深度學習Deep Learning (3)

112-1

朱學亭老師

課程大綱

- W1-課程介紹/Introduction
- W2-Python/Colab and TensorFlow
- W3-神經網路/Numpy/Pandas
- W4-機器學習/Sklearn/PyTorch
- W5-CNN/Encoder–Decoder /GAN
- W6-RNN
- W7-Transformer
- W8-Computer Vision
- W9-Midterm presentation
- W10-Seq2Seq/Word2Vec
- W11-BERT
- W12-LLM
- W13-NLP1
- W14-NLP2
- W15-Audio Analysis
- W16-AICUP 1
- W17-AICUP 2
- W18-Final presentation

W3大綱

- W2: Python & Colab
- W2: TensorFlow
- W3: 神經網路模型概念
- W3: PyTorch

https://line.me/R/ti/g/mOuuu4F75E

(1) AICUP

秋季賽 2023.9~12

https://www.aicup.tw/copy-of-ai-cup-2023-1

隱私保護與醫學數據標準化競賽時程

⏰ 隱私保護與醫學數據標準化競賽時程 ⏰

\ 🔔 隱私保護與醫學數據標準化競賽時程公佈囉 /

報名時間 👉 即日起 ~ 12/01

報名方式 👉 AI CUP 報名系統 & CodaLab 註冊報名

🕰 報名時程 🕰
2023/09/18 00:00:00 ~ 2023/12/01 23:59:59
開放報名及組隊及上傳資料使用同意書

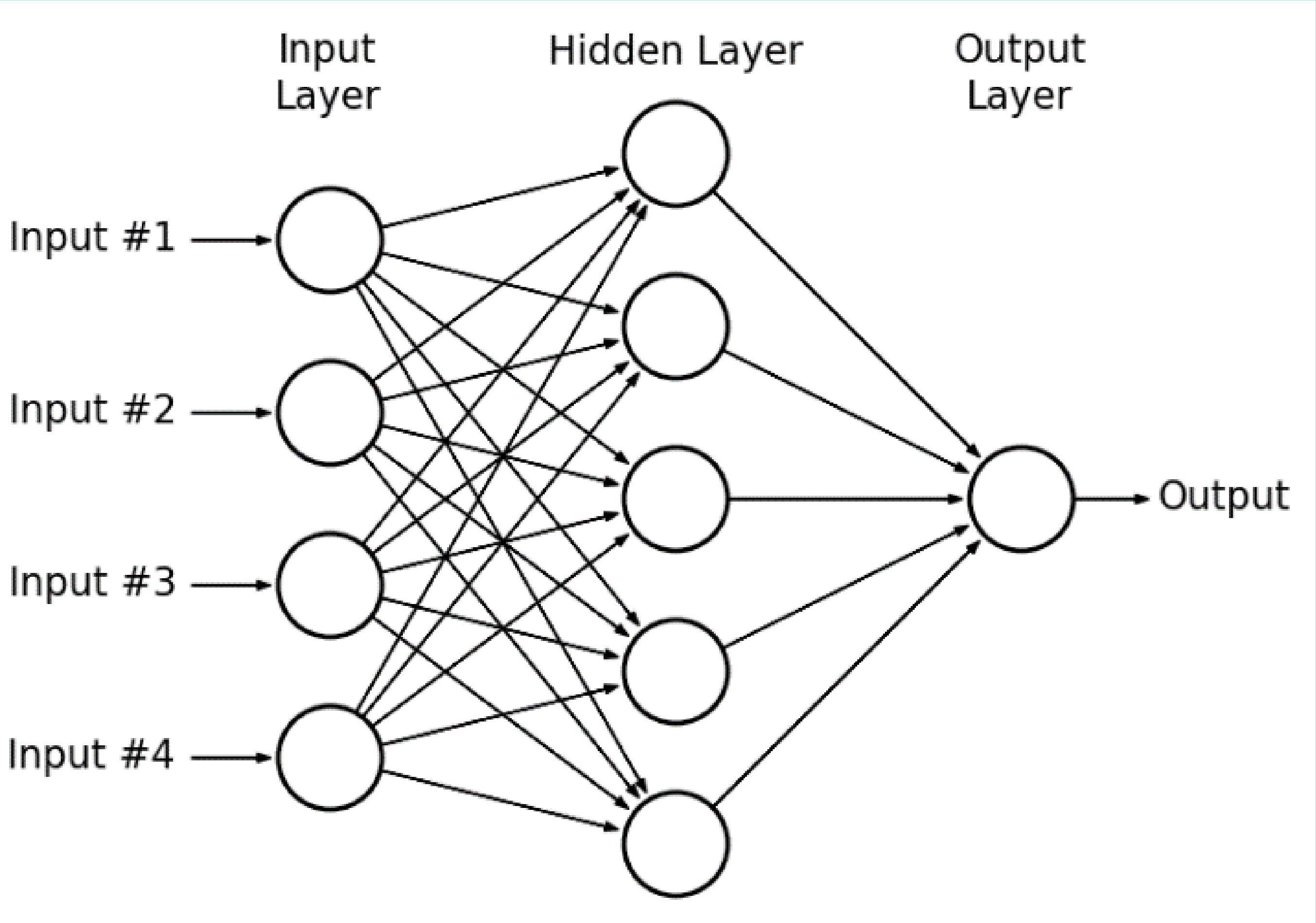
📚 第一部份訓練集下載 📚
2023/09/25 12:00:00 ~ 2023/12/01 23:59:59
訓練集(共 1,120 篇)下載

📚 第二部分訓練集下載 📚
2023/10/13 12:00:00 ~ 2023/12/01 23:59:59
額外的訓練集(共 614 篇)下載

https://reurl.cc/A0O76Z

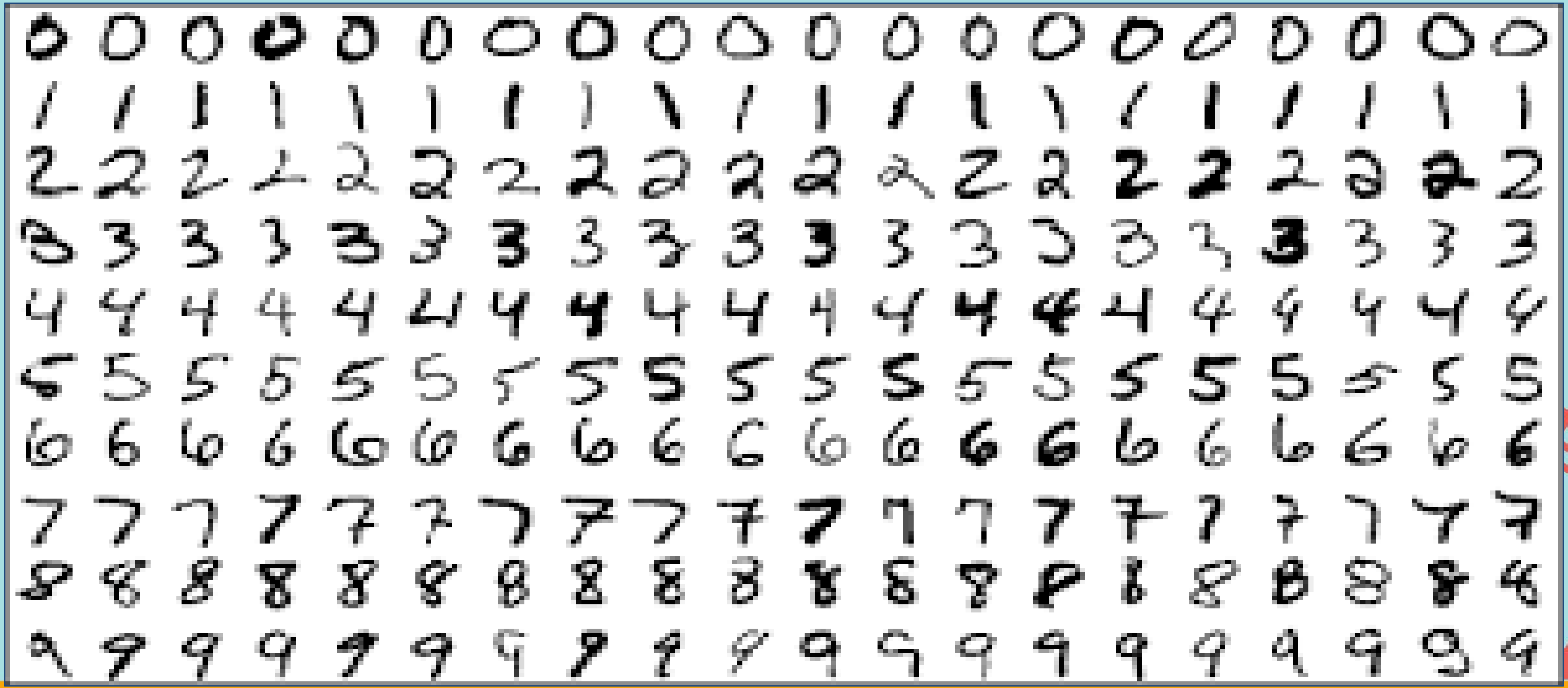
(2) 神經網路模型概念

神經網路模型

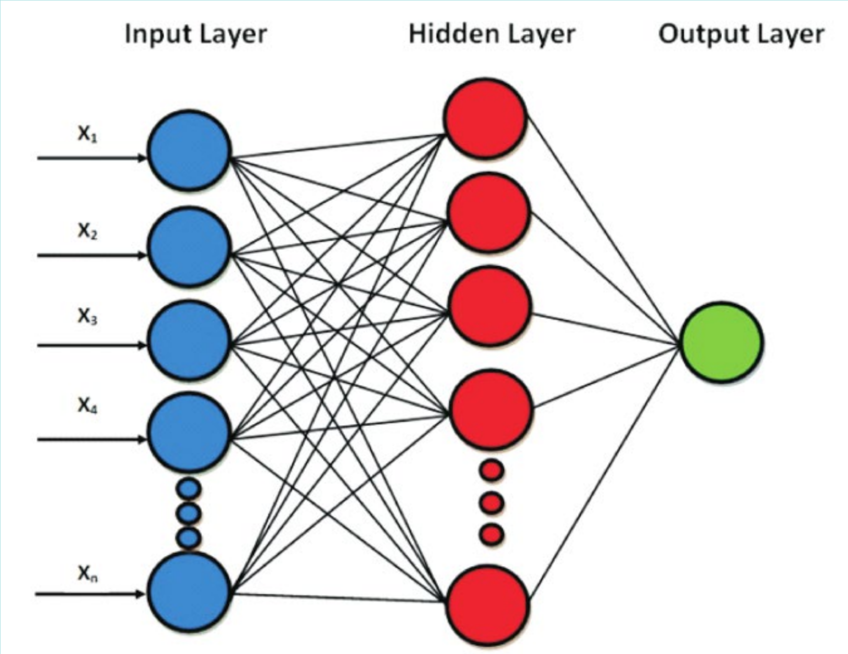

Input Layer
Hidden Layer
Output Layer
Input #1
Input #2
Input #3
Input #4
Output

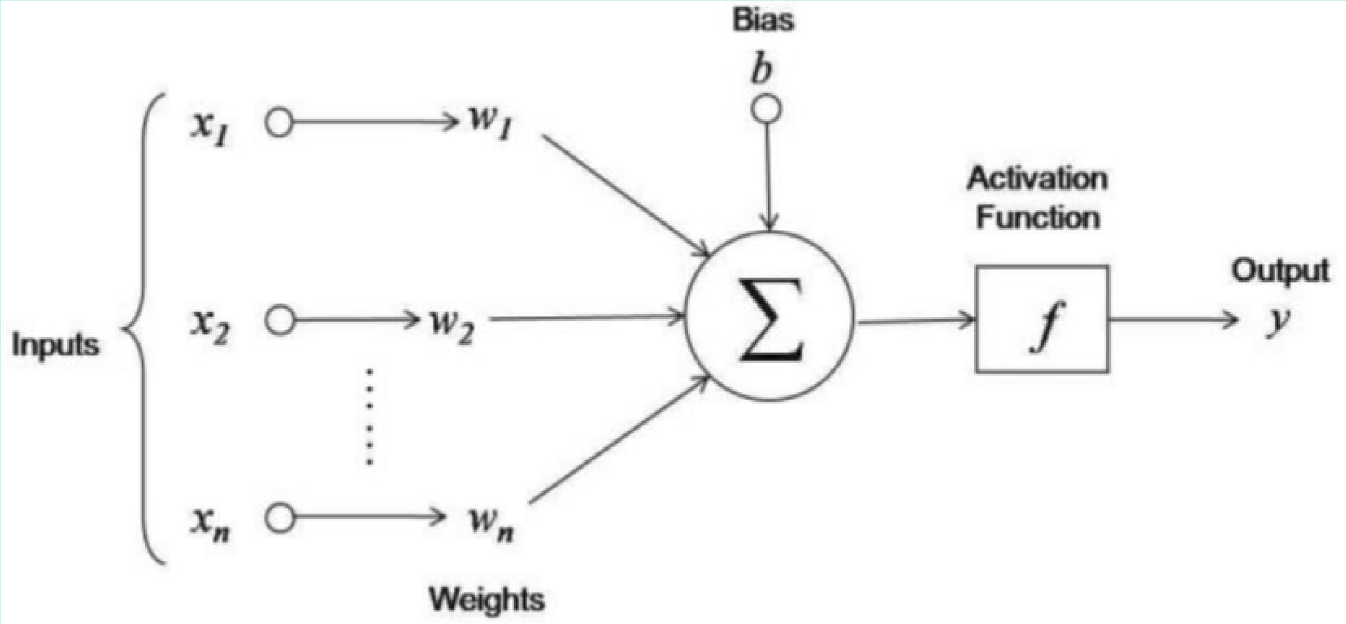
手寫數位識別

機器學習的起始點



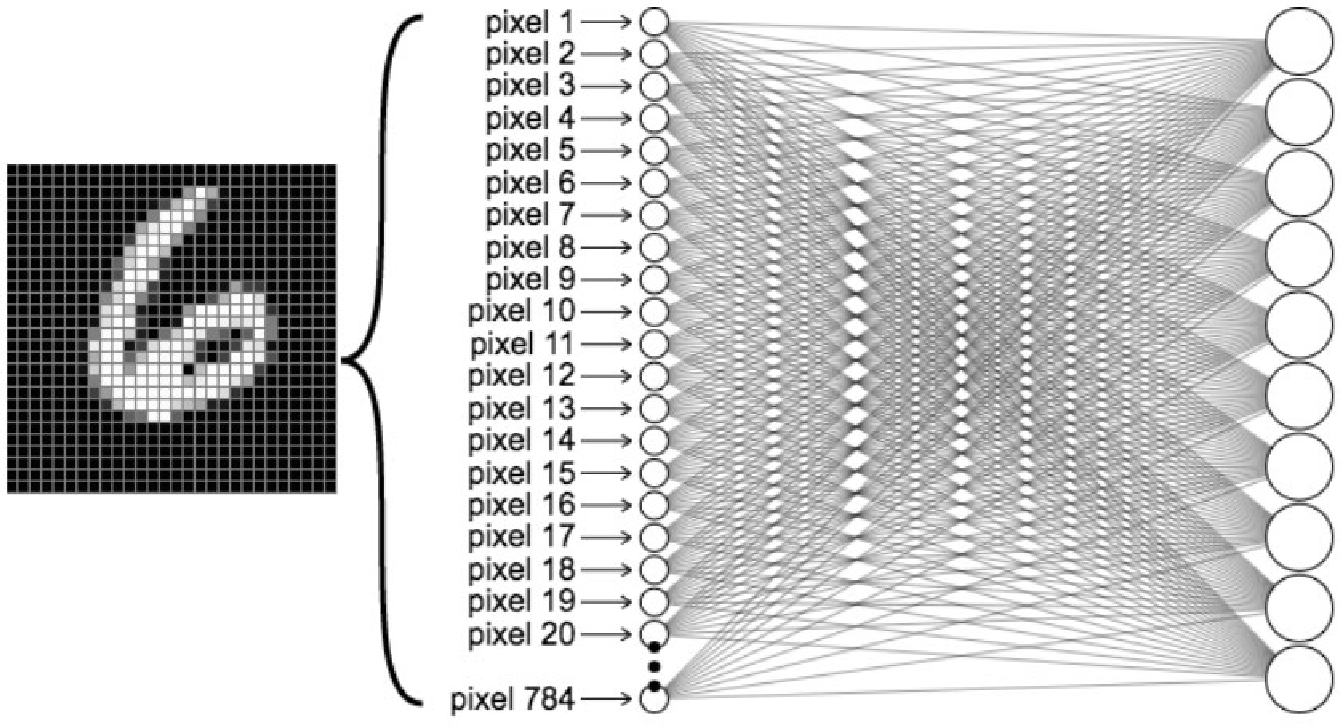
Feed Forward Neural Networks





MNIST handwritten digit dataset

Feed Forward Neural Networks



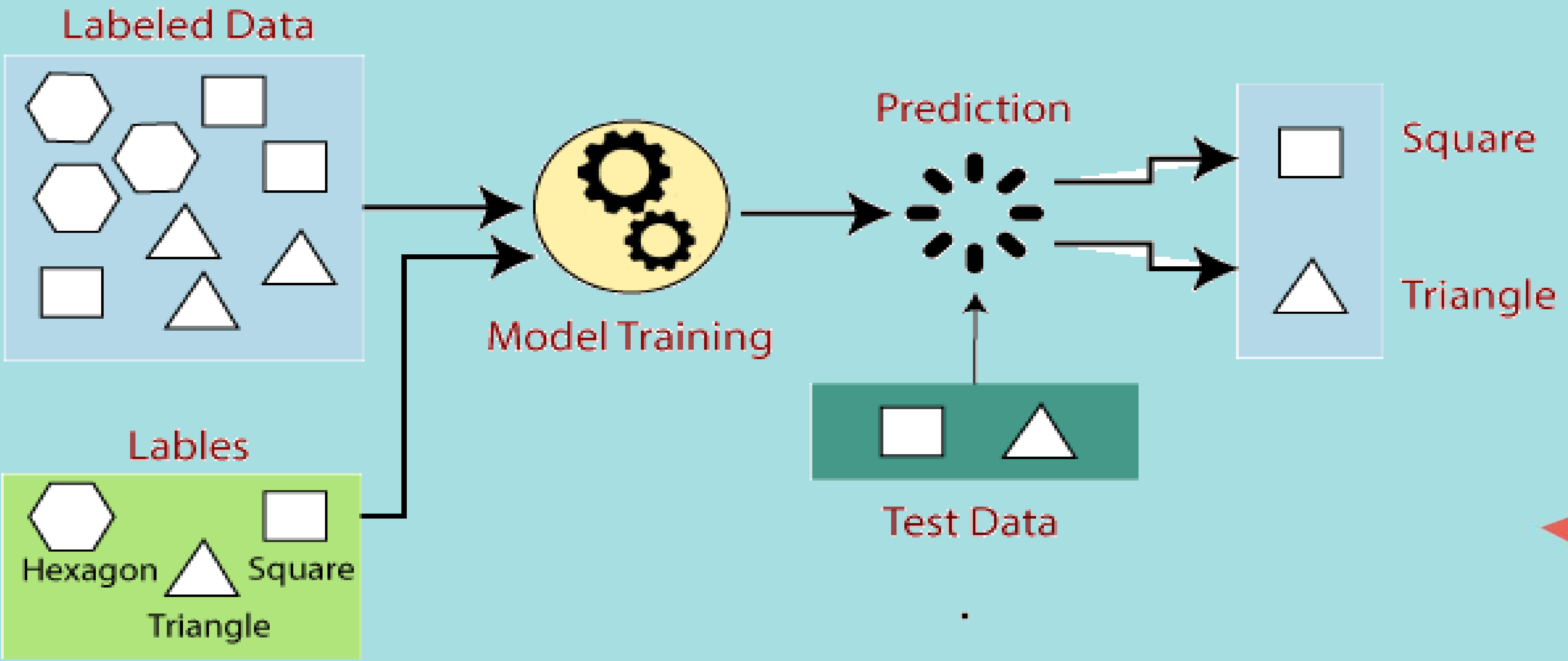
NN with input and output layer, hidden layer is hidden :)

```
model = keras.models.Sequential([
  keras.layers.Flatten(input_shape=(28, 28)),
  keras.layers.Dense(128, activation='relu'),
  keras.layers.Dropout(0.2),
  keras.layers.Dense(10, activation='softmax')
])

model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)

model.evaluate(x_test, y_test)
```

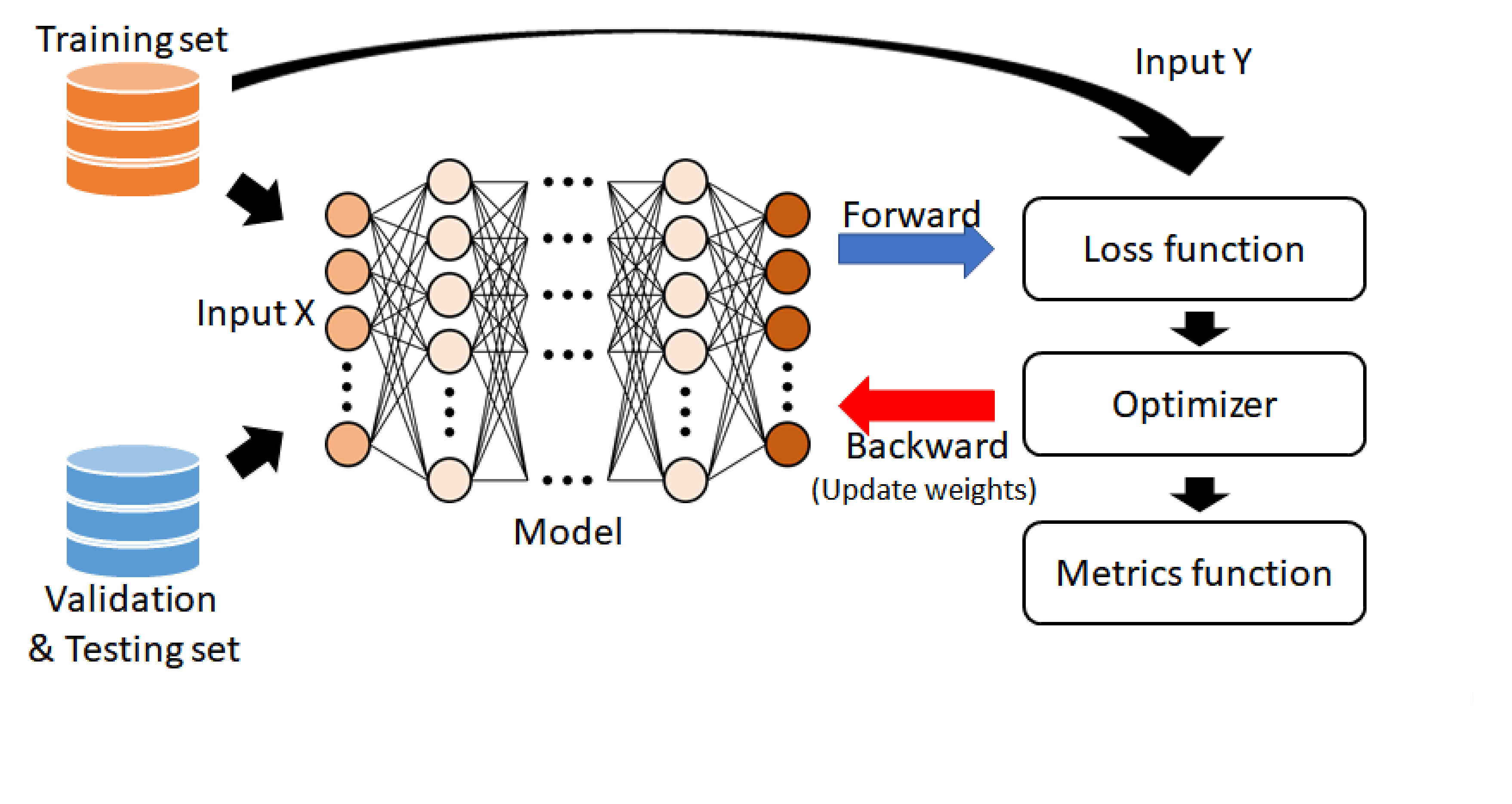
機器學習主要術語

- 什麼是（監督式）機器學習？簡單來說，它的定義如下：
  - 機器學習系統通過學習如何組合輸入資訊來對從未見過的資料做出有用的預測。
- 機器學習的基本術語
  - **標籤 (Labels)**
  - **特徵 (Features)**
  - **樣本 (Examples)**
  - **模型 (Models)**
  - 回歸與分類 (Regression vs. classification)

Features and Labels
特徵和標籤

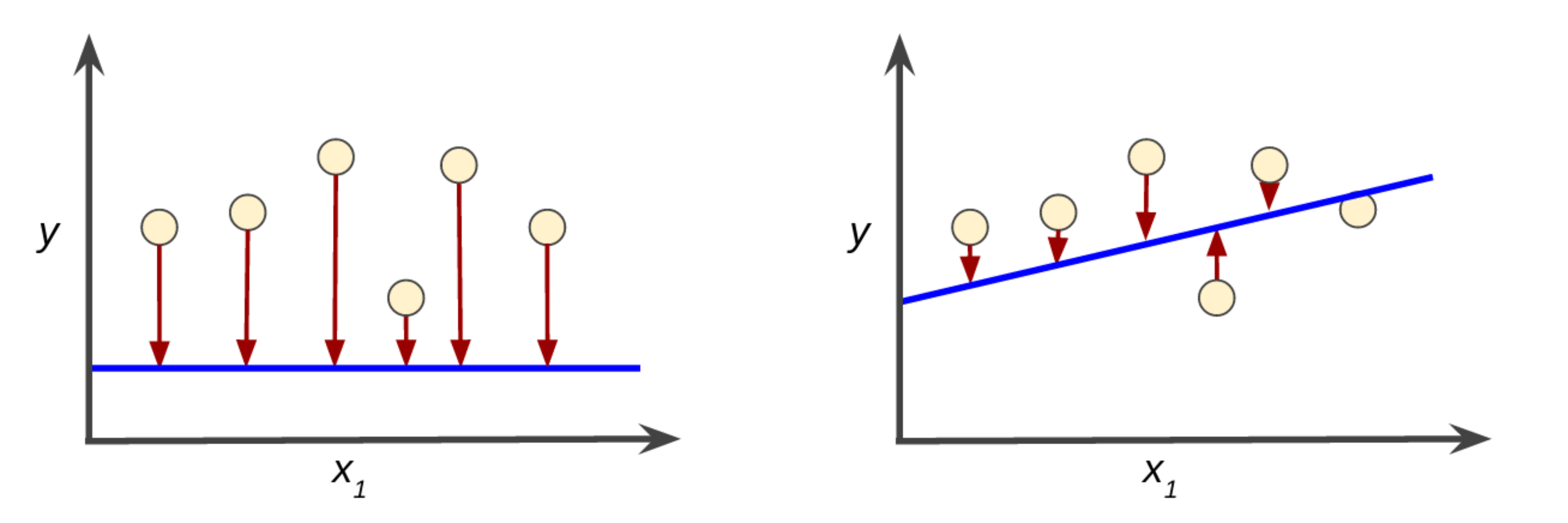


Deep learning 流程



Training and Loss

- Training a model means learning (determining) good values for all the weights and the bias from labeled examples.



誤差值Error

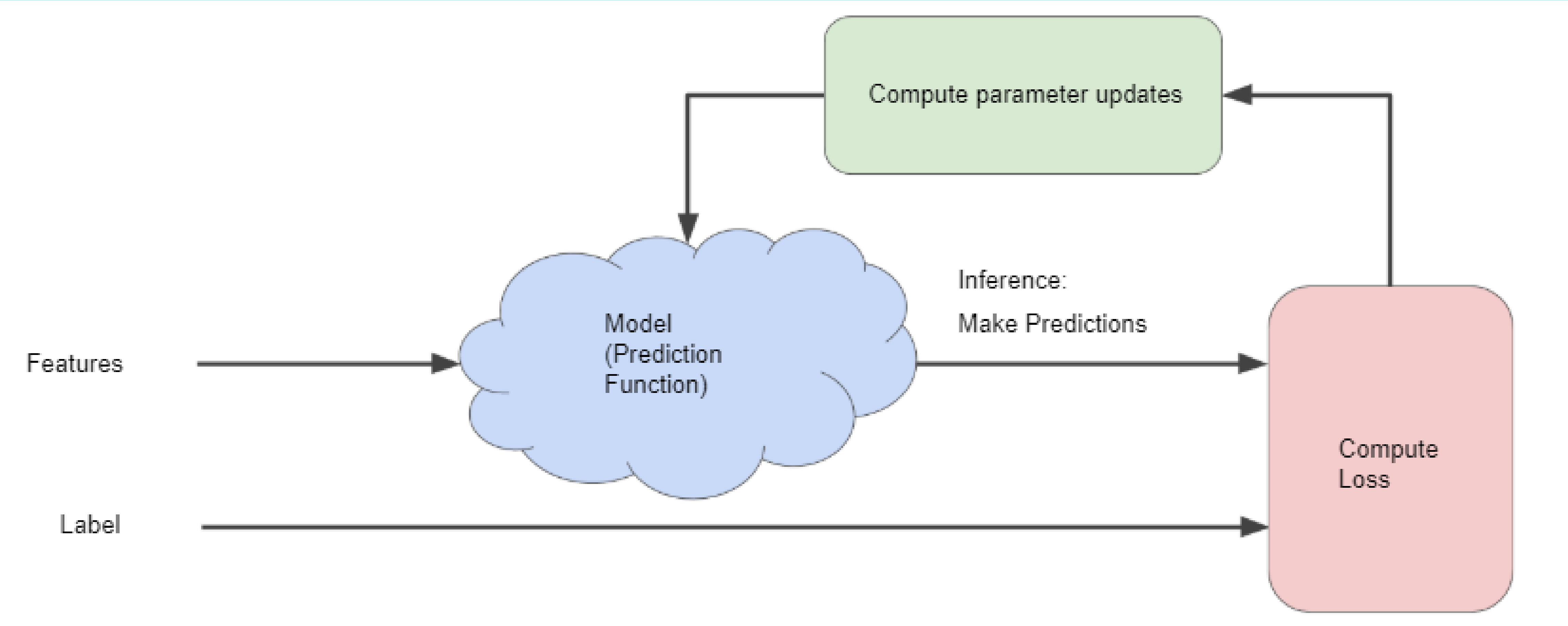
- 均方誤差(Mean square error，MSE) $MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{y}_i\right)^2$
- 平均絕對值誤差(Mean absolute error，MAE)，$MAE = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \widehat{y}_i\right|$
- 交叉熵(cross-entropy)
- 損失函數(loss function)=**實際值-預測值**
- 熵函數 $H(X) = \sum_{i} -p_i log_2(p_i)$

如何根據誤差來調整參數

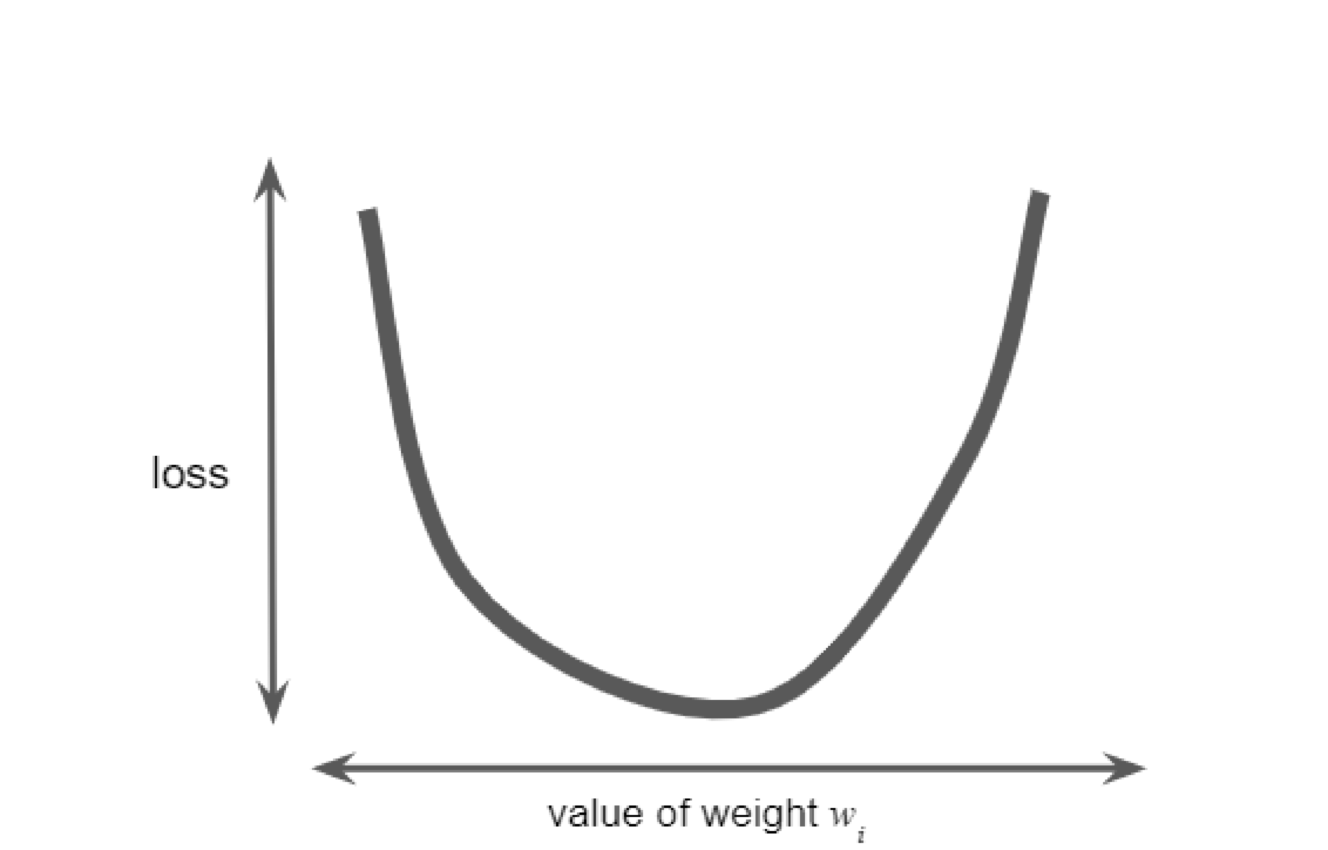
- 反向傳播（Backpropagation)
- 最優化算法

- 梯度下降法(Gradient descent)

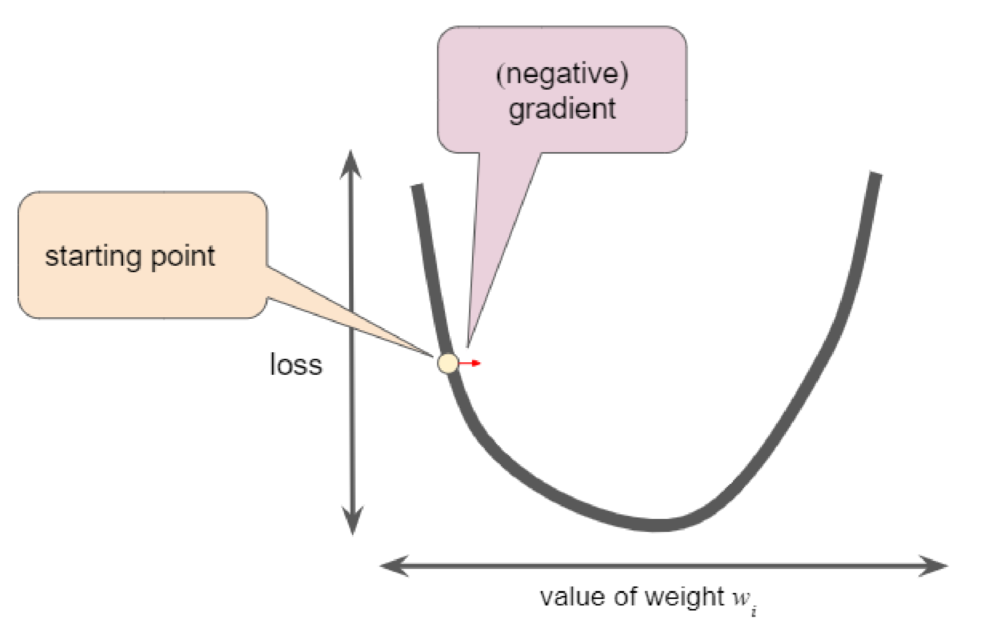
Reducing Loss: An Iterative Approach

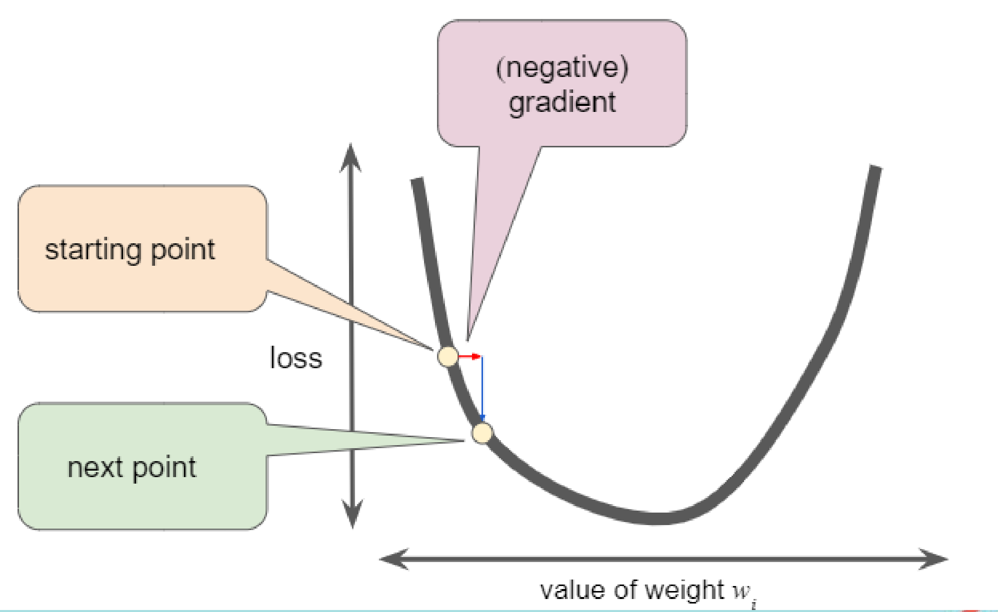


Reducing Loss: Gradient Descent (梯度下降)

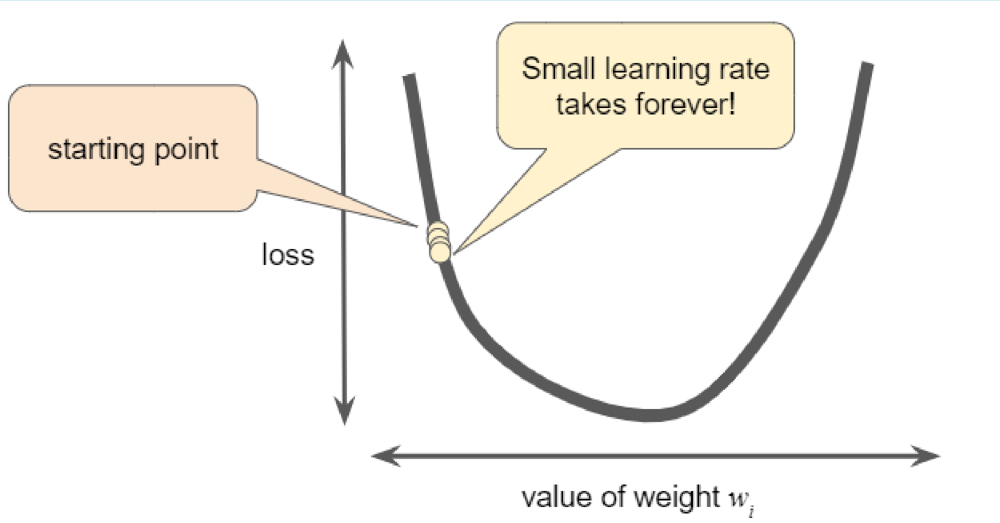


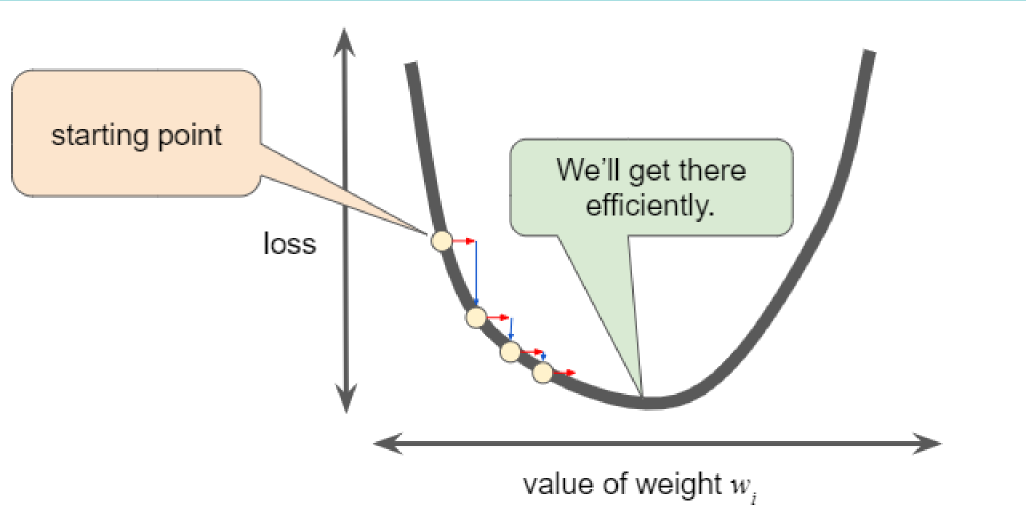
Gradient descent relies on negative gradients

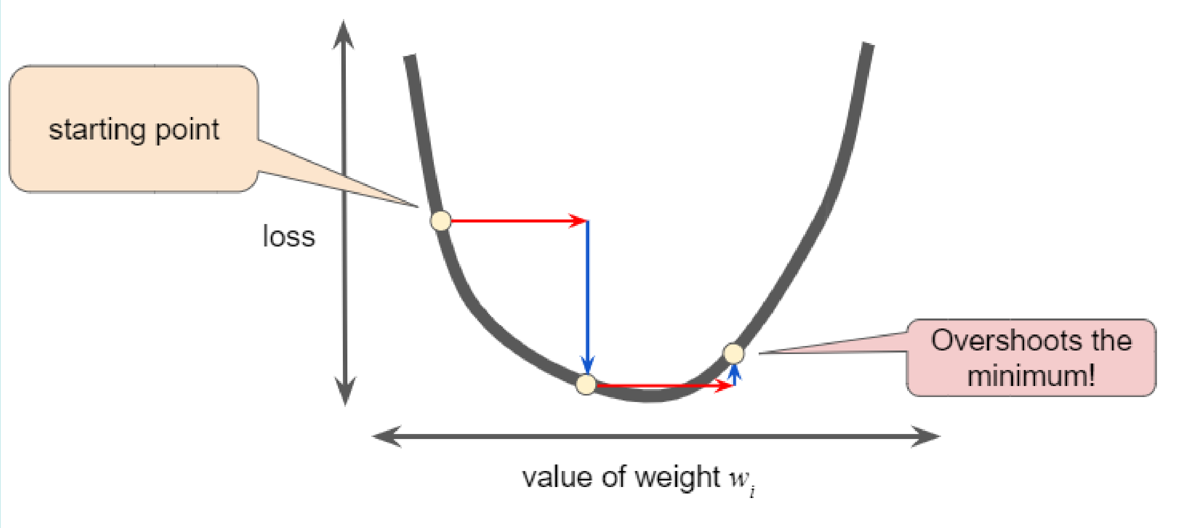




Reducing Loss: Learning Rate(學習率)







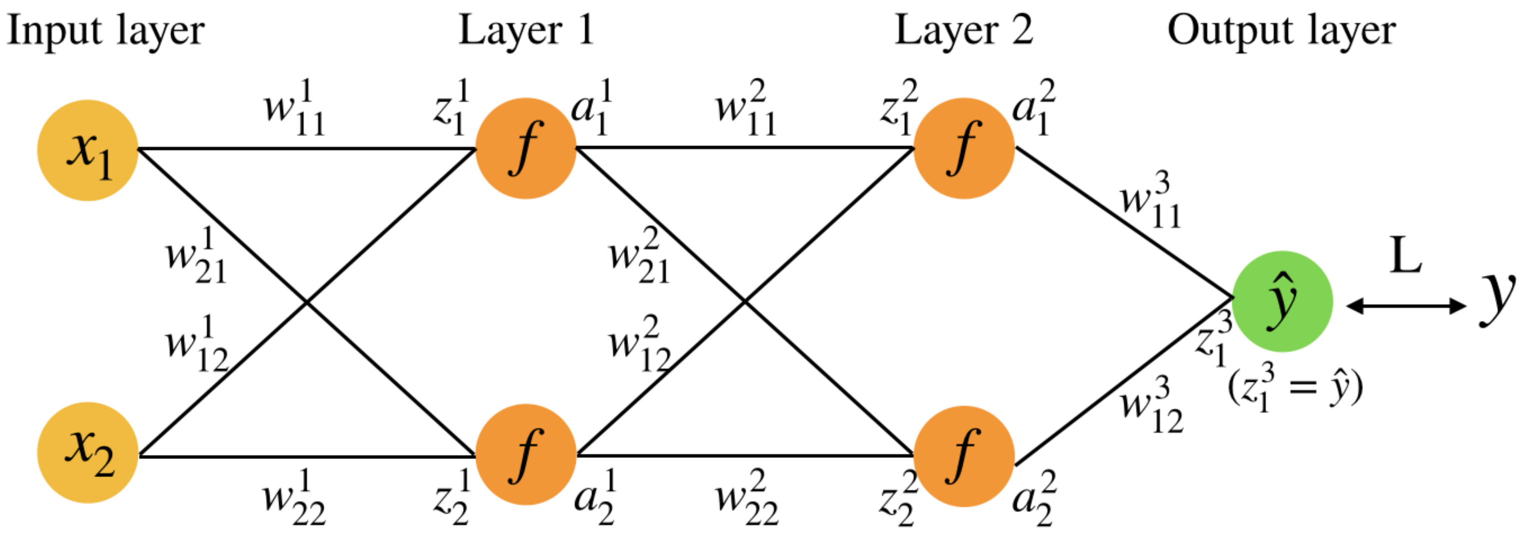
優化器(Optimizer)

- *SGD-準確率梯度下降法 (stochastic gradient decent)*
- *Momentum*
- *AdaGrad-* Adaptive *gradient*
- *Adam= AdaGrad+ Momentum*
- RMSProp (Root Mean Square Prop)

反向傳播計算

- 反向傳播算法的兩個階段：激勵傳播與權重更新。
- 激勵傳播
  - （前向傳播階段）將訓練輸入送入網絡以獲得激勵響應；
  - （反向傳播階段）將激勵響應同訓練輸入對應的目標輸出求差，從而獲得輸出層和隱藏層的響應誤差。
- 權重更新
  - 將輸入激勵和響應誤差相乘，從而獲得權重的梯度；
  - 將這個梯度乘上一個比例並取反後加到權重上。

Backpropagation(**BP**)-Notation



w^1_{11} ← Layer 1 (superscript)

w^1_{11} ← Layer 1 Neuron 1 to Input layer Neuron 1 (subscript)

$z^1_1 = x_1 w^1_{11} + x_2 w^1_{12}$, z_1 : activation function input

$a^1_1 = f(z^1_1)$, f : activation function, a^1_1 : activation output

$\hat{y}$: predict value

Backpropagation(**BP**)-Operation

- **Chain rule:**
  - $y = f(x), z = g(y),\ \ z = g(f(x)) = (g \circ f)(x)$

$$(g \circ f)'(x) = \frac{dg}{dx} = \frac{dg}{df}\frac{df}{dx}$$

  - $z = f(x, y)$, where $x = g(t), y = h(t)$

$$\frac{df}{dt} = \frac{\partial f}{\partial g}\frac{dg}{dt} + \frac{\partial f}{\partial h}\frac{dh}{dt}$$

- **Loss function:**

$$L = \frac{1}{2n}\sum_{i=1}^{n}(\hat{y} - y)^2$$

$$L' = \frac{1}{n}\sum_{i=1}^{n}(\hat{y} - y)$$

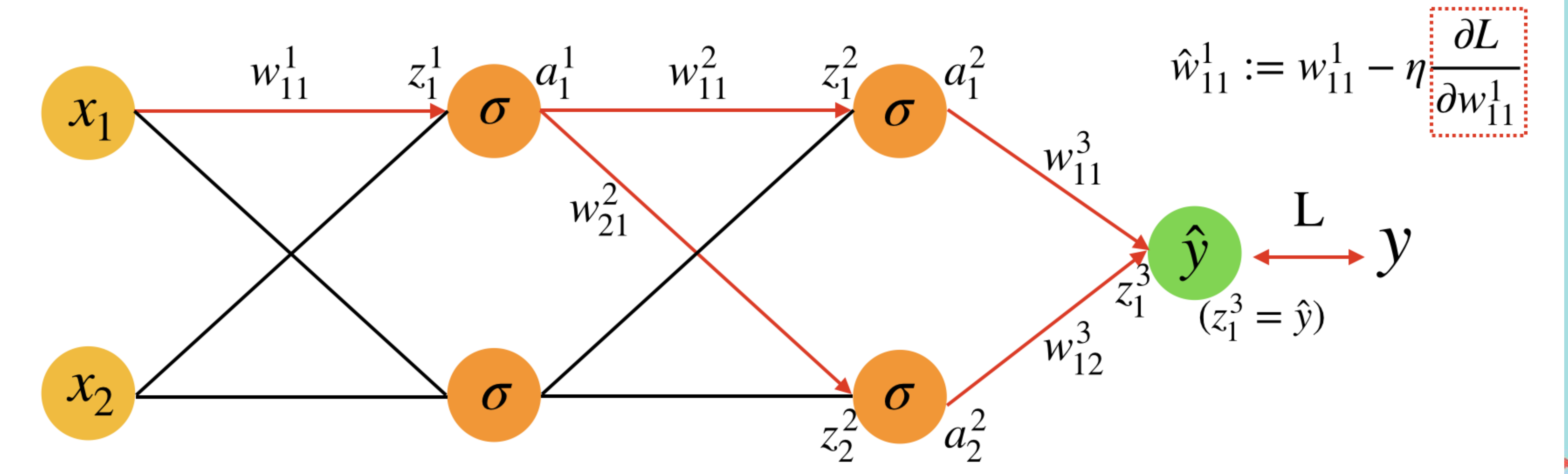
- **Sigmoid function:**

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

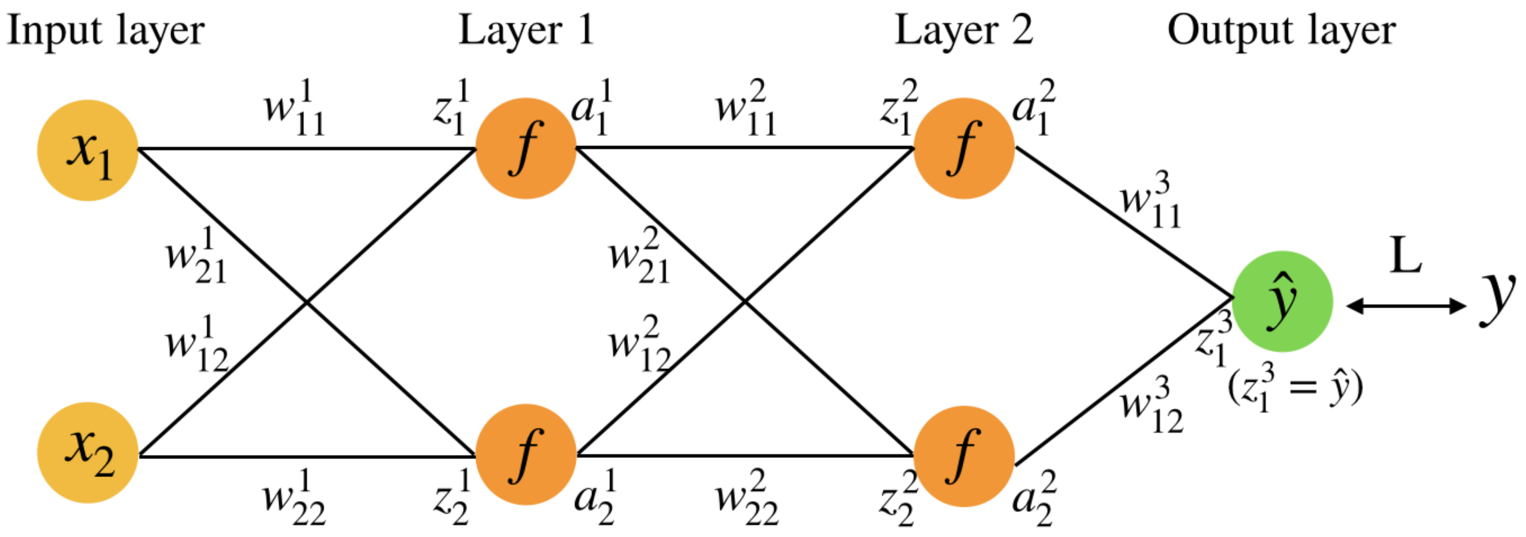
- **Derivative of sigmoid function:**

$$\begin{aligned}\sigma(x)' &= \frac{d}{dx}(1 + e^{-x})^{-1} \\ &= -(1 + e^{-x})^{-2}(-e^{-x}) \\ &= \frac{e^{-x}}{(1 + e^{-x})^2} \\ &= \frac{1}{1 + e^{-x}}\frac{e^{-x}}{1 + e^{-x}} \\ &= \sigma(x)(1 - \sigma(x))\end{aligned}$$

Backpropagation(**BP**)-w¹₁₁的梯度下降



Backpropagation(**BP**)



w_{11}^1 ← Layer 1 (superscript)

w_{11}^1 ← Layer 1 Neuron 1 to Input layer Neuron 1 (subscript)

$z_1^1 = x_1 w_{11}^1 + x_2 w_{12}^1$, z_1 : activation function input

$a_1^1 = f(z_1^1)$, f : activation function, a_1^1 : activation output

$\hat{y}$: predict value

Backpropagation(**BP**)

- The gradient of w_{11}^1 :

$$\frac{\partial L}{\partial w_{11}^1} = \frac{\partial L}{\partial z_1^3}[\sum_{i=1}^{2} \frac{\partial z_1^3}{\partial a_i^2} \frac{\partial a_i^2}{\partial z_i^2} \frac{\partial z_i^2}{\partial a_1^1}] \frac{\partial a_1^1}{\partial z_1^1} \frac{\partial z_1^1}{\partial w_{11}^1} = (\hat{y} - y)[\sum_{i}^{2} w_{1i}^3 \sigma'(z_i^2) w_{i1}^2] \sigma'(z_1^1) x_1$$

1. 2. 3.

1. $$\frac{\partial L}{\partial z_1^3} = \frac{\partial}{\partial \hat{y}} \frac{1}{2} (\hat{y} - y)^2 = (\hat{y} - y),\ (z_1^3 = \hat{y})$$

2. $$\frac{\partial z_1^3}{\partial a_1^2} = \frac{\partial}{\partial a_1^2}(a_1^2 w_{11}^3 + a_2^2 w_{12}^3) = w_{11}^3,\ (z_1^3 = a_1^2 w_{11}^3 + a_2^2 w_{12}^3)$$

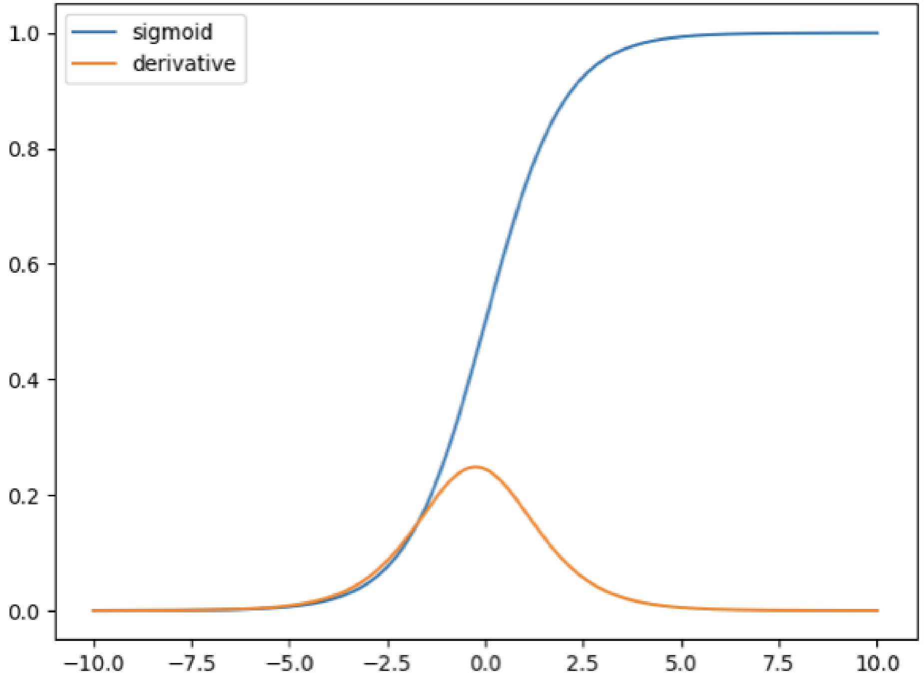
3. $$\frac{\partial a_1^2}{\partial z_1^2} = \sigma'(z_1^2) = \sigma(z_1^2)(1 - \sigma(z_1^2)),\ (a_1^2 = \sigma(z_1^2))$$

Gradient Problem

- **Gradient vanishing (exploding)**

$$\frac{\partial L}{\partial w_{11}^1} = \underset{1.}{(\hat{y} - y)}[\sum_i^2 \underset{2.}{w_{1i}^3}\underset{3.}{\sigma'(z_i^2)}w_{i1}^2]\sigma'(z_1^1)x_1$$

1. $(\hat{y} - y)$: Defined by your loss function .
2. w : Defined by your initialization weight .
3. $\sigma'(z)$: Defined by your activation function .



Stochastic Gradient Descent(隨機梯度下降)

- Different learning rates are used at different learning time points, and of course different directions are considered.
- 在不同學習的時間點用不同的學習率，當然還有考慮不同的方向。

優化器(Optimizer)

- Update weights by gradient $$W \leftarrow W - \eta \frac{\partial L}{\partial W}$$

- W為權重(weight)參數，L為損失函數(loss function)， η是學習率(learning rate)， $\partial L/\partial W$是損失函數對參數的梯度(微分)

AdaGrad

$$W \leftarrow W - \eta \frac{1}{\sqrt{n+\epsilon}} \frac{\partial L}{\partial W}$$

$$n = \sum_{r=1}^{t} \left(\frac{\partial L_r}{\partial W_r}\right)^2$$

$$W \leftarrow W - \eta \frac{1}{\sqrt{\sum_{r=1}^{t}\left(\frac{\partial L_r}{\partial W_r}\right)^2 + \epsilon}} \frac{\partial L}{\partial W}$$

- Adaptive Gradient

- 在AdaGrad Optimizer 中，*η* 乘上 *1/√(n+ϵ)* 再做參數更新，出現了一個*n*的參數，*n*為前面所有梯度值的平方和，利用前面學習的梯度值**平方和**來調整learning rate ，*ϵ*為平滑值，加上 *ϵ*的原因是為了不讓分母為0，*ϵ*一般值為1e-8

Adam

- Adam Optimizer 把Momentum 跟 AdaGrad這二種Optimizer結合

$$W \leftarrow W - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) \frac{\partial L_t}{\partial W_t}$$

$$v_t = \beta_1 v_{t-1} + (1 - \beta_2) (\frac{\partial L_t}{\partial W_t})^2$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}$$

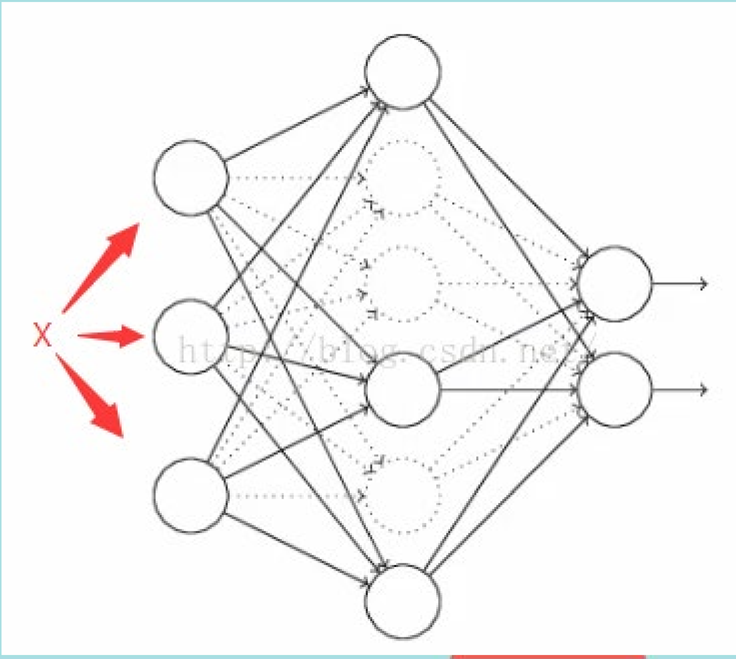
$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Optimizer比較

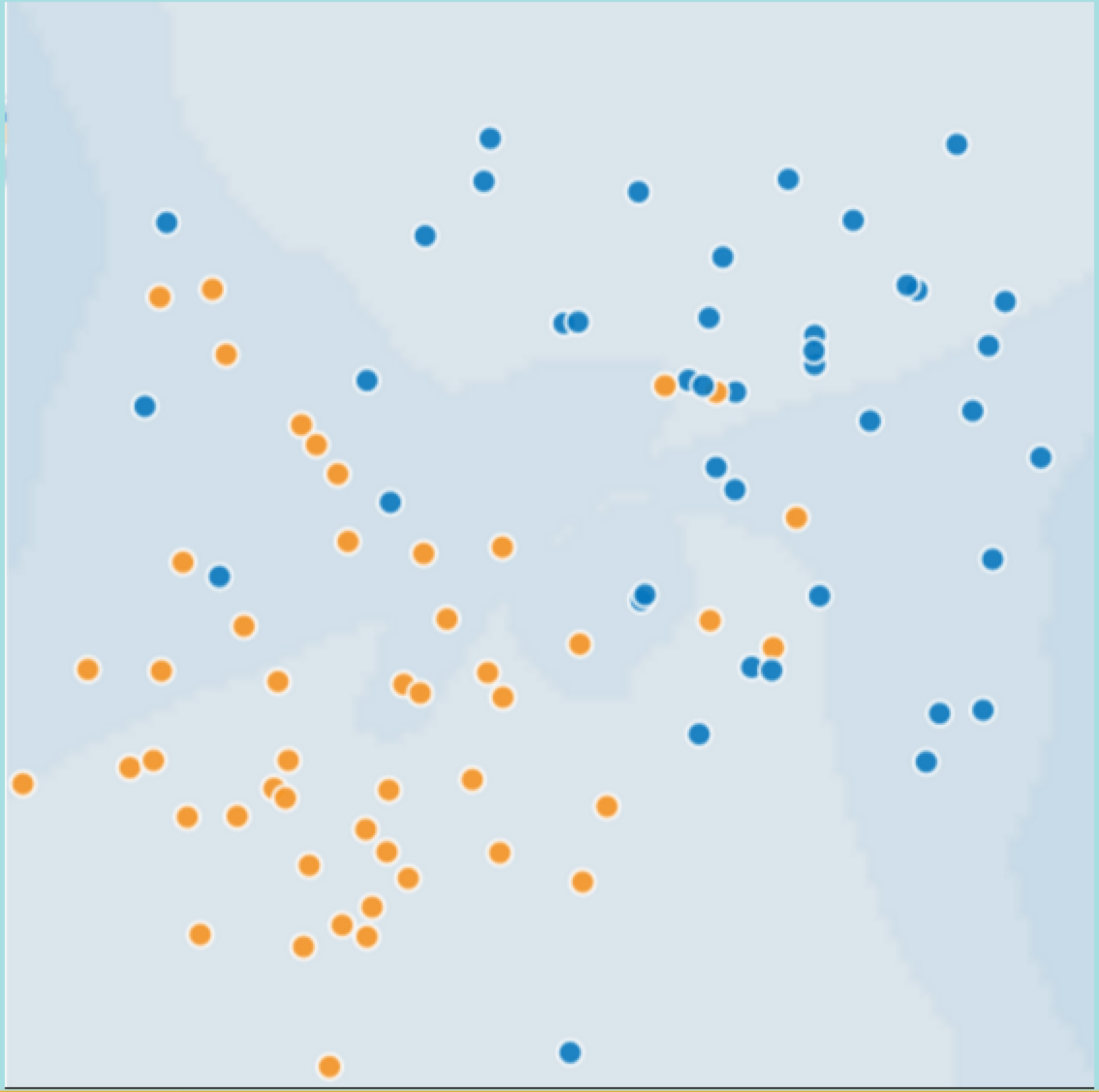
Optimizer	特點
SGD	• 有機會跳出目前局部收斂進而進到另一個局部收斂而得到最小值，而得到全局最小值 • 需自行設定learning rate，較難選擇到合適的learning rate • 會造成loss function有嚴重的震蕩 • 需要較長時間收斂至最小值
Momentum	• 能夠在相關方向加速SGD，抑制SGD的嚴重震蕩，進而加快收斂 • 需自行設定learning rate與β，有可能會使參數的移動方向偏移梯度下分的方向，進而導至沒有那麼快速的收斂
AdaGrad	• 能夠自動調整learning rate，進而調整收斂 • 適合處理稀疏梯度 • 依然需要人工設置一個全局的learning rate • 後期，分母梯度平方的累加會越來越大，會使梯度趨近於0，使得訓練結束
Adam	• 結合了AdaGrad與Momentum的優點 • 適用於大數據集和高維空間的資料 • 目前最常使用的一個Optimizer

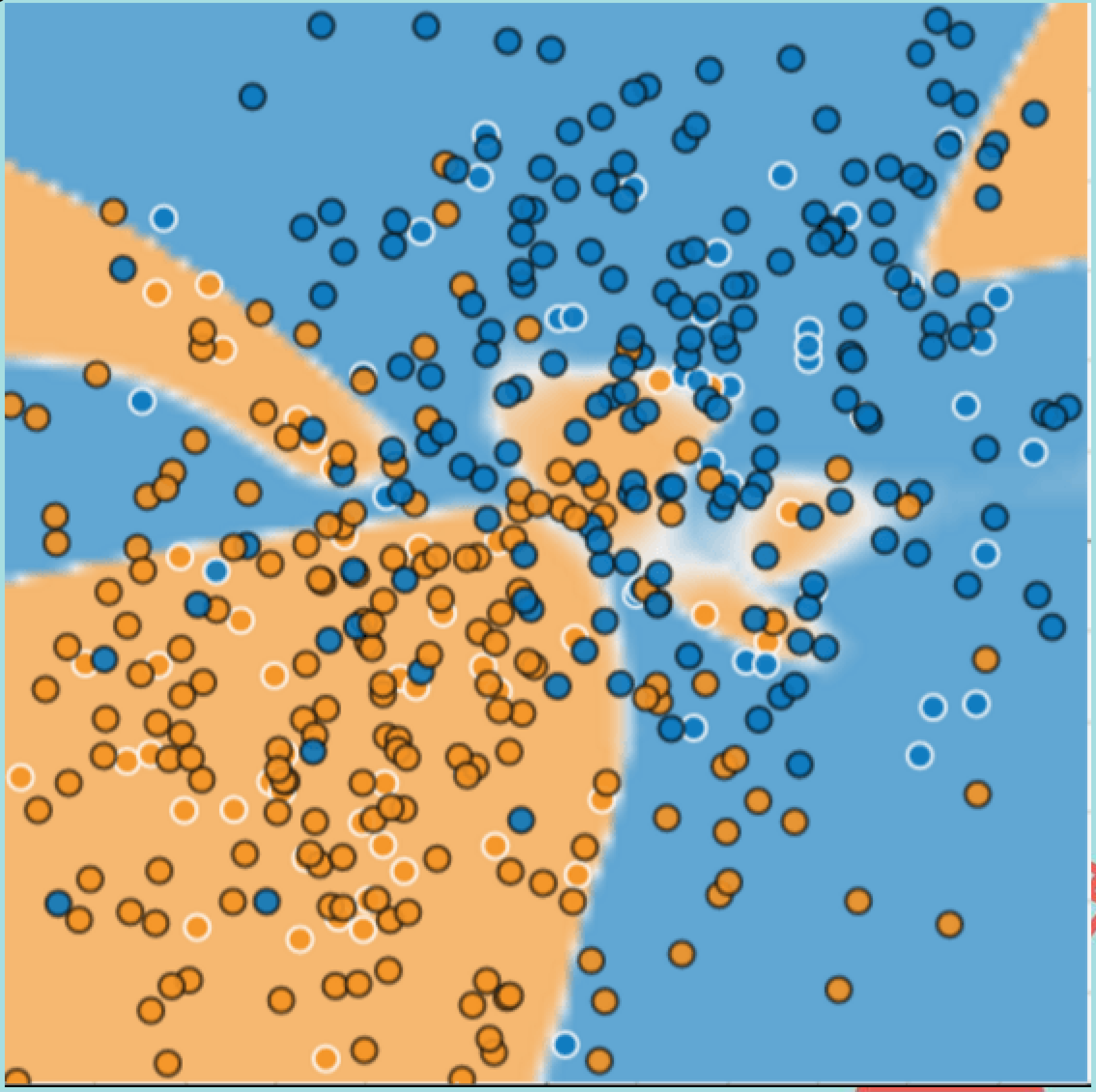
Overfitting和regularization

- Overfitting
- Regularization
- Dropout: 在訓練的時候只算部分的神經元

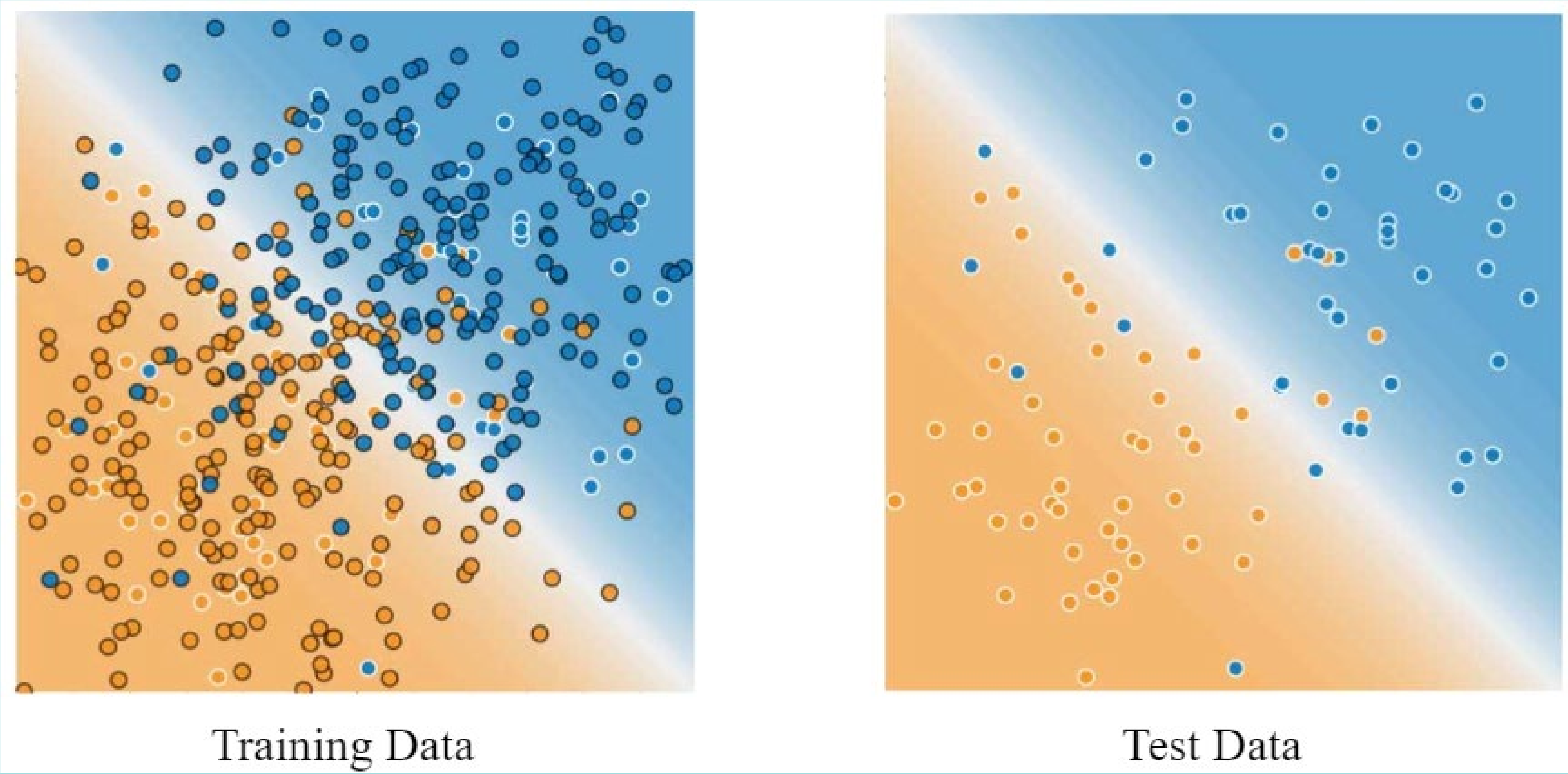


Overfitting

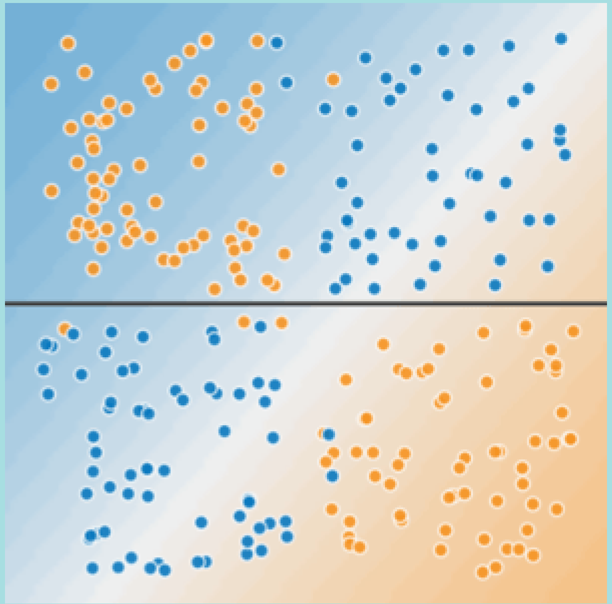




Training and Test Sets

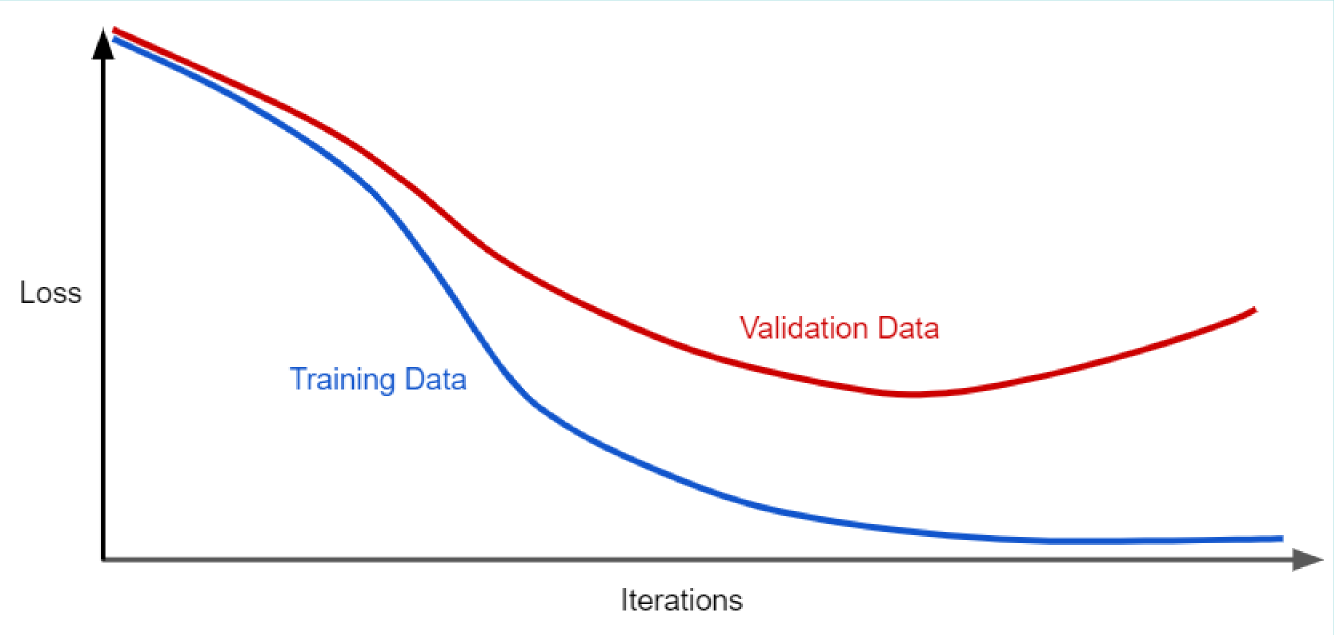


Feature crosses (特徵交叉)



- feature cross is a synthetic feature that encodes nonlinearity in the feature space by multiplying two or more input features together. (The term cross comes from cross product.)
- 特徵交叉是一種合成特徵，它通過將兩個或多個輸入特徵相乘在一起來編碼特徵空間中的非線性。（術語“交叉”來自交叉積。）

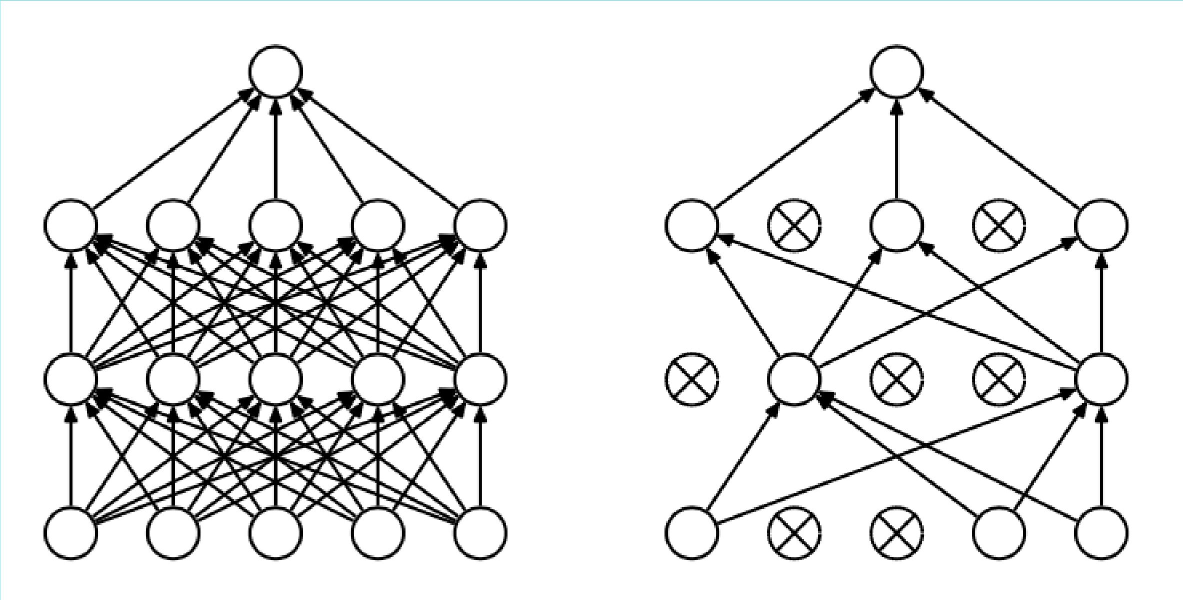
Regularization (正規化)



$$\text{minimize}(\text{Loss}(\text{Data}|\text{Model}) + \text{complexity}(\text{Model}))$$

$$L_2 \text{ regularization term} = ||\boldsymbol{w}||_2^2 = w_1^2 + w_2^2 + \ldots + w_n^2$$

Dropout (丟棄)



- A form of regularization useful in training neural networks. Dropout regularization works by removing a random selection of a fixed number of the units in a network layer for a single gradient step.

(3) NUMPY/PANDAS

NumPy & Pandas

NumPy

Scipy.org

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary seamlessly and speedily integrate with a wide variety of databases.

NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Getting Started

- Getting NumPy
- Installing the SciPy Stack
- NumPy and SciPy documentation page
- NumPy Tutorial
- NumPy for MATLAB© Users
- NumPy functions by category
- NumPy Mailing List

For more information on the SciPy Stack (for which NumPy provides the fundamental array data structure), see scipy.org.

pandas

About us Getting started Documentation Community Contribute

Library Highlights

- A fast and efficient **DataFrame** object for data manipulation with integrated indexing;
- Tools for **reading and writing data** between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format;
- Intelligent **data alignment** and integrated handling of **missing data**: gain automatic label-based alignment in computations and easily manipulate messy data into an orderly form;
- Flexible **reshaping** and pivoting of data sets;
- Intelligent label-based **slicing**, **fancy indexing**, and **subsetting** of large data sets;
- Columns can be inserted and deleted from data structures for **size mutability**;
- Aggregating or transforming data with a powerful **group by** engine allowing split-apply-combine operations on data sets;
- High performance **merging and joining** of data sets;
- **Hierarchical axis indexing** provides an intuitive way of working with high-dimensional data in a lower-dimensional data structure;
- **Time series**-functionality: date range generation and frequency conversion, moving window statistics, date shifting and lagging. Even create domain-specific time offsets and join time series without losing data;
- Highly **optimized for performance**, with critical code paths written in Cython or C.
- Python with *pandas* is in use in a wide variety of **academic and commercial** domains, including Finance, Neuroscience, Economics, Statistics, Advertising, Web Analytics, and more.

https://numpy.org/ https://pandas.pydata.org/

NumPy cheat sheet

Numpy Cheat Sheet

Python Package

Created By: Arianne Colton and Sean Chen

Numpy (Numerical Python)

What is NumPy?

Foundation package for scientific computing in Python

Why NumPy?

- Numpy '**ndarray**' is a much more efficient way of storing and manipulating "**numerical data**" than the built-in Python data structures.
- Libraries written in lower-level languages, such as C, can operate on data stored in Numpy '**ndarray**' without copying any data.

N-Dimensional Array (NDArray)

What is NdArray?

Fast and space-efficient multidimensional array (container for homogeneous data) providing vectorized arithmetic operations

Create NdArray	`np.array(seq1)` # seq1 - is any sequence like object, i.e. [1, 2, 3]
Create Special NdArray	1, `np.zeros(10)` # one dimensional ndarray with 10 elements of value 0 2, `np.ones(2, 3)` # two dimensional ndarray with 6 elements of value 1 3, `np.empty(3, 4, 5)` * # three dimensional ndarray of uninitialized values 4, `np.eye(N)` or `np.identity(N)` # creates N by N identity matrix
NdArray version of Python's range	`np.arange(1, 10)`
Get # of Dimension	`ndarray1.ndim`
Get Dimension Size	`dim1size, dim2size, .. = ndarray1.shape`
Get Data Type **	`ndarray1.dtype`
Explicit Casting	`ndarray2 = ndarray1.astype(np.int32)` ***

* Cannot assume empty() will return all zeros. It could be garbage values.

** Default data type is '**np.float64**'. This is equivalent to Python's float type which is 8 bytes (64 bits); thus the name 'float64'.

*** If casting were to fail for some reason, '**TypeError**' will be raised.

Slicing (Indexing/Subsetting)

- Slicing (i.e. `ndarray1[2:6]`) is a '**view**' on the original array. **Data is NOT copied.** Any modifications (i.e. `ndarray1[2:6] = 8`) to the 'view' will be reflected in the original array.
- Instead of a 'view', explicit copy of slicing via :

```
ndarray1[2:6].copy()
```

- Multidimensional array indexing notation :

```
ndarray1[0][2] or ndarray1[0, 2]
```

*** Boolean indexing** :

```
ndarray1[(names == 'Bob') | (names == 'Will'), 2:]
# '2:' means select from 3rd column on
```

* Selecting data by boolean indexing **ALWAYS** creates a copy of the data.

* The 'and' and 'or' keywords do NOT work with boolean arrays. Use & and |.

*** Fancy indexing** (aka 'indexing using integer arrays')

Select a subset of rows in a particular order :

```
ndarray1[ [3, 8, 4] ]
ndarray1[ [-1, 6] ]
# negative indices select rows from the end
```

* Fancy indexing **ALWAYS** creates a copy of the data.

Numpy (Numerical Python)

Setting data with assignment :

```
ndarray1[ndarray1 < 0] = 0 *
```

* If ndarray1 is two-dimensions, ndarray1 < 0 creates a two-dimensional boolean array.

Common Operations

1. **Transposing**
   - A special form of reshaping which returns a '**view**' on the underlying data without copying anything.

```
ndarray1.transpose()      or
ndarray1.T                or
ndarray1.swapaxes(0, 1)
```

2. **Vectorized wrappers (for functions that take scalar values)**
   - `math.sqrt()` works on only a `scalar`

```
np.sqrt(seq1) # any sequence (list, ndarray, etc) to return a ndarray
```

3. **Vectorized expressions**
   - `np.where(cond, x, y)` is a vectorized version of the expression 'x if condition else y'

```
np.where([True, False], [1, 2], [2, 3]) => ndarray (1, 3)
```

   - Common Usages :

```
np.where(matrixArray > 0, 1, -1)
=> a new array (same shape) of 1 or -1 values
np.where(cond, 1, 0).argmax() *
=> Find the first True element
```

* `argmax()` can be used to find the index of the maximum element. Example usage is find the first element that has a "price > number" in an array of price data.

4. **Aggregations/Reductions Methods (i.e. mean, sum, std)**

Compute mean	`ndarray1.mean()` or `np.mean(ndarray1)`
Compute statistics over axis *	`ndarray1.mean(axis = 1)` `ndarray1.sum(axis = 0)`

* axis = 0 means column axis, 1 is row axis.

5. **Boolean arrays methods**

Count # of 'Trues' in boolean array	`(ndarray1 > 0).sum()`
If at least one value is 'True'	`ndarray1.any()`
If all values are 'True'	`ndarray1.all()`

Note: These methods also work with non-boolean arrays, where non-zero elements evaluate to True.

6. **Sorting**

Inplace sorting	`ndarray1.sort()`
Return a sorted copy instead of inplace	`sorted1 = np.sort(ndarray1)`

7. **Set methods**

Return sorted unique values	`np.unique(ndarray1)`
Test membership of ndarray1 values in [2, 3, 6]	`resultBooleanArray = np.in1d(ndarray1, [2, 3, 6])`

   - Other set methods : `intersect1d()`, `union1d()`, `setdiff1d()`, `setxor1d()`

8. **Random number generation** (np.random)
   - Supplements the built-in Python random * with functions for efficiently generating whole arrays of sample values from many kinds of probability distributions.

```
samples = np.random.normal(size =(3, 3))
```

* Python built-in random ONLY samples one value at a time.

Created by Arianne Colton and Sean Chen

www.datasciencefree.com

Based on content from 'Python for Data Analysis' by Wes McKinney

Updated: August 18, 2016

Pandas cheat sheet

Python For Data Science *Cheat Sheet*

Pandas Basics

Learn Python for Data Science Interactively at www.DataCamp.com

Pandas

The Pandas library is built on NumPy and provides easy-to-use data structures and data analysis tools for the Python programming language.

pandas

Use the following import convention:

```
>>> import pandas as pd
```

Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type

Index	
A	3
B	-5
C	7
D	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

A two-dimensional labeled data structure with columns of potentially different types

Columns

Index	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasilia	207847528

```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasilia'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

Asking For Help

```
>>> help(pd.Series.loc)
```

Selection

Also see NumPy Arrays

Getting

`>>> s['b']` `-5`	Get one element

`>>> df[1:]`
 `Country Capital Population`
 `1 India New Delhi 1303171035`
 `2 Brazil Brasilia 207847528`
	Get subset of a DataFrame

Selecting, Boolean Indexing & Setting

By Position

`>>> df.iloc([0],[0])`
 `'Belgium'`
 `>>> df.iat([0],[0])`
 `'Belgium'`
	Select single value by row & column

By Label

`>>> df.loc([0], ['Country'])`
 `'Belgium'`
 `>>> df.at([0], ['Country'])`
 `'Belgium'`
	Select single value by row & column labels

By Label/Position

`>>> df.ix[2]`
 `Country Brazil`
 `Capital Brasilia`
 `Population 207847528`
	Select single row of subset of rows

`>>> df.ix[:,'Capital']`
 `0 Brussels`
 `1 New Delhi`
 `2 Brasilia`
	Select a single column of subset of columns

`>>> df.ix[1,'Capital']` `'New Delhi'`	Select rows and columns

Boolean Indexing

`>>> s[~(s > 1)]`	Series s where value is not >1
`>>> s[(s < -1) | (s > 2)]`	s where value is <-1 or >2
`>>> df[df['Population']>1200000000]`	Use filter to adjust DataFrame

Setting

`>>> s['a'] = 6`	Set index a of Series s to 6

Dropping

`>>> s.drop(['a', 'c'])`	Drop values from rows (axis=0)
`>>> df.drop('Country', axis=1)`	Drop values from columns(axis=1)

Sort & Rank

`>>> df.sort_index()`	Sort by labels along an axis
`>>> df.sort_values(by='Country')`	Sort by the values along an axis
`>>> df.rank()`	Assign ranks to entries

Retrieving Series/DataFrame Information

Basic Information

`>>> df.shape`	(rows,columns)
`>>> df.index`	Describe index
`>>> df.columns`	Describe DataFrame columns
`>>> df.info()`	Info on DataFrame
`>>> df.count()`	Number of non-NA values

Summary

`>>> df.sum()`	Sum of values
`>>> df.cumsum()`	Cummulative sum of values
`>>> df.min()/df.max()`	Minimum/maximum values
`>>> df.idxmin()/df.idxmax()`	Minimum/Maximum index value
`>>> df.describe()`	Summary statistics
`>>> df.mean()`	Mean of values
`>>> df.median()`	Median of values

Applying Functions

`>>> f = lambda x: x*2`	
`>>> df.apply(f)`	Apply function
`>>> df.applymap(f)`	Apply function element-wise

Data Alignment

Internal Data Alignment

NA values are introduced in the indices that don't overlap:

```
>>> s3 = pd.Series([7, -2, 3], index=['a', 'c', 'd'])
>>> s + s3
  a    10.0
  b     NaN
  c     5.0
  d     7.0
```

Arithmetic Operations with Fill Methods

You can also do the internal data alignment yourself with the help of the fill methods:

```
>>> s.add(s3, fill_value=0)
  a    10.0
  b    -5.0
  c    5.0
  d    7.0
>>> s.sub(s3, fill_value=2)
>>> s.div(s3, fill_value=4)
>>> s.mul(s3, fill_value=3)
```

I/O

Read and Write to CSV

```
>>> pd.read_csv('file.csv', header=None, nrows=5)
>>> pd.to_csv('myDataFrame.csv')
```

Read and Write to Excel

```
>>> pd.read_excel('file.xlsx')
>>> pd.to_excel('dir/myDataFrame.xlsx', sheet_name='Sheet1')
```

Read multiple sheets from the same file

```
>>> xlsx = pd.ExcelFile('file.xls')
>>> df = pd.read_excel(xlsx, 'Sheet1')
```

Read and Write to SQL Query or Database Table

```
>>> from sqlalchemy import create_engine
>>> engine = create_engine('sqlite:///:memory:')
>>> pd.read_sql("SELECT * FROM my_table;", engine)
>>> pd.read_sql_table('my_table', engine)
>>> pd.read_sql_query("SELECT * FROM my_table;", engine)
```

`read_sql()` is a convenience wrapper around `read_sql_table()` and `read_sql_query()`

```
>>> pd.to_sql('myDf', engine)
```

DataCamp
Learn Python for Data Science Interactively

Thanks! Q&A