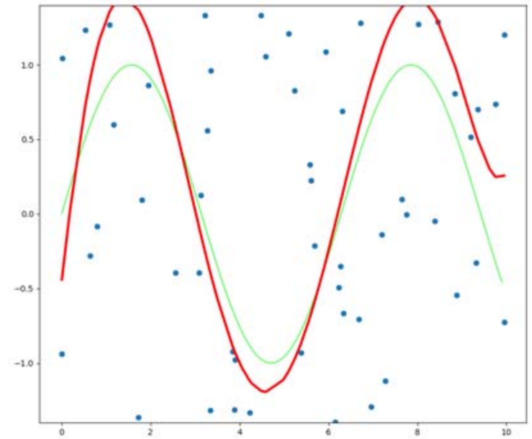
- Green line is $f(x) = \sin(x_1)$
- Red line is $f(x) = \sin(x_1) + 0.3 * x_2$



Implicit factors







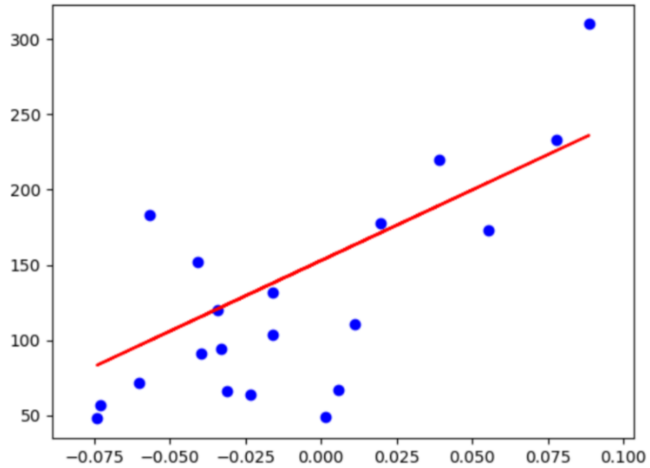
$f(x) = \sin(x_1) + 0.3 * x_2$

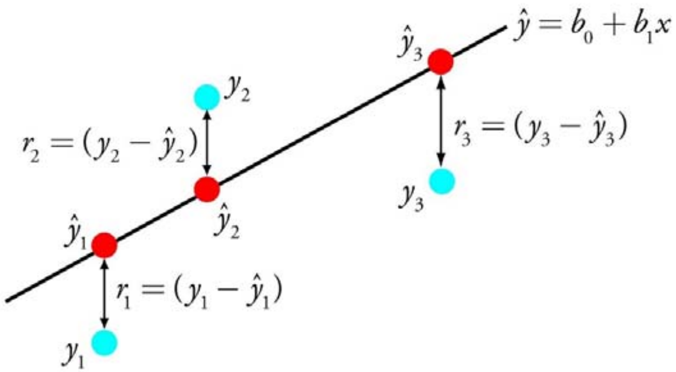
$f(x) = \sin(x_1) + 5.0 * x_2$

$f(x) = \sin(x_1) + 1.0 * x_2$

3 Scikit-Learn

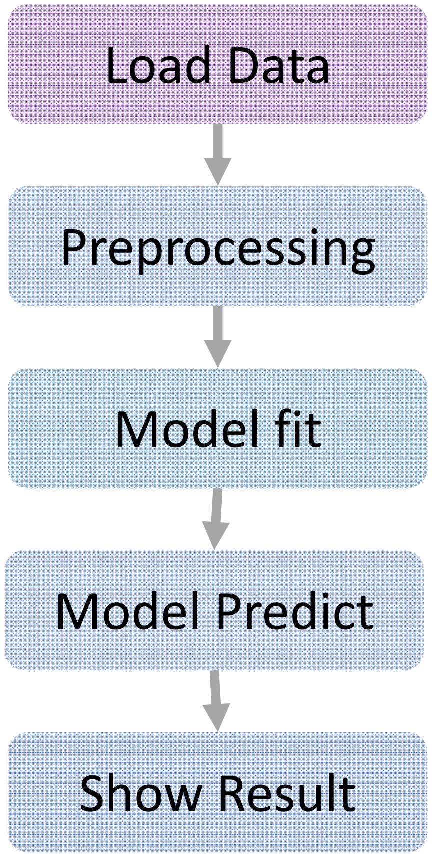
Linear Regression Example






```
from sklearn import linear_model, datasets
import numpy as np
import matplotlib.pyplot as plt

# Load the diabetes dataset
diabetes = datasets.load_diabetes()

# Use one feature
x = diabetes.data[:, np.newaxis, 2]

# split dataset
x_train = x[:-20]
x_test = x[-20:]
y_train = diabetes.target[:-20]
y_test = diabetes.target[-20:]

regr = linear_model.LinearRegression()
regr.fit(x_train, y_train)
y_pred = regr.predict(x_test)

plt.scatter(x_test, y_test,  color='blue')
plt.plot(x_test, y_pred, color='red')

plt.show()
```

How to use?



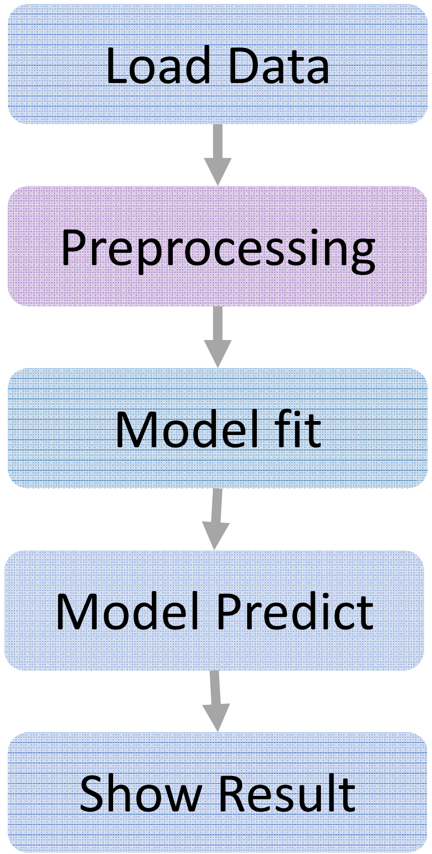
- diabetes dataset from sklearn

```
diabetes = datasets.load_diabetes()
```

- If use csv file

```
import pandas as pd
df = pd.read_csv('file.csv')
```

How to use?



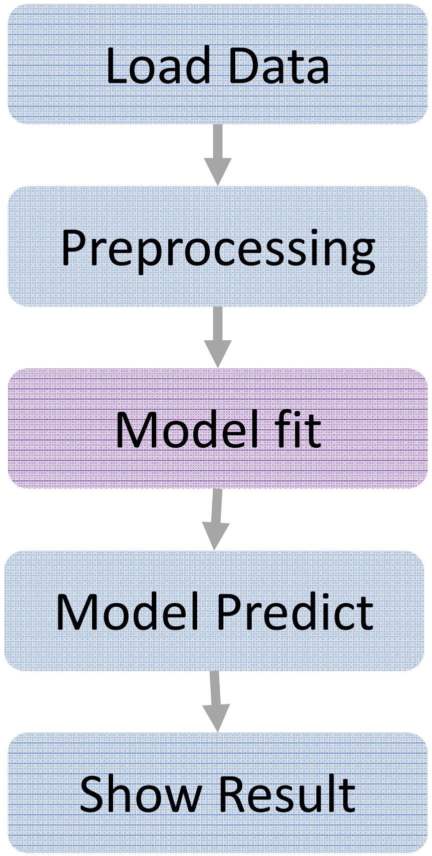
- Split dataset

```
x_train = x[:-20]
x_test = x[-20:]
y_train = diabetes.target[:-20]
y_test = diabetes.target[-20:]
```

- Can use normalize or other method

```
from sklearn import preprocessing
x = preprocessing.scale(x)
y = preprocessing.scale(y)
```

How to use?



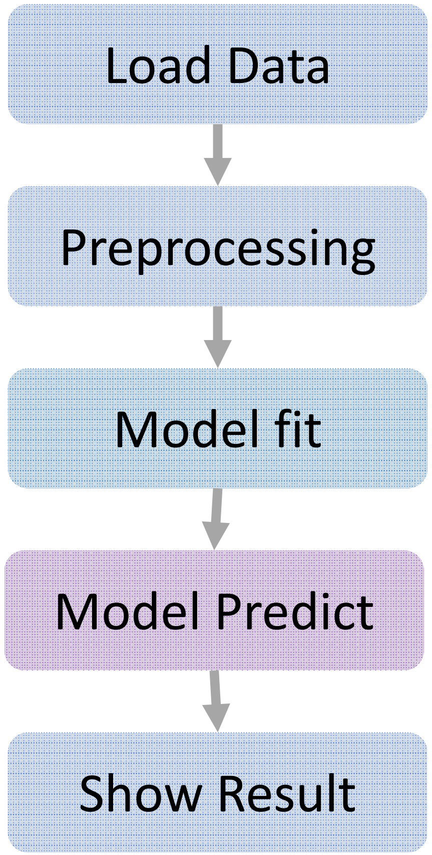
◈Use linear regression model from sklearn

regr = linear_model.LinearRegression()

◈Use this model to train

regr.fit(x_train, y_train)

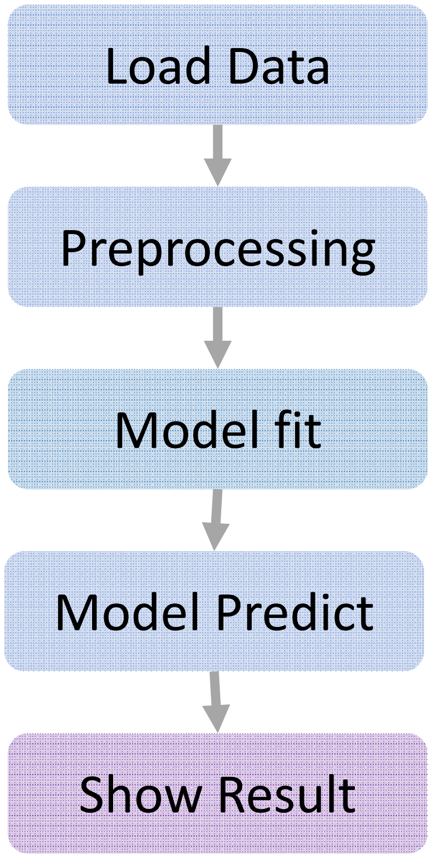
How to use?



- Get predict

```
y_pred = regr.predict(x_test)
```

How to use?



◈Use matplotlib

```
import matplotlib.pyplot as plt
plt.scatter(x_test, y_test,  color='blue')
plt.plot(x_test, y_pred, color='red')
plt.show()
```

How to use?

Preprocessing	Preprocessor.fit_transform()
↓	
Model fit	Model.fit(train_x, train_y)
↓	
Model Predict	Model.predict(test_x)

4

Solving extrema of functions

How to solve function's minimum/maximum

- Suppose we want to find the parameter w to minimize $L(w)$
  - The most direct naïve approach is to find many w candidates.
  - What if we have multiple parameters to solve?

$$L(w_1, w_2, w_3, \cdots, w_{100})$$

    - the number of candidates will be very huge. Ex: 10^{100}
  - the actual weights to train could be up to million.

$$w \to \boxed{L(w)} \to L$$

Gradient Descent

$$\arg\min_{w} L(w)$$

- Suppose we want to find w to minimize $L(w)$
  - We don't know the exact form of $L(w)$

$w' = w_0 \boxed{+ |\Delta w|}$

$w = w_0 \rightarrow L(w) \rightarrow L_0 = L(w_0)$

Start with a initial solution w_0

1 $L' = L_0 + |\Delta L|$ Slope + :

$\boxed{w \leftarrow w_0 - |\Delta w|}$

Slope 0:

3 $L' = L_0 + 0$

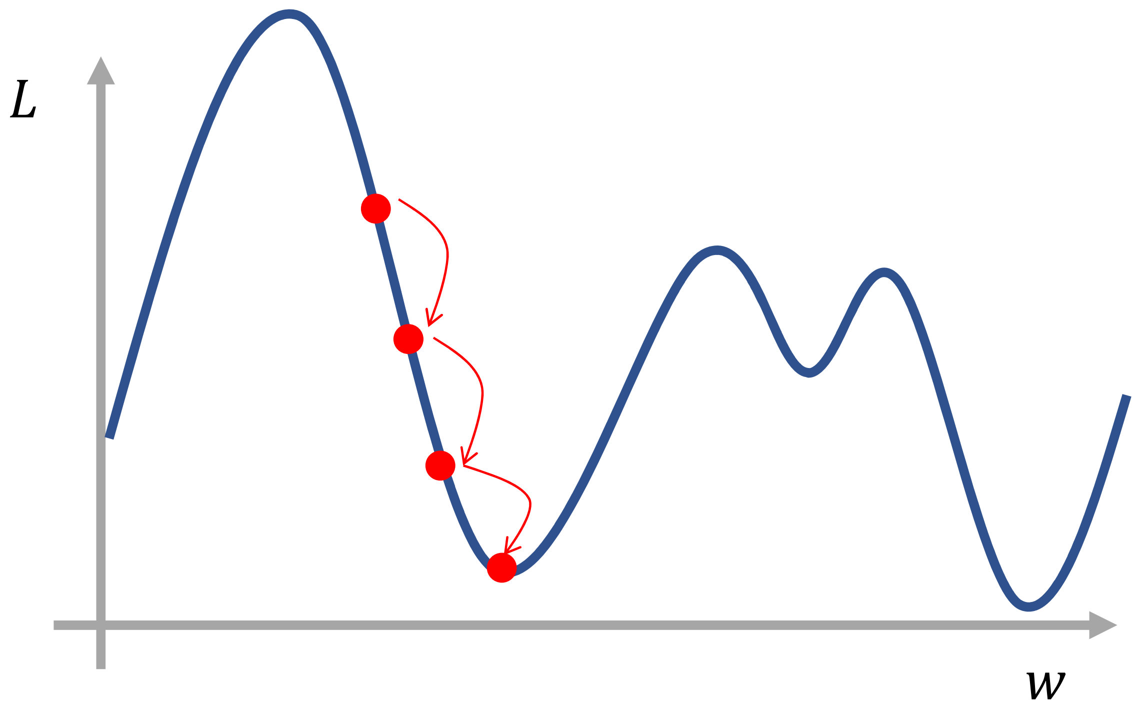
$\boxed{optimized\ w}$

2 $L' = L_0 - |\Delta L|$

Slope - :

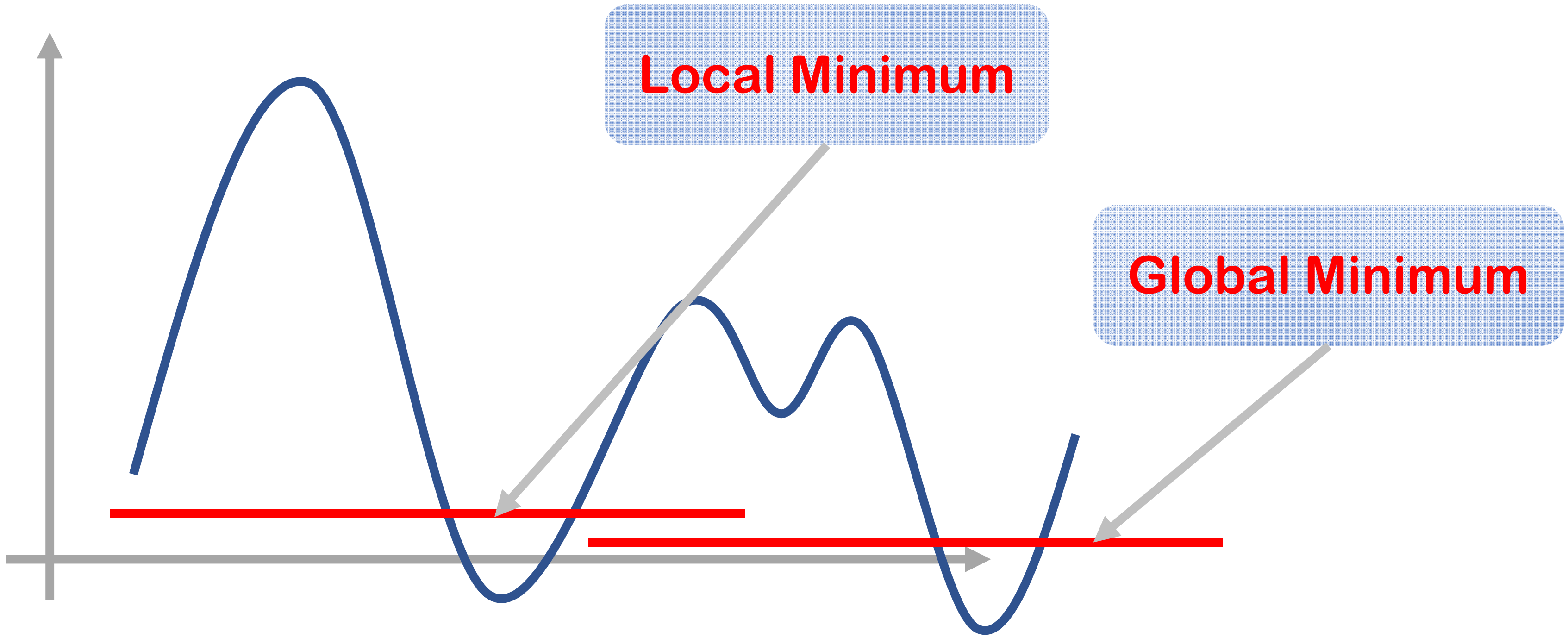
$\boxed{w \leftarrow w_0 + |\Delta w|}$

Gradient Descent



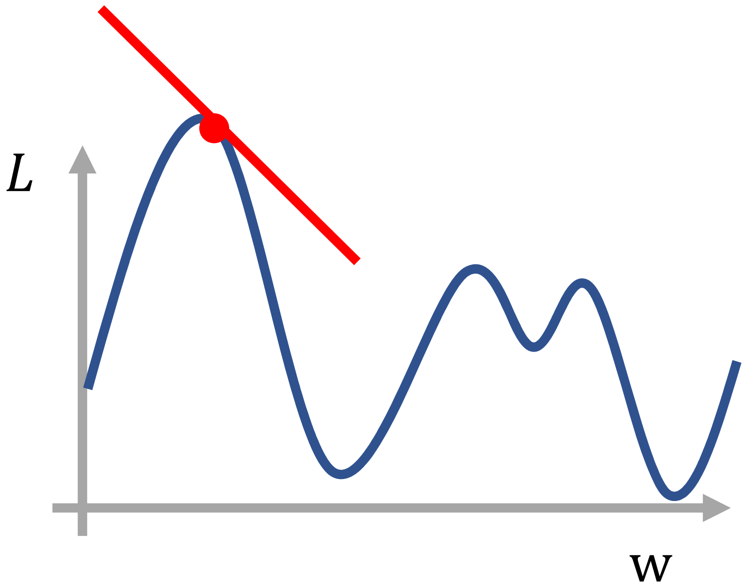
Problem of Gradient Descent

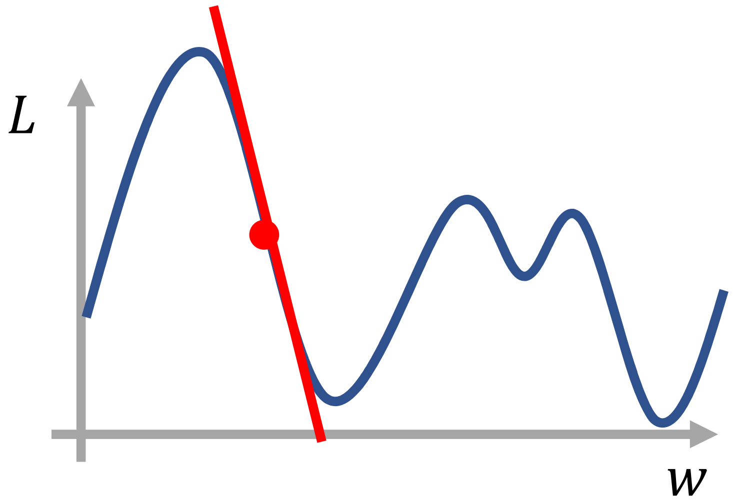
- Stuck at the local minimum
- Oscillate around the minimum point



Differential equation

$$L(w) = aw^n \Rightarrow \frac{\partial L}{\partial w} = \frac{\partial (aw^n)}{\partial w} = naw^{(n-1)}$$

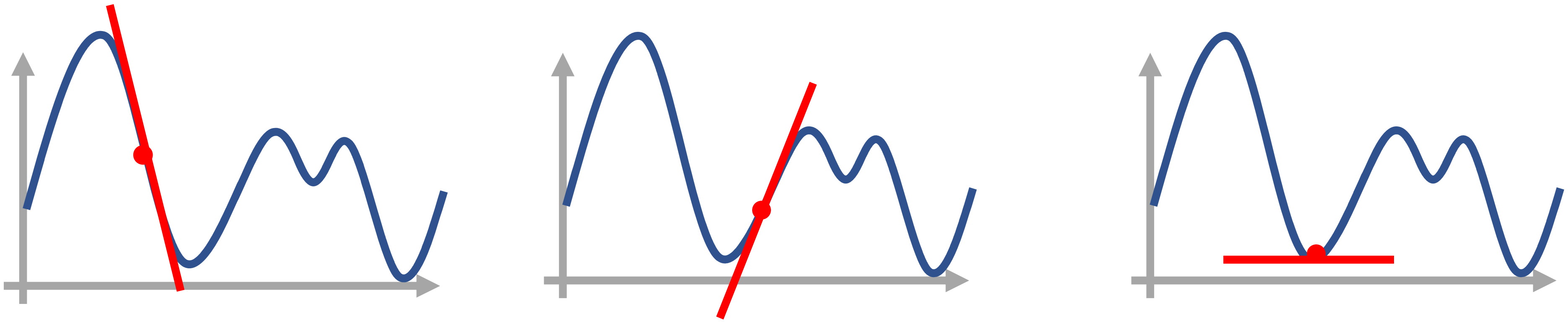




Adjustment rate η

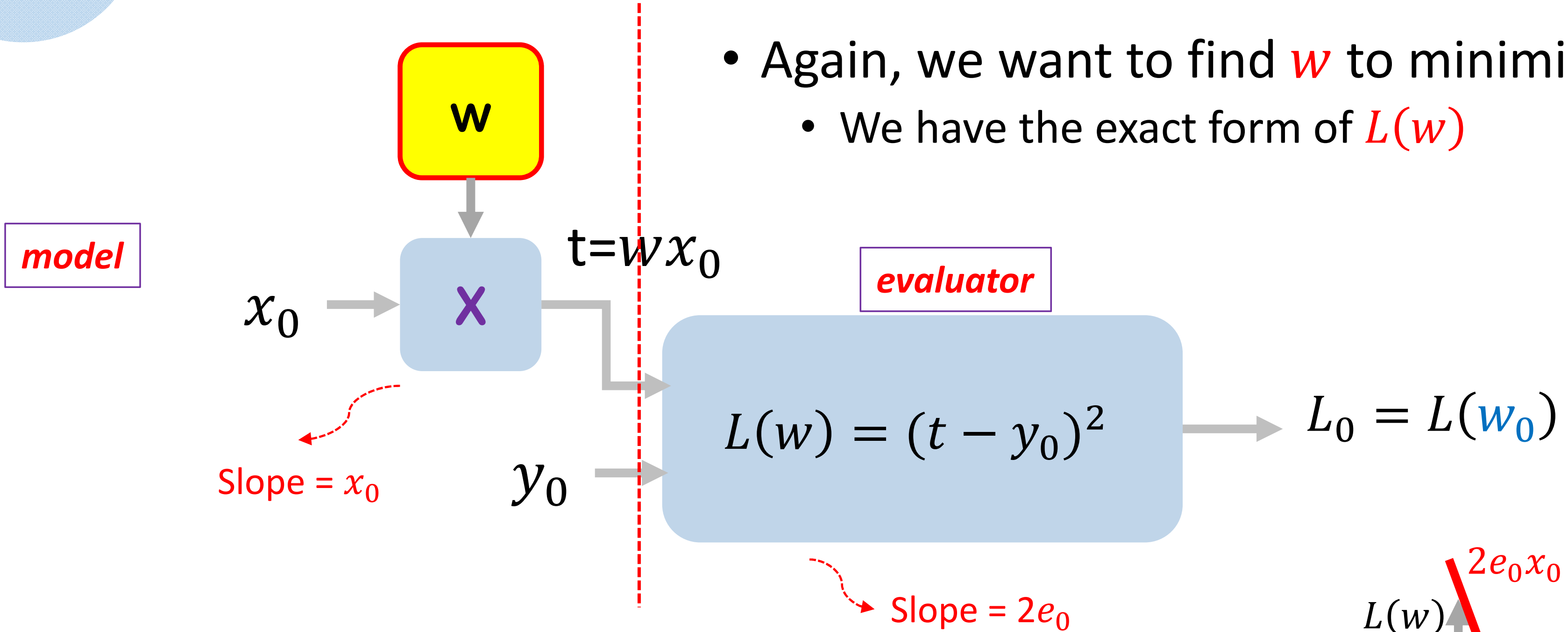
$$slope = -\eta \frac{\partial L}{\partial w}$$

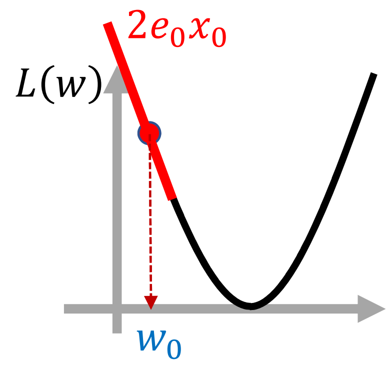
$$\frac{\partial L}{\partial w} = -3 \quad \xrightarrow{-\eta \frac{\partial L}{\partial w} = 3\eta} \quad \frac{\partial L}{\partial w} = 1 \quad \xrightarrow{-\eta \frac{\partial L}{\partial w} = -\eta} \quad \frac{\partial L}{\partial w} = 0$$



Gradient Descent for linear regression

- Again, we want to find w to minimize $L(w)$
  - We have the exact form of $L(w)$





Slope = $\left.\frac{\partial L(w)}{\partial w}\right|_{w=w_0} = 2(t - y_0)x_0 = 2(wx_0 - y_0)x_0 = 2e_0x_0$

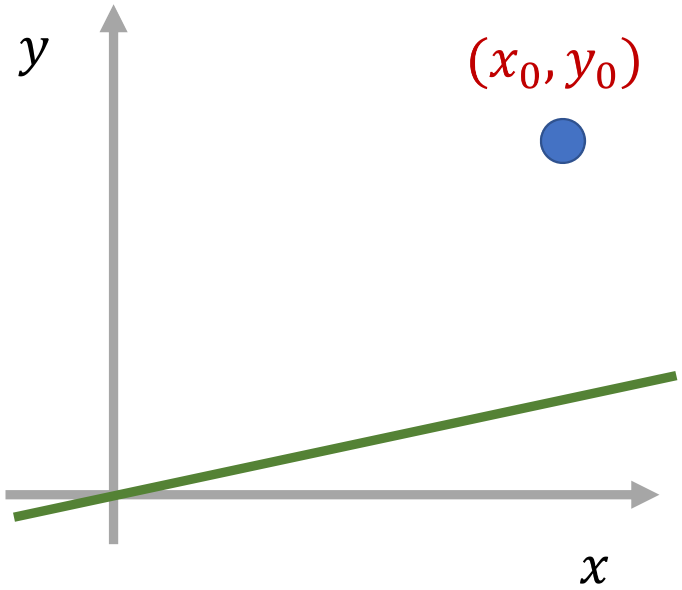
◇If $e_0 < 0$, increase w in order to increase t

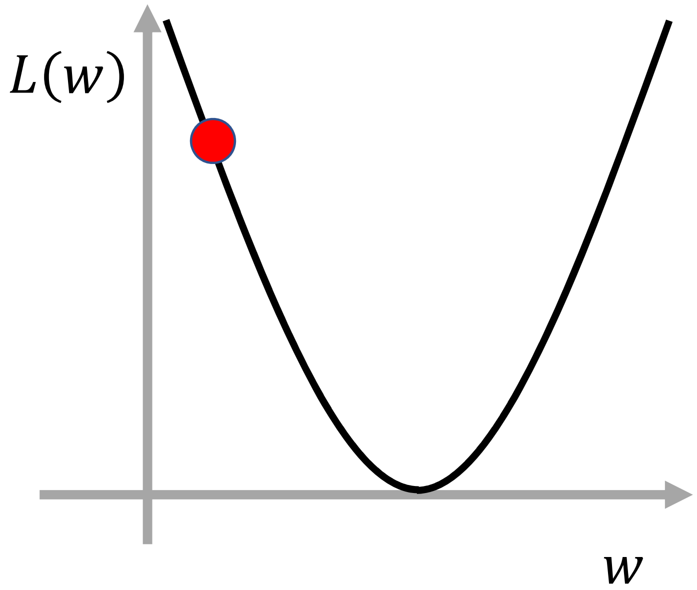
Gradient Descent

Model

Epoch = 0

Loss



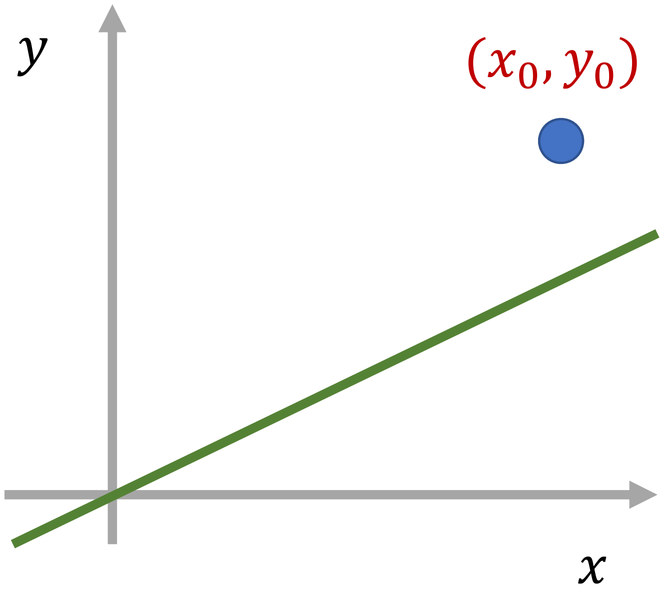


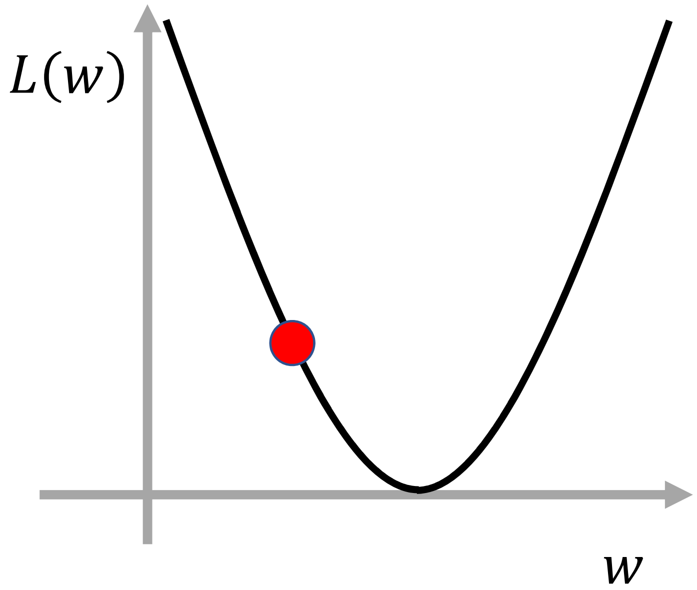
Gradient Descent

Model

Epoch = 10

Loss



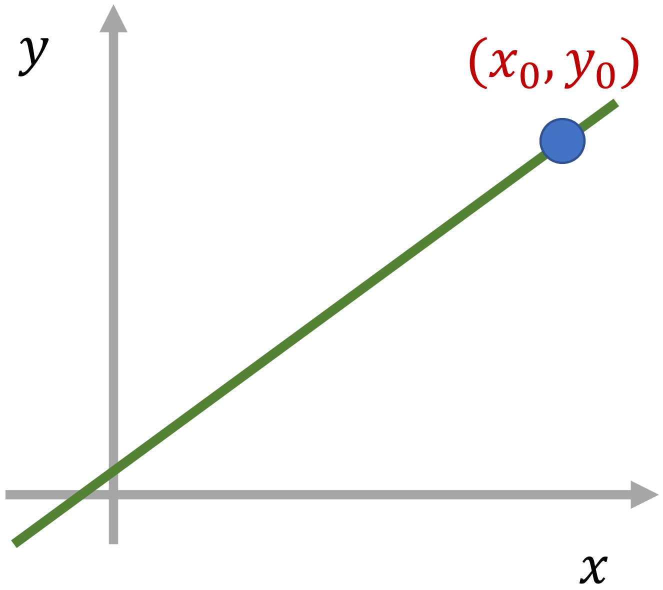


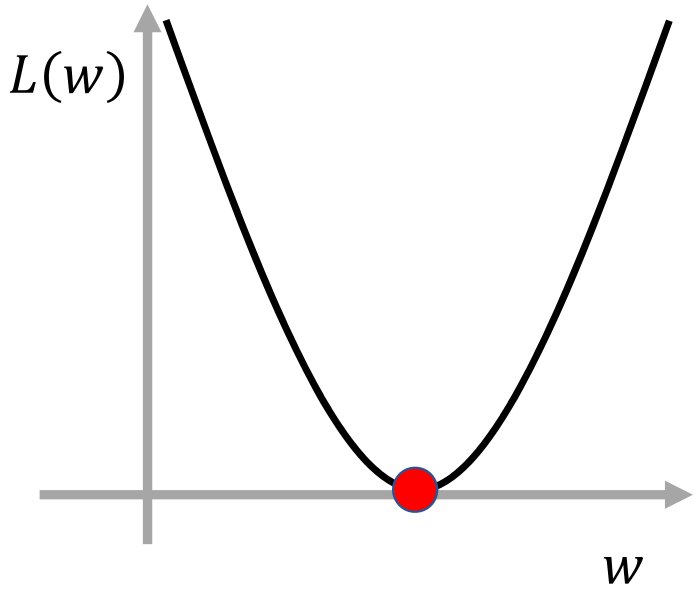
Gradient Descent

Model

Epoch = 100

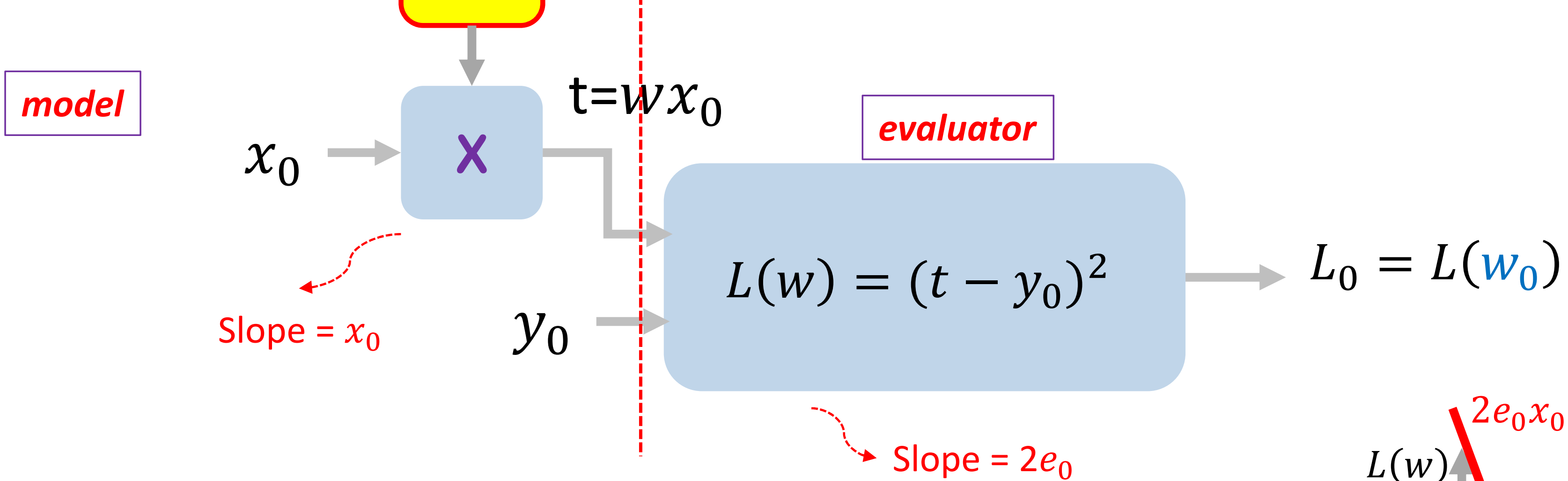
Loss

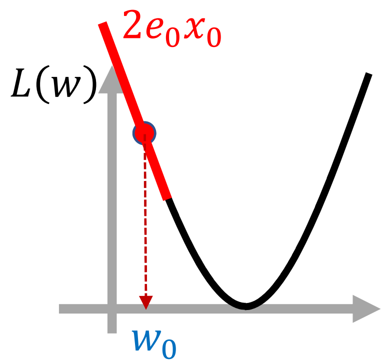




Gradient Descent for multi-variable linear regression (v)

- Again, we want to find w to minimize $L(w)$
  - We have the exact form of $L(w)$



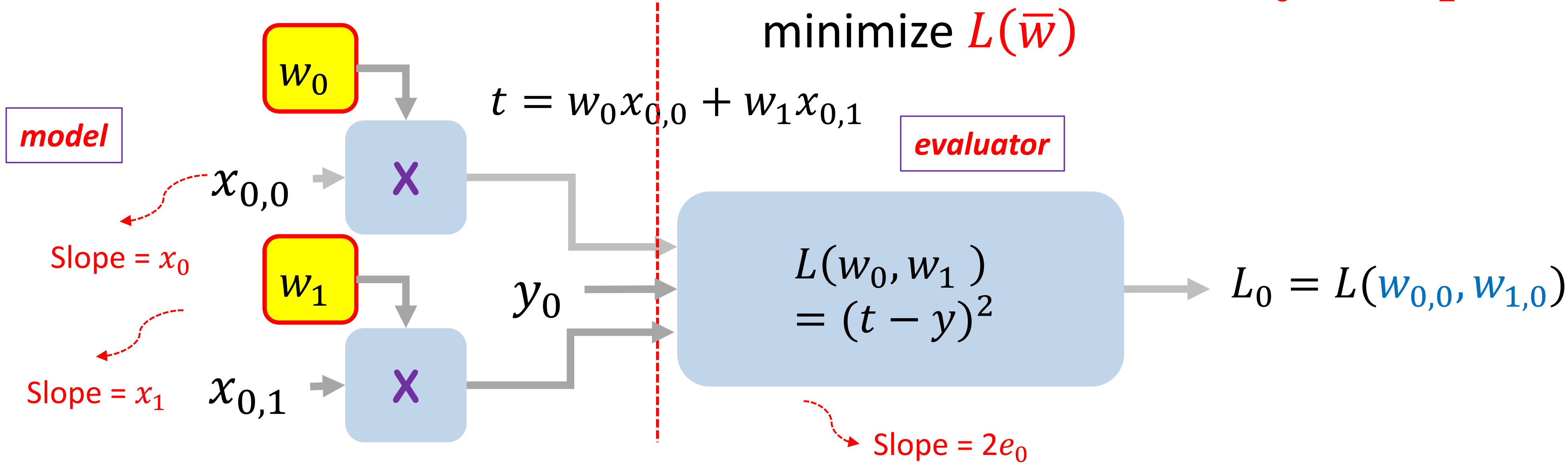


Slope = $\left.\frac{\partial L(w)}{\partial w}\right|_{w=w_0} = 2(t - y_0)x_0 = 2(wx_0 - y_0)x_0 = 2e_0x_0$

◇If $e_0 < 0$, increase w in order to increase t

Gradient Descent for multi-variable linear regression

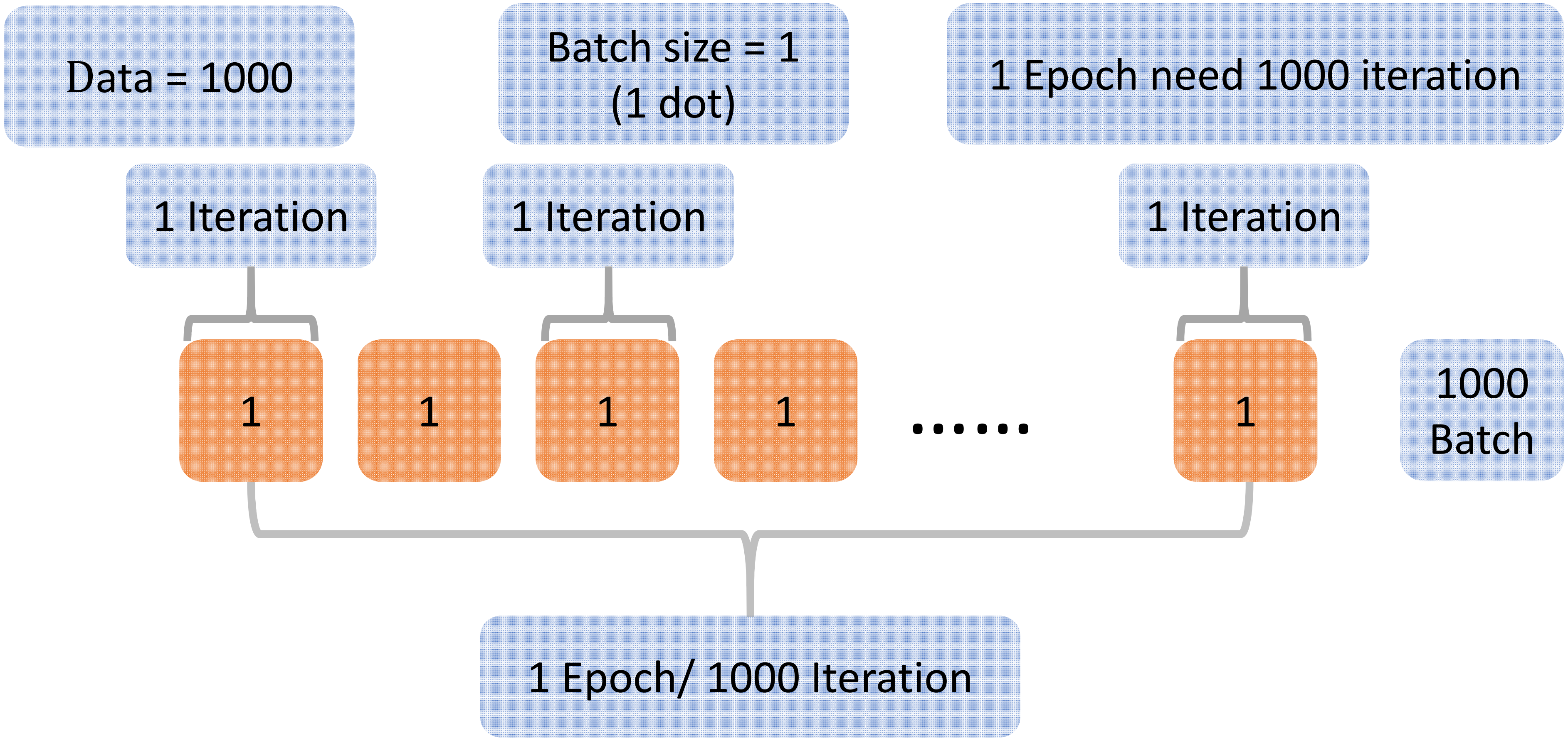
- Here, we want to find w_0 and w_1 to minimize $L(\overline{w})$



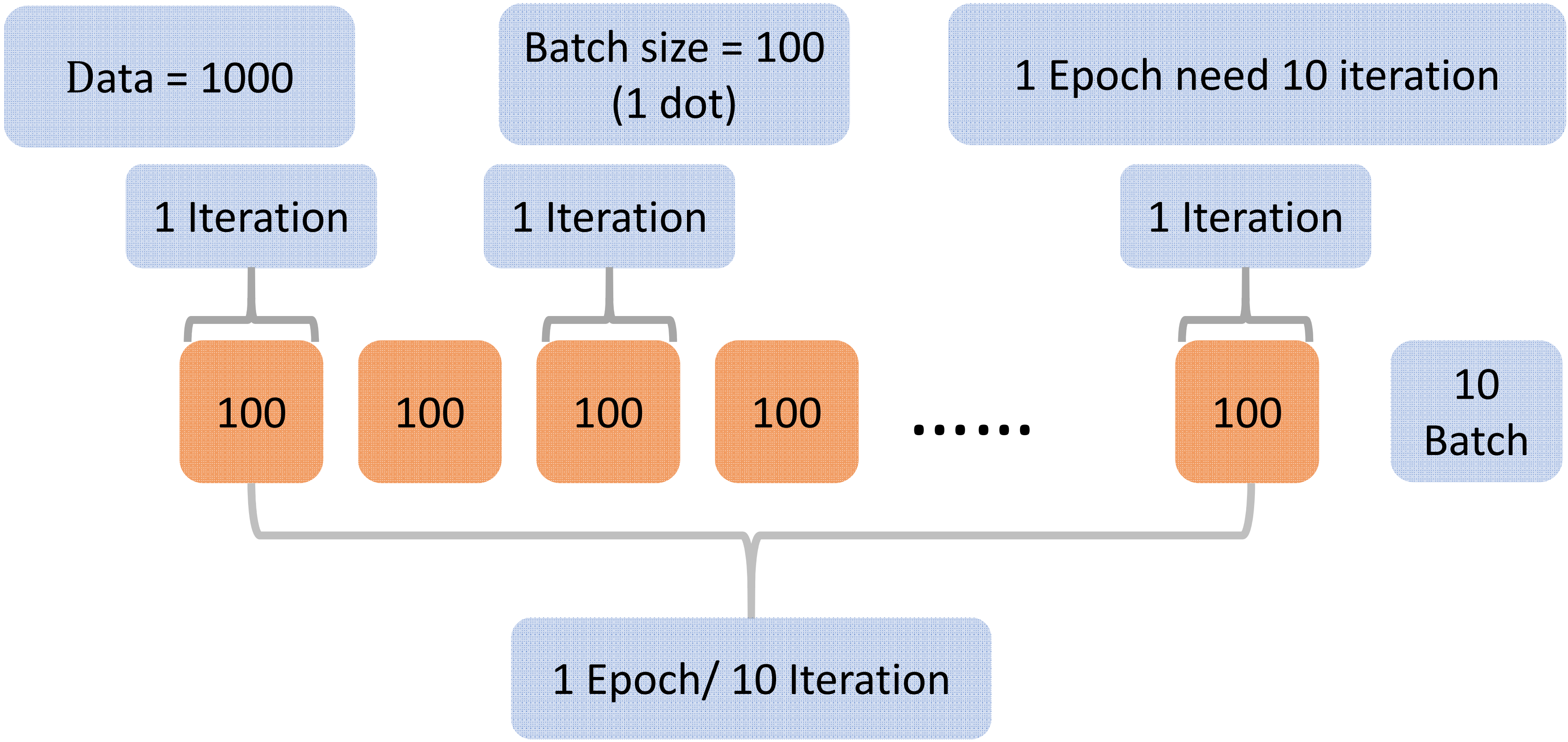
Slope = $\left.\frac{\partial \mathrm{L}(w_0, w_1)}{\partial w_0}\right|_{w_0, = w_{0,0}} = 2(t - y_0)x_{0,0} = 2\left(w_0 x_{0,0} + w_1 x_{0,1} - y_0\right)x_{0,0} = 2e_0 x_{0,0}$

◈ If $e_0 < 0$, increase w_0 in order to increase t

Batch & Epoch & Iteration

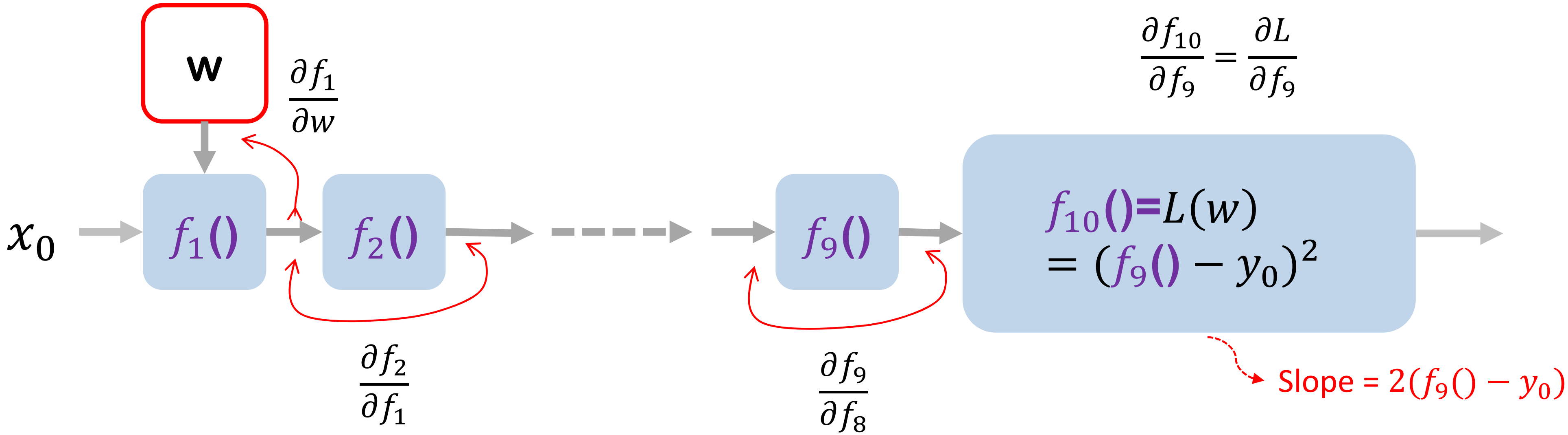


Batch & Epoch



Data = 1000
Batch size = 100
(1 dot)
1 Epoch need 10 iteration
1 Iteration
1 Iteration
1 Iteration
100
100
100
100
……
100
10
Batch
1 Epoch/ 10 Iteration

Back propagation



Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial f_1}{\partial w} \times \frac{\partial f_2}{\partial f_1} \times \cdots \times \frac{\partial L}{\partial f_9}$

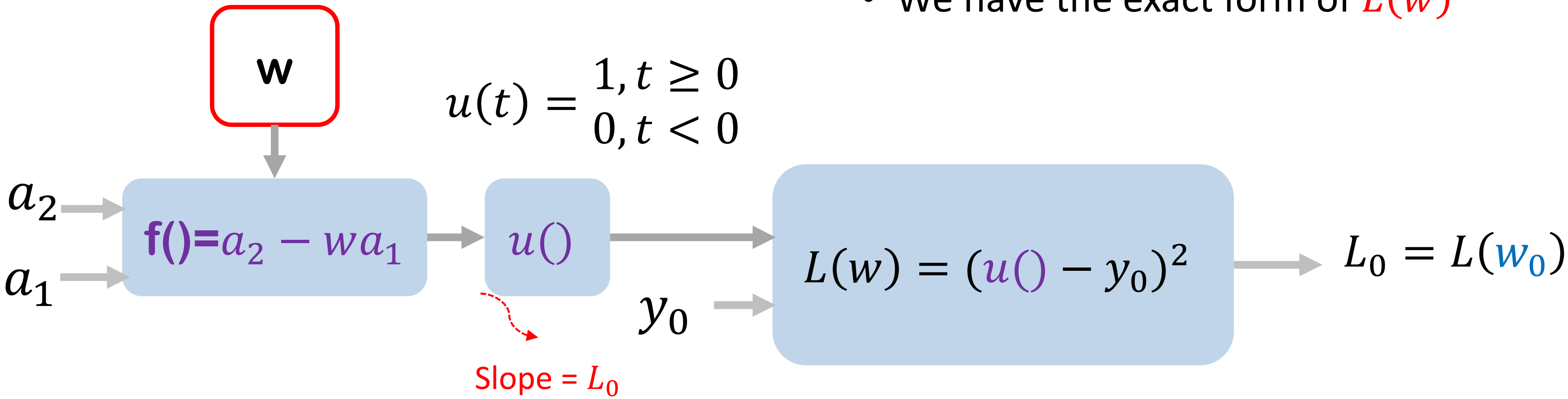
Gradient vanish or explode

Chain rule: $\frac{\partial L}{\partial w} = \frac{\partial f_1}{\partial w} \times \frac{\partial f_2}{\partial f_1} \times \cdots \times \frac{\partial L}{\partial f_9}$

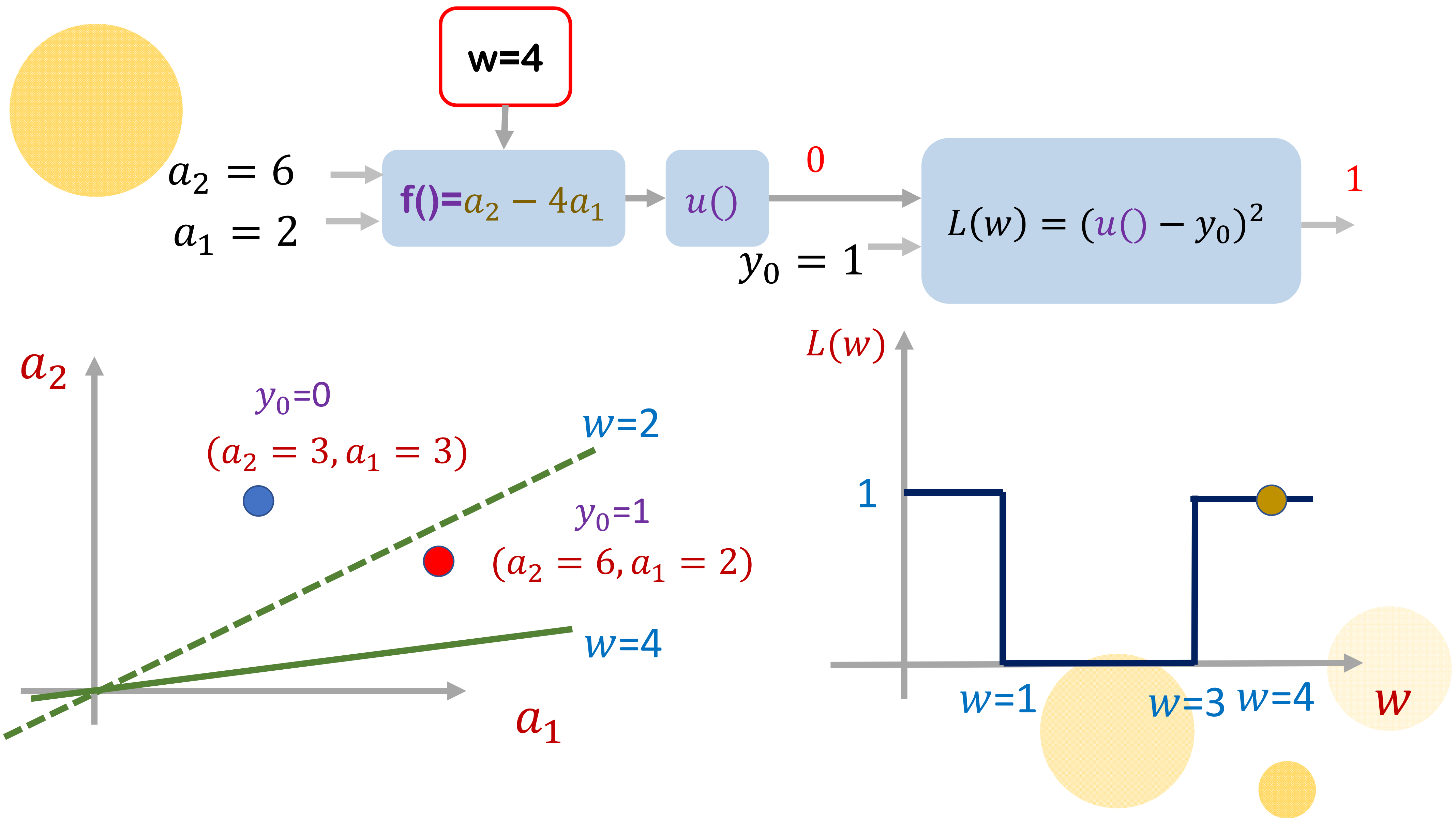
- What if $\frac{\partial f_{i+1}}{\partial f_i} \ll 1$ ⟹ Gradient vanish
- What if $\frac{\partial f_{i+1}}{\partial f_i} \gg 1$ ⟹ Gradient explode

Gradient Descent for logistic regression

- Again, we want to find w to minimize $L(w)$
  - We have the exact form of $L(w)$

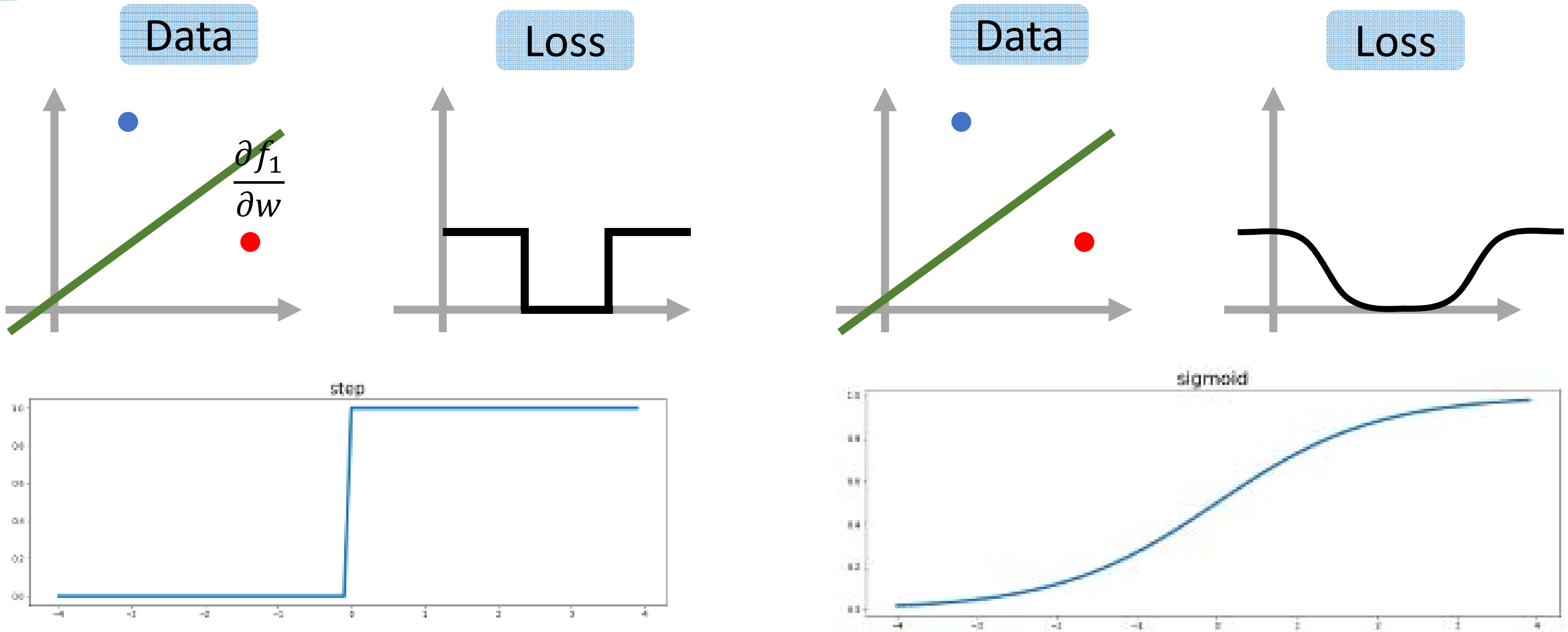


$$\left.\frac{\partial \mathrm{L(w)}}{\partial \mathrm{w}}\right|_{w=w_0} = 2(u() - y_0) \times u'() \times x_0$$



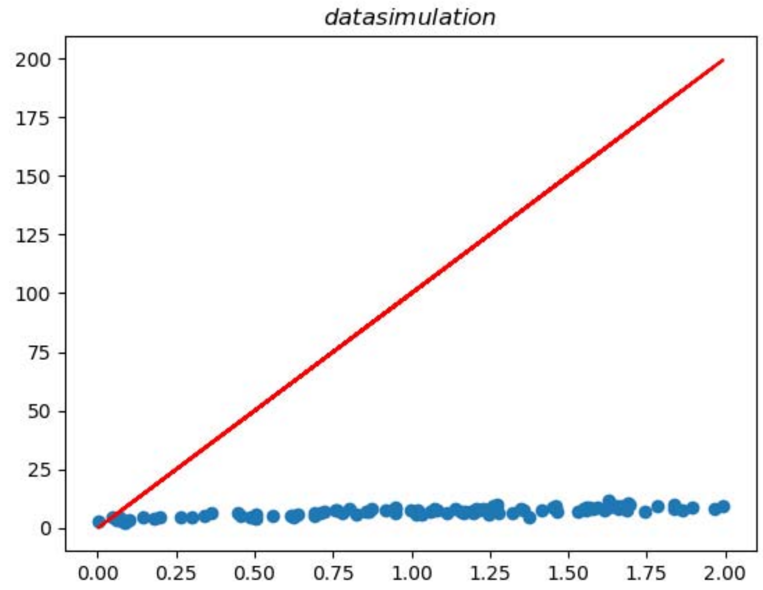
w=4
$a_2 = 6$
$a_1 = 2$
f()=$a_2 - 4a_1$
$u()$
0
$y_0 = 1$
$L(w) = (u() - y_0)^2$
1
a_2
y_0=0
$(a_2 = 3, a_1 = 3)$
w=2
y_0=1
$(a_2 = 6, a_1 = 2)$
w=4
a_1
$L(w)$
1
w=1
w=3
w=4
w

Activation function

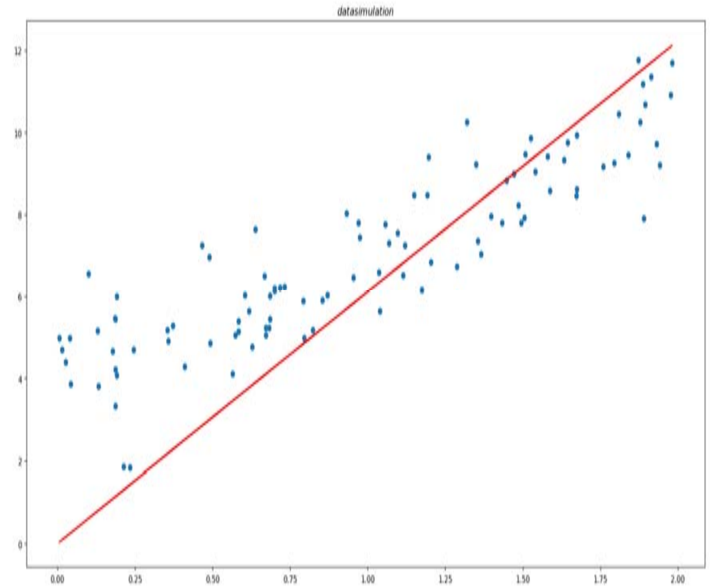


Gradient Descent

Before



After



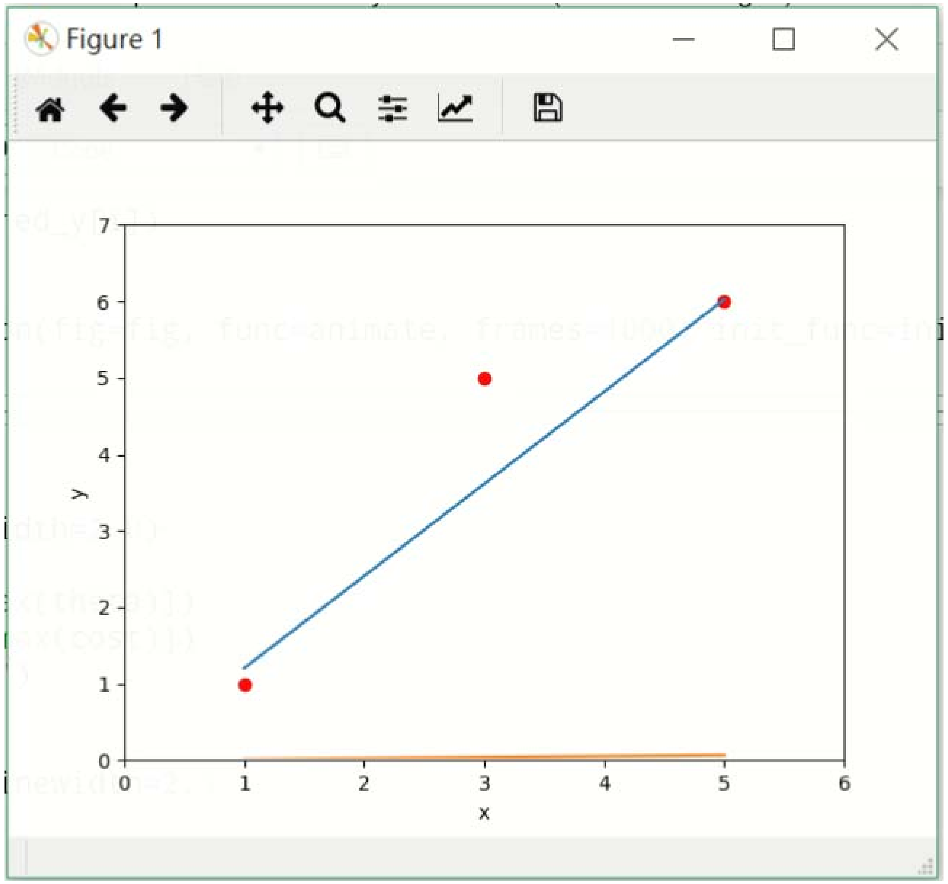
Training process

◇Use Linear Regression.

◇Three data

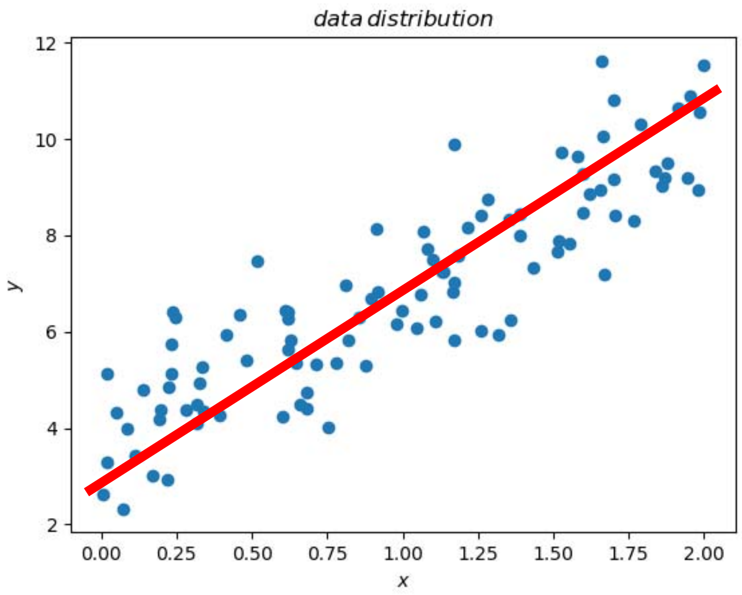
◇Learning = 0.001

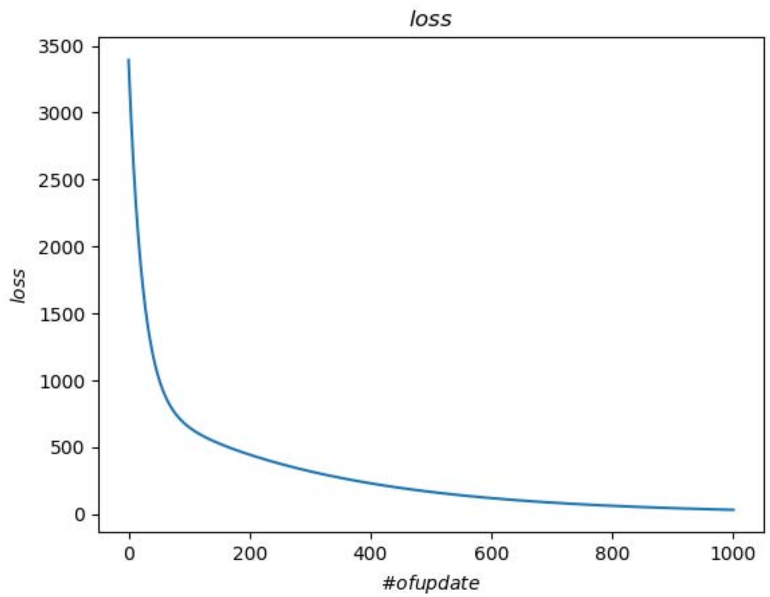
◇Epoch = 1000



Linear Regression Training process

◈Data and loss





Linear Regression Training process

◈Training process.