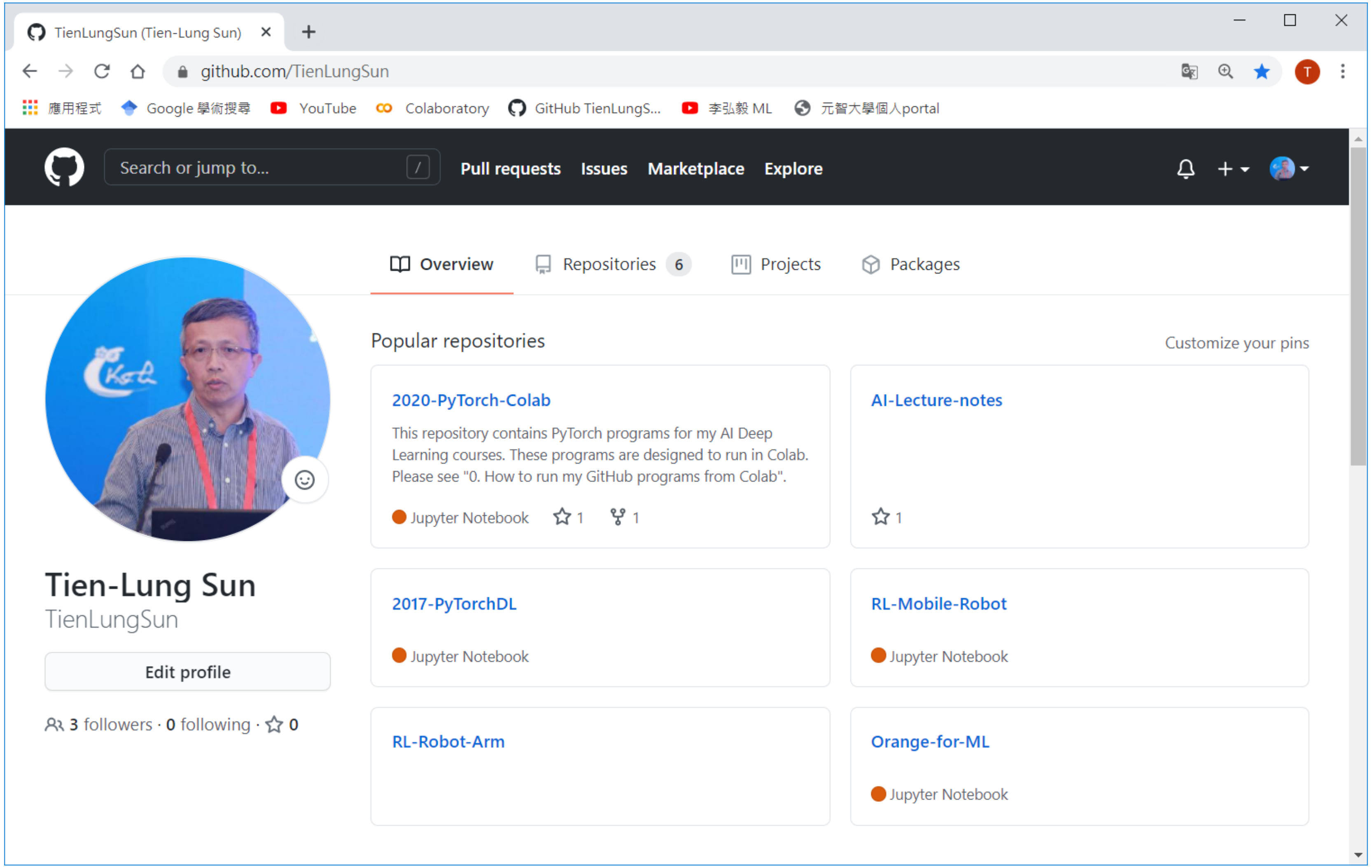
Introduction to Deep Learning – Concepts and PyTorch Development

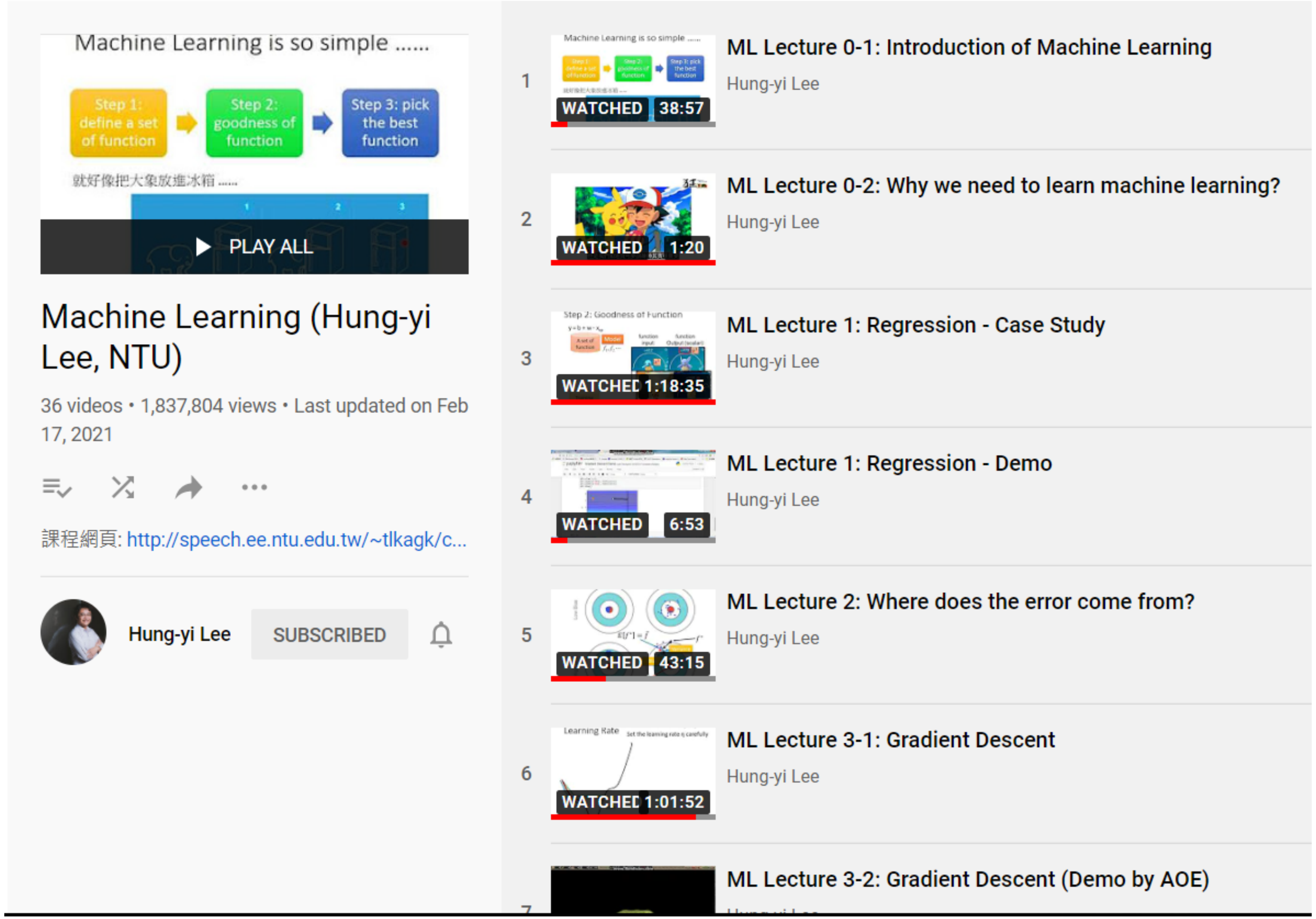
Tien-Lung Sun, Dept. of Industrial Engineering and Management, Yuan-Ze University
孫天龍 元智大學 工業工程與管理系 教授

My GitHub



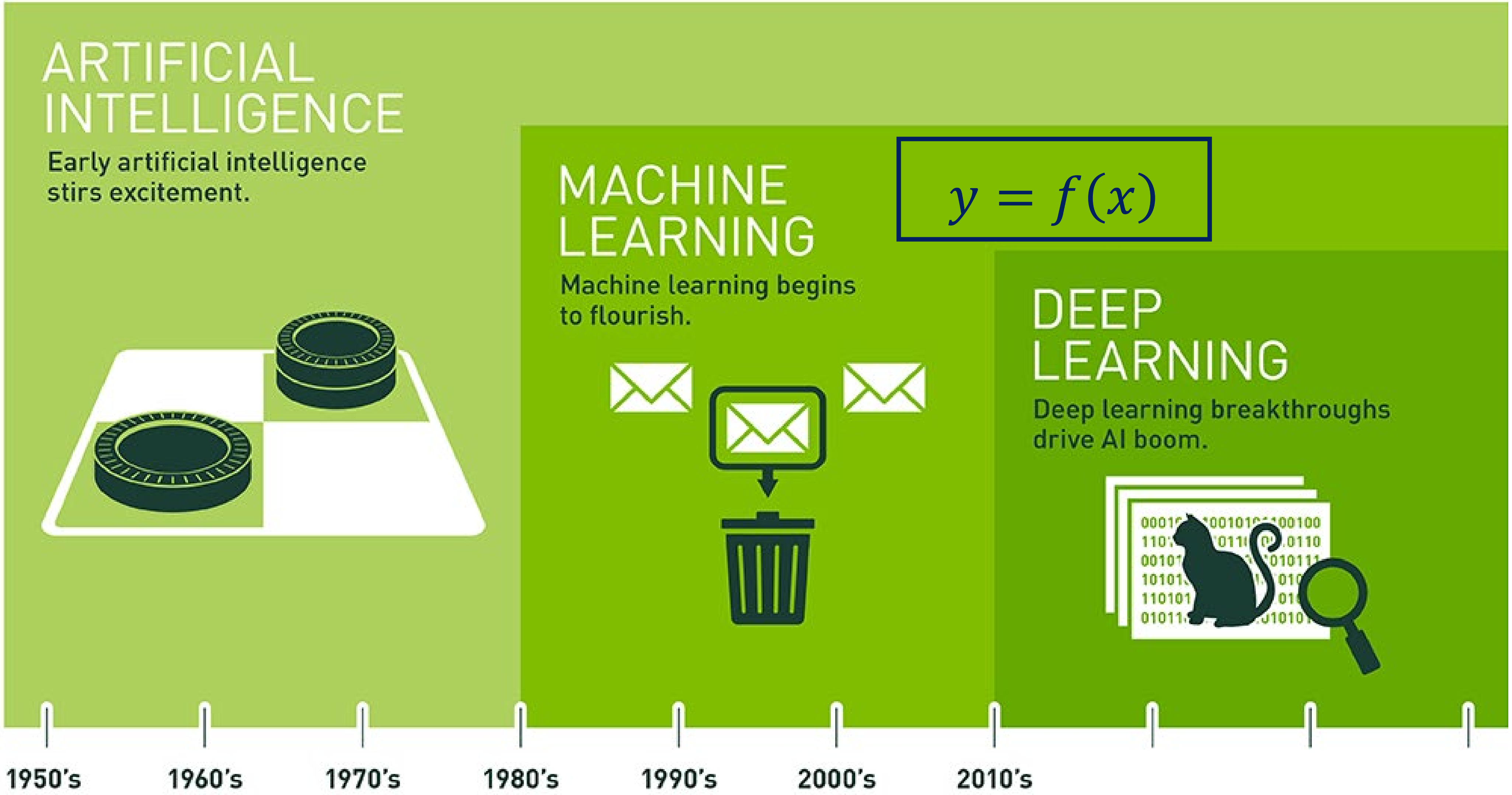
http://github.com/TienLungSun

Acknowledgement



https://www.youtube.com/playlist?list=PLJV_el3uVTsPy9oCRY30oBPNLCo89yu49

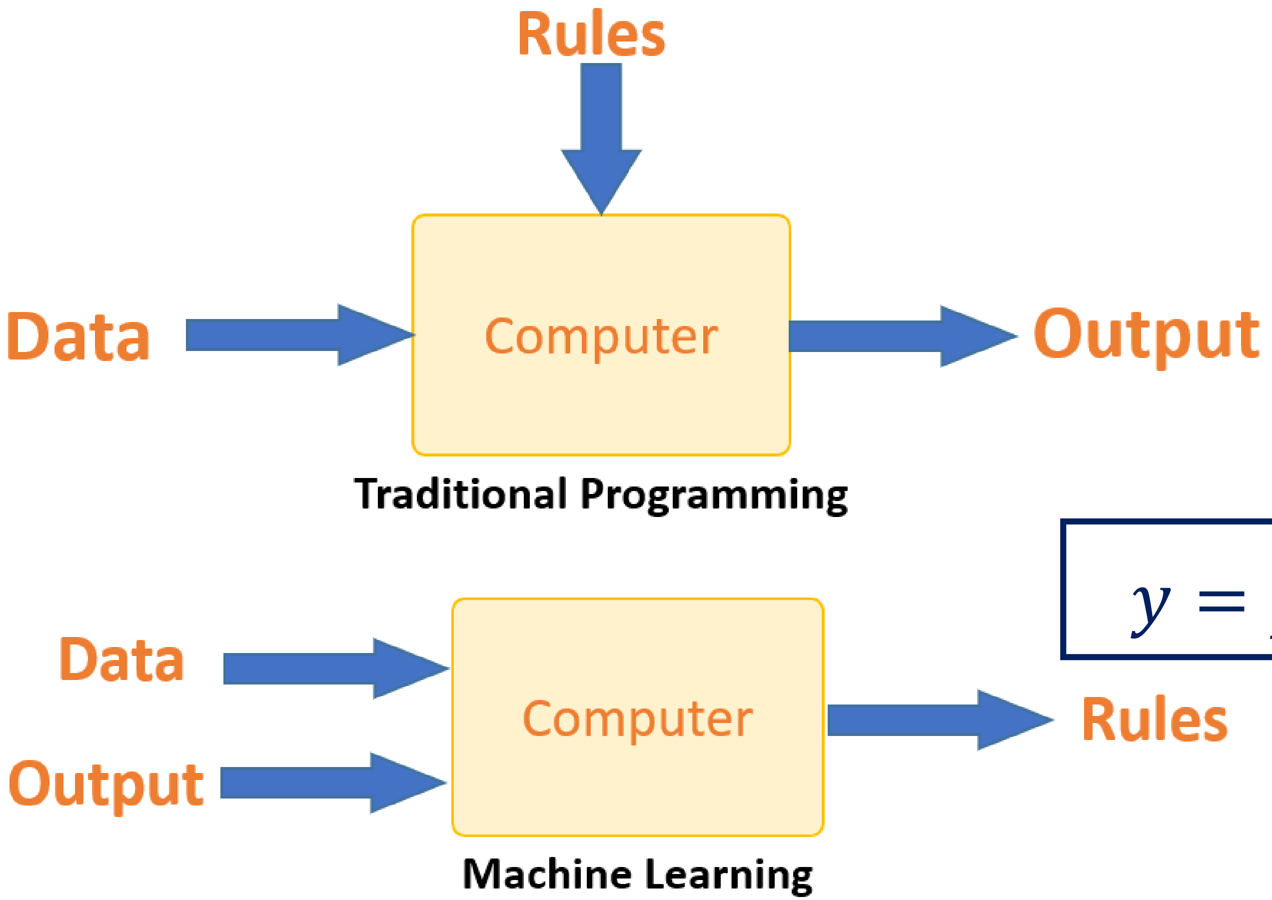
AI, ML and DL



$$y = f(x)$$

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

What are the differences between ML and traditional programming approach to let computer have intelligence?

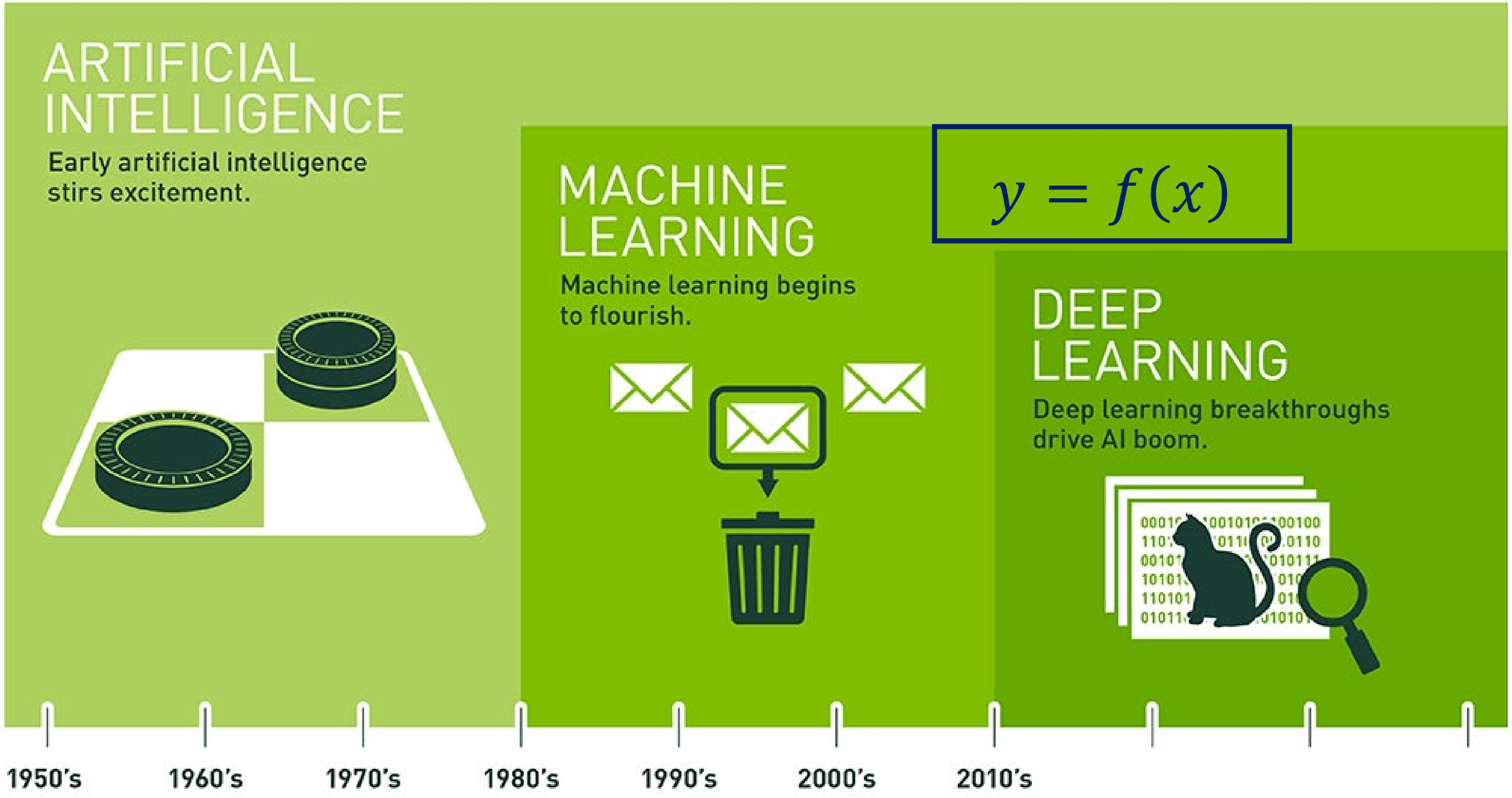


$$\boxed{y = f(x)}$$

https://www.guru99.com/machine-learning-tutorial.html

Regression vs Classification

y learned is continuous → Regression
y learned is categorical → Classification



$$\boxed{y = f(x)}$$

https://blogs.nvidia.com/blog/2016/07/29/whats-difference-artificial-intelligence-machine-learning-deep-learning-ai/

AI model is as good as its training data

1. 電腦在學習的時候，是依賴「彙整數據資料」來判斷，並沒有真正思考，如果資料來源太狹隘、不夠多元，資料寬廣度不足，電腦判斷就會出現偏差，「你跟電腦講清楚 input（輸入）、output（輸出），提供足夠的數據資料，它可以對應、學得很好，但還有很多面向 AI 做不到。」
2. AI 另一項挑戰是，它無法清楚分辨「不曾出現」與「不能出現」（無法出現）之間的區別，只是從資料統計出要學的東西，無法像人類一樣進行邏輯思辨。

https://www.managertoday.com.tw/articles/view/62902

UCLA CS 助理教授張凱崴 (台大資工， UIUC PhD， 2021 史隆研究獎)

AI model is as good as its training data

- 在製程數據上的收集，一定數據無法cover的地方。
- 而且在產線上，通常相同的錯誤不會一直出現，這是在AI model應用在工業最大的弱點。

(Ref: Dicky)

目前業界都用人+ AI 協同 decision making

1. 通常製程數據都是正常的居多，因此會先建 abnormality detection model (AE, VAE, GAN, OCSVM)，將要預測的製程數據看是否在所預測的範圍內，若在預測的範圍內才會相信預測的結果。
2. 如果建分類模型，會同時使用2~3 的不同演算法 (如: SVM or Random forest) 去預測，當結果都是一樣的，我才相信此演算法。
3. 當有發生上述的情形時，AI演算法就不會預測，會交給現場工程師去判斷，所以在工業界都說是提供人機協同的介面。

(Ref: Dicky)

目前業界都用人+ AI 協同decision making

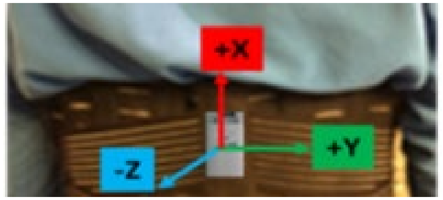
經理人雖然不一定具備 AI 方面的專業知識，但只要掌握觀念，再透過 AI 領域專才協助，也能優化系統。張凱崴指出，最直接的方法是，設計 AI 模型時，要把來源群組不同的資料分門別類測試，在測試階段讓群體多元化，並確保不同特色的使用者，用起來都沒有問題。

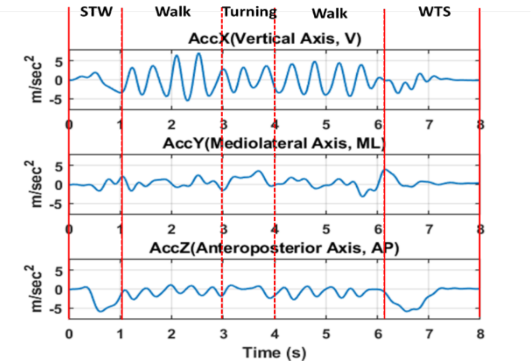
https://www.managertoday.com.tw/articles/view/62902

UCLA CS 助理教授張凱崴 (台大資工， UIUC PhD， 2021 史隆研究獎)

Case 1 – Fall risk prediction

1. Sensors and instruments to collect X and Y
2. Why X? Why X is important to predict Y?
3. Data collection protocol
4. Data pre-processing (data visualization)
5. Feature engineering
6. AI model development
7. AI模型解讀與現場臨床經驗交叉比對





在社區中篩選平衡能力或姿勢穩定度不佳的長者到醫院進行進一步檢查

Case 2 – Dementia risk prediction

1. Sensors and instruments to collect X and Y
2. Why X? Why X is important to predict Y?
3. Data collection protocol
4. Data pre-processing (data visualization)
5. Feature engineering
6. AI model development
7. AI模型解讀與現場臨床經驗交叉比對

(1) 人口學資料、疾病變項
(2) 認知量表分數：CDR， MMSE， GDS
(3) 失智相關生理訊號：EEG， Eye Tracker， HRV， GSR
(4) 血液檢查
(5) MRI， fMRI

(1)失智共同照護據點
(2) 樂齡學習中心

(1) 認知量表分數沒有差異、但血液檢查有差別的失智 vs 非失智個案，AI模型判讀結果為何? 影響AI模型決策的重要生理訊號特徵值為何? 與現場人員的過往臨床經驗之異同處為何?
(2) 認知量表分數與血液檢查都沒有差異，但fMRI結果有差別的失智 vs 非失智個案，分析AI模型解讀與現場臨床經驗交叉比對。

1. 在社區中篩選有失智風險的長者到診所或醫院進行進一步檢查
2. AI 模型有機會與基層診所及健保給付結合，**基層醫師使用AI模型結合簡易量表及生理訊號量測，並使用常規血液檢查內的數值，即可協助民眾及早診斷、就醫與治療**。

Case 3 – Spirometry reading prediction

1. Sensors and instruments to collect X and Y
2. Why X? Why X is important to predict Y?
3. Data collection protocol
4. Data pre-processing (data visualization)
5. Feature engineering
6. AI model development
7. AI模型解讀與現場臨床經驗交叉比對

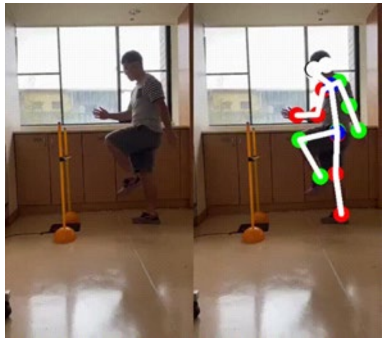
R5Hz	R10Hz	R15Hz	R20Hz	R25Hz	R35Hz	X5Hz	X10Hz	X15Hz	X20Hz	X25Hz
X1	X2	X3	X4	X5	X6	X7	X8	X9	X10	X11

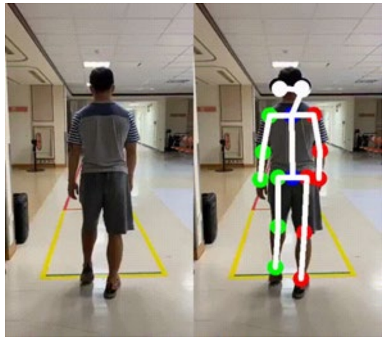
X35Hz	Z5Hz	VT	Rc	Rp	Fres	AX	R5Hz-R20Hz	Gender	Age
X12	X13	X14	X15	X16	X17	X18	X19	X20	X21

協助判讀無法完成吹氣的病患之COPD與SAD病程

Other elderly-care application cases

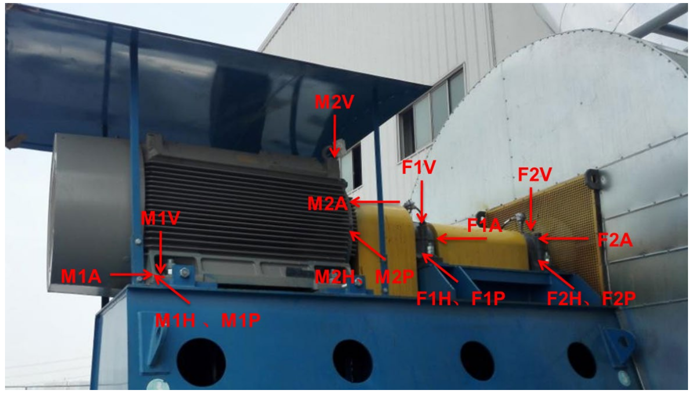
1. Sensors and instruments to collect X and Y
2. Why X? Why X is important to predict Y?
3. Data collection protocol
4. Data pre-processing (data visualization)
5. Feature engineering
6. AI model development
7. AI模型解讀與現場臨床經驗交叉比對



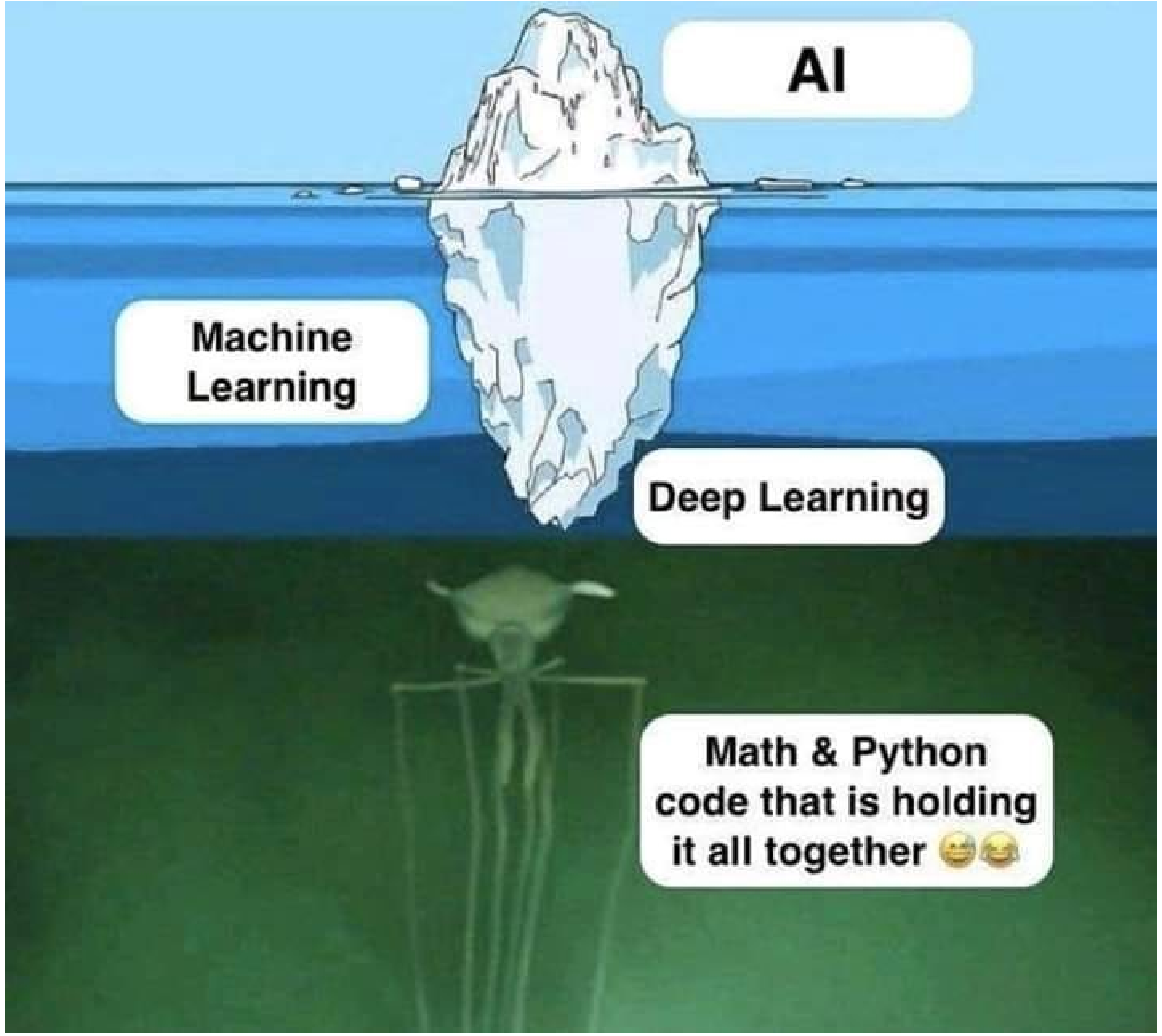


Manufacturing application cases

1. Sensors and instruments to collect X and Y
2. Why X? Why X is important to predict Y?
3. Data collection protocol
4. Data pre-processing (data visualization)
5. Feature engineering
6. AI model development
7. AI模型解讀與現場經驗交叉比對



Warning of math and coding

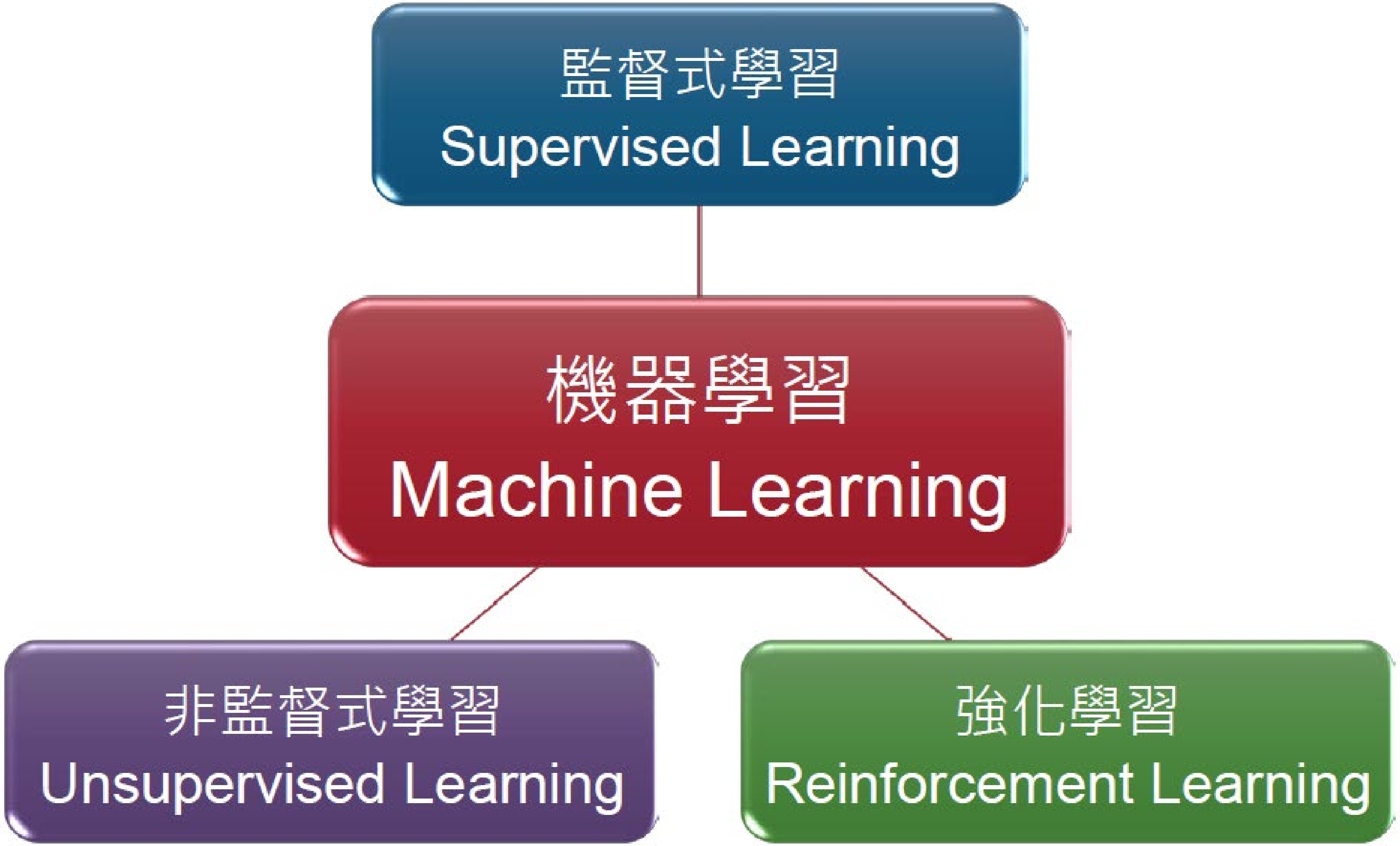


- Statistics
- Linear algebra
- Non-linear optimization

https://www.facebook.com/257118421832544/photos/a.320314752179577/540486140162436/

Python coding

Three ML/DL strategies



(Ref: 林柏江 教授)

Notations

x_i x_i^n x^n $\hat{y}^n$

No	age	t1	t2	t3	t4	t5	t6	time	Step frequency	n1	n2	n3	n4	n5	n6	px	py	pz	Steps	Gender	TUG	y1	BBS	y2
1	70	1.76	2.64	6.24	7.02	10	12.8	11	2.285	80	120	282	317	453	575	11.67	1.809	-1.99	13	F	11	0	26	0
2	86	1.64	2.6	5.82	7.27	10.4	12.6	11	1.934	75	118	263	328	470	570	11.14	2.302	-4.651	12	F	11	0	24	0
3	76	1.76	2.93	6.27	7.04	10.3	12.8	11	2.109	80	133	283	318	465	575	11.53	2.169	-3.253	14	F	11	0	22	1
4	70	2.38	3.29	5.58	6.47	9.02	10.4	8	2.461	108	149	252	292	407	468	11.6	1.838	-3.138	12	F	8	0	24	0
5	66	3.09	4.07	6.6	7.4	10.2	12.1	9	2.461	140	184	298	334	462	545	11.55	2.531	-2.742	12	F	9	0	26	0
6	79	1.76	2.91	5.87	6.6	10.2	12.8	11	2.109	80	132	265	298	462	575		1.788	-1.349	13	F	11	0	26	0
7	85	1.2	2.33	5.42	8.31	12.1	17.2	16	2.988	55	106	245	375	545	775		2.203	-4.89	17	M	16	1	18	1
8	81	1.64	2.93	5.98	7.47	10.9	13.6	12	1.758	75	133	270	337	493	615	11.1	2.667	-4.594	10	F	12	0	24	0
9	82	0.64	1.47	4.76	5.76	9.36	11.6	11	2.109	30	67	215	260	422	525	11.26	4.1	-2.693	14	M	11	0	24	0
10	69	1.64	2.49	5.02	5.98	9.82	12.6	11	2.637	75	113	227	270	443	570	11.27	3.292	-3.522	13	F	11	0	20	1
11	84	0.64	1.4	5.67	7.29	11.5	14.6	14	1.934	30	64	256	329	520	660	11.53	2.335	-2.999	15	M	14	1	26	0
12	69	1.09	1.98	5	5.62	8.38	10.1	9	2.109	50	90	226	254	378	455	11.15	1.919	-4.608	11	M	9	0	26	0
13	73	1.09	2.13	6.78	8.38	12.4	17.1	16	3.691	50	97	306	378	558	770	11.46	2.264	-3.333	16	F	16	1	14	1
14	81	0.64	1.87	9.24	11.2	19	22.6	22	1.934	30	85	417	507	857	1020	11.58	2.511	-2.157	27	M	22	1	24	0
15	80	0.76	1.71	3.98	5	7.58	9.76	9	2.109	35	78	180	226	342	440	11.33	2.821	-3.595	10	M	9	0	26	0
16	88	0.98	2.13	6.31	7.44	11.5	14	13	1.934	45	97	285	336	518	630	11.38	2.498	-3.702	16	M	14	1	26	0
17	81	1.09	2.09	4.18	5.16	7.76	10.1	9	2.285	50	95	189	233	350	455	11.21	2.241	-4.337	10	M	9	0	28	0
18	76	1.76	2.64	5.87	6.98	9.98	12.8	11	1.406	80	120	265	315	450	575	11.33	2.679	-3.736	10	M	11	0	26	0
19	69	0.36	3.76	13.3	16.7	24.2	29.4	29	3.691	17	170	598	753	1090	1322	11.31	1.361	-4.171	28	F	29	1	10	1
20	75	1.98	2.93	5.98	7.91	12.2	15	13	1.934	90	133	270	357	551	675	11.5	2.202	-1.495	14	M	13	0	28	0
21	87	1.53	3.2	10.9	13.8	21.3	26.5	25	2.9	70	145	492	624	960	1195	11.6	2.199	-2.54	19	F	25	1	16	1
22	72	0.2	1.02	3.36	4.11	7.42	10.2	10	1.758	10	47	152	186	335	460	11.52	2.658	-2.081	9	M	10	0	28	0
23	109	0.64	1.93	5.04	5.71	9.13	10.6	10	2.285	30	88	228	258	412	480	11.51	2.056	-3.158	15	F	10	0	28	0

Mechanism for computer to learn from data

- Define a function to be learned: $y^n = f(x^n)$
- Define a loss function $\mathcal{L}(f)$ to describe the error between y^n and $\hat{y}^n$
- Find the optimal parameters that minimize $\mathcal{L}(f)$

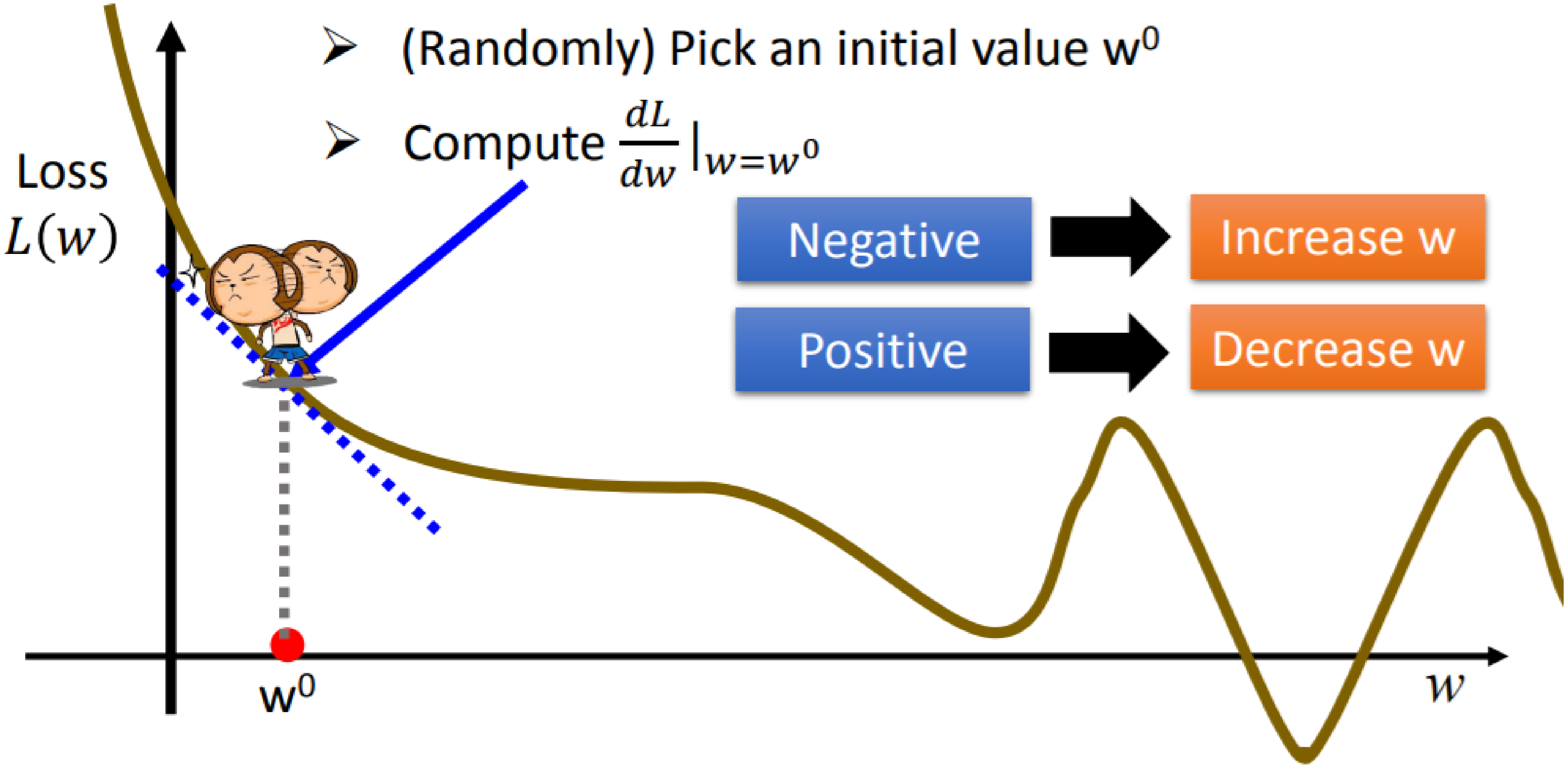
ML model (1) – Linear regression

- Linear model: $y^n = \sum_i (w_i x_i^n) + b$
- Loss function: $L(w，b) = \sum_{n=1}^{N} (\hat{y}^n - y^n)^2 = \sum_{n=1}^{N} (\hat{y}^n - (\sum_i (w_i x_i^n) + b))^2$
- Find the optimal parameters that minimize loss: $\arg \min_{w，b} L(w，b)$

Use gradient decent to find optimal parameters

$$w^* = arg \min_w L(w)$$

- Consider loss function $L(w)$ with one parameter w:
  - (Randomly) Pick an initial value w^0
  - Compute $\frac{dL}{dw}|_{w=w^0}$



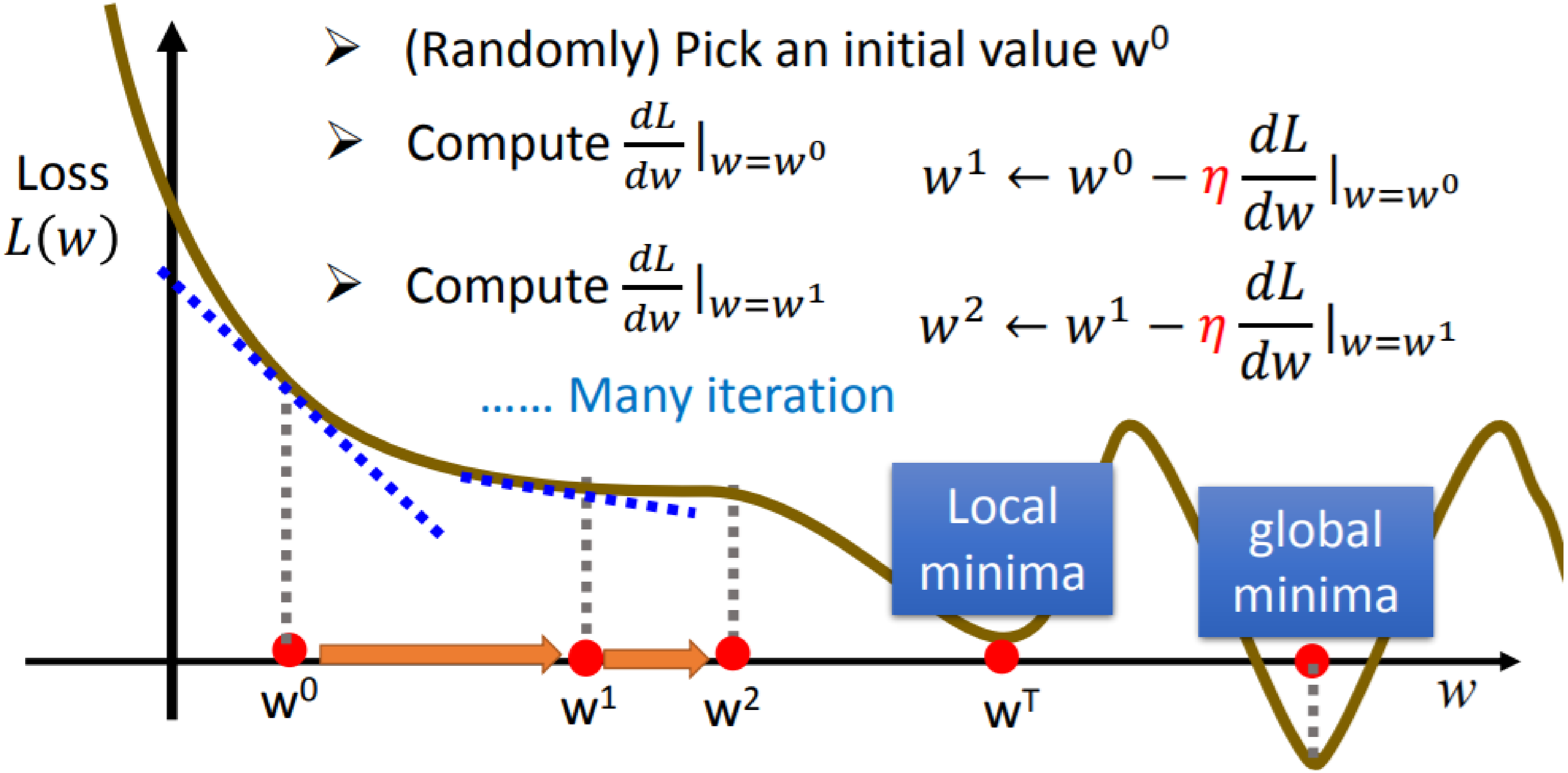
Reference: 李弘毅 ML Lecture 1 https://youtu.be/CXgbekl66jc

Gradient decent

$$w^* = arg \min_w L(w)$$

- Consider loss function $L(w)$ with one parameter w:
  - ➢ (Randomly) Pick an initial value w^0
  - ➢ Compute $\frac{dL}{dw}|_{w=w^0}$ $\qquad w^1 \leftarrow w^0 - \eta \frac{dL}{dw}|_{w=w^0}$
  - ➢ Compute $\frac{dL}{dw}|_{w=w^1}$ $\qquad w^2 \leftarrow w^1 - \eta \frac{dL}{dw}|_{w=w^1}$

…… Many iteration



Gradient decent to find two parameters w^* and b^*

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

  - ➢ (Randomly) Pick an initial value w^0, b^0

  - ➢ Compute $\frac{\partial L}{\partial w}|_{w=w^0,b=b^0}, \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$

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

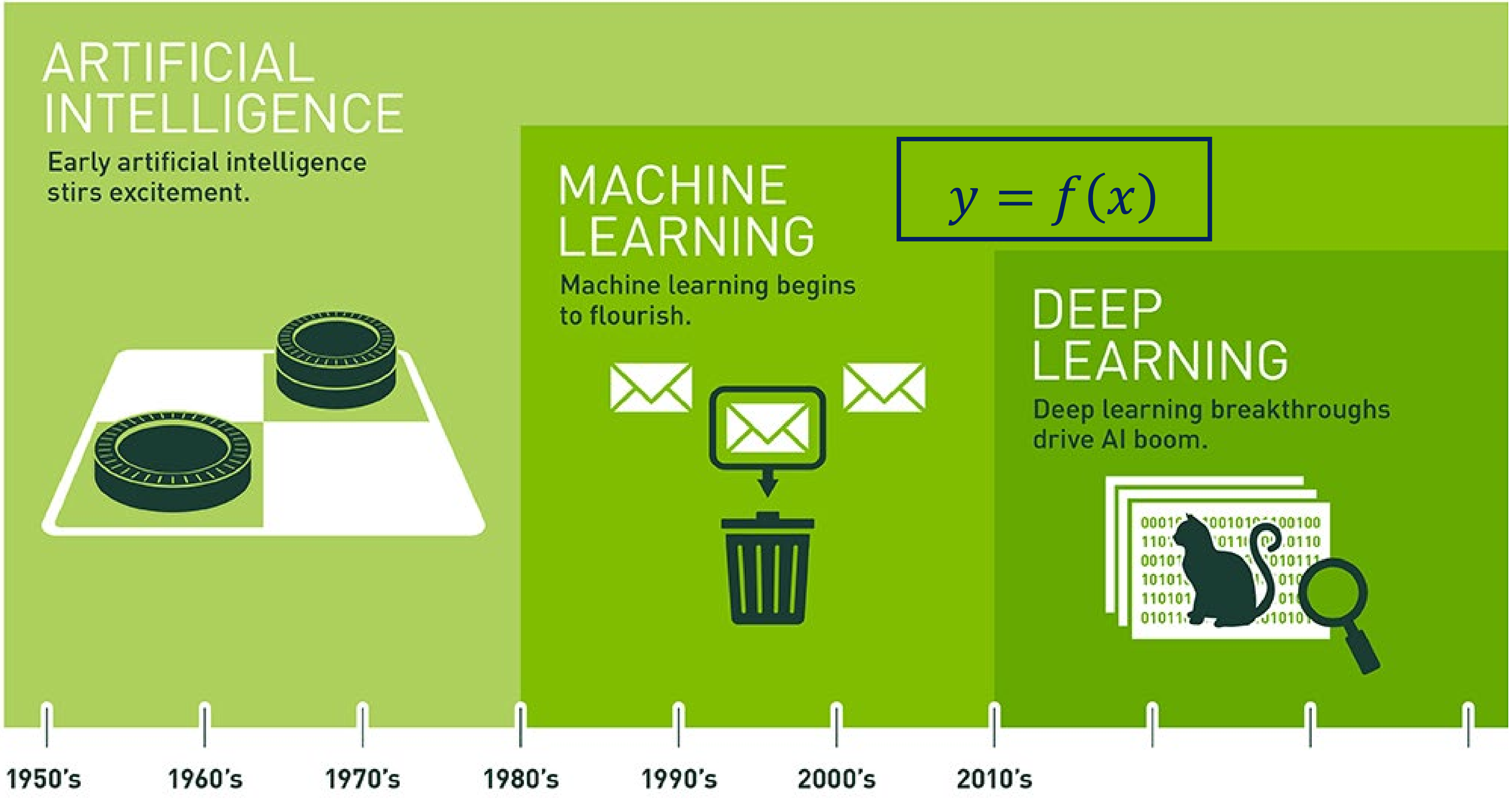
$$w^1 \leftarrow w^0 - \eta \frac{\partial L}{\partial w}|_{w=w^0,b=b^0} \qquad b^1 \leftarrow b^0 - \eta \frac{\partial L}{\partial b}|_{w=w^0,b=b^0}$$

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

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

Reference: 李弘毅 ML Lecture 1 https://youtu.be/CXgbekl66jc

What are the difference between ML and DL?



Jeffery Hinton

Geoffrey Hinton spent 30 years hammering away at an idea most other scientists dismissed as nonsense. Then, one day in 2012, he was proven right. Canada's most influential thinker in the field of artificial intelligence is far too classy to say I told you so

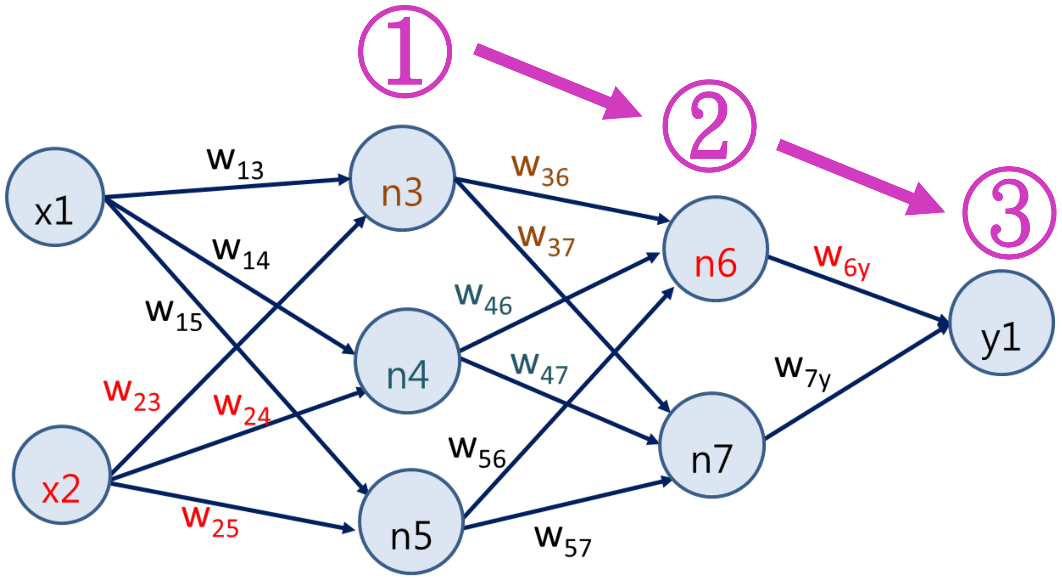
https://torontolife.com/tech/ai-superstars-google-facebook-apple-studied-guy/

For more than 30 years， Geoffrey Hinton hovered at the edges of artificial intelligence research，an outsider clinging to a simple proposition: that computers could think like humans do—using intuition rather than rules.

Geoffrey Hinton 多年來堅持着一個簡單的觀點：電腦可以像人類一樣思考-用直覺而不是規則。Hinton 一直好奇的是，電腦能不能像人類大腦一樣的工作：信息通過一個巨大的，由神經元圖譜連接起來的細胞網絡傳播，在多達十億條的路徑上發射、連接和傳輸。

Machine learning model (2) – Deep learning

Define a function to be learned: $y^n = f(x^n)$



①

$$n_3 = \sigma(x_1 * w_{13} + x_2 * w_{23} + b_3)$$
$$n_4 = \sigma(x_1 * w_{14} + x_2 * w_{24} + b_4)$$
$$n_5 = \sigma(x_1 * w_{15} + x_2 * w_{25} + b_5)$$

②

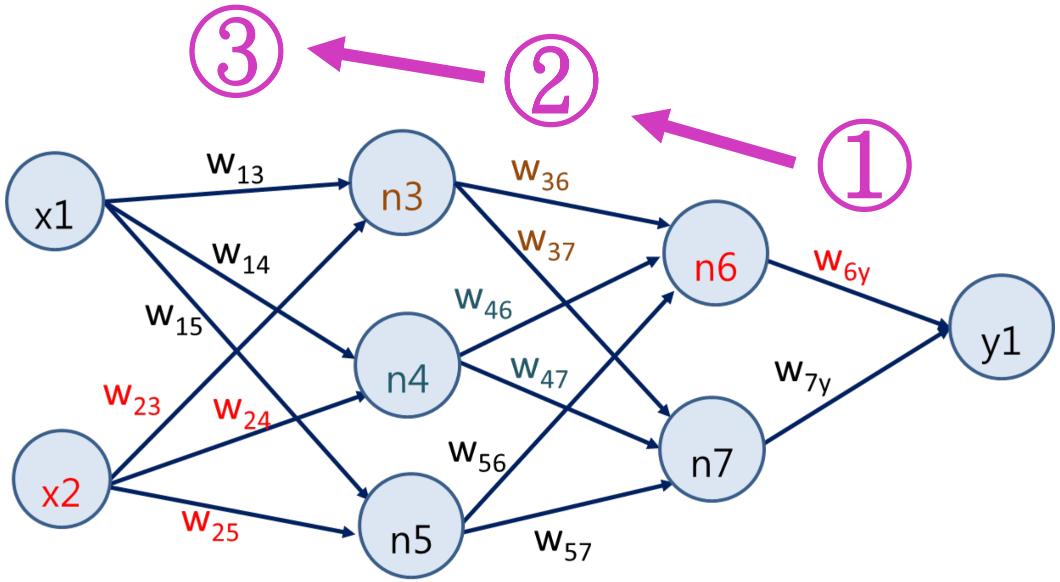
$$n_6 = \sigma(n_3 * w_{36} + n_4 * w_{46} + n_5 * w_{56} + b_6)$$
$$n_7 = \sigma(n_3 * w_{37} + n_4 * w_{47} + n_5 * w_{57} + b_7)$$

③

$$y_1 = \sigma(n_6 * w_{6y} + n_7 * w_{7y} + b_y)$$

Machine learning model (2) – Deep learning

Use gradient decent to find optimal parameters



$$w_i \leftarrow w_i - \eta \frac{\partial e}{\partial w_i}$$

$$L = g(y - \hat{y}) \qquad y = \sigma(n_6 * w_{6y} + n_7 * w_{7y} + b_y)$$

①

$$w_{6y} \leftarrow w_{6y} - \eta \frac{\partial L}{\partial w_{6y}} \qquad \frac{\partial L}{\partial w_{6y}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{6y}}$$

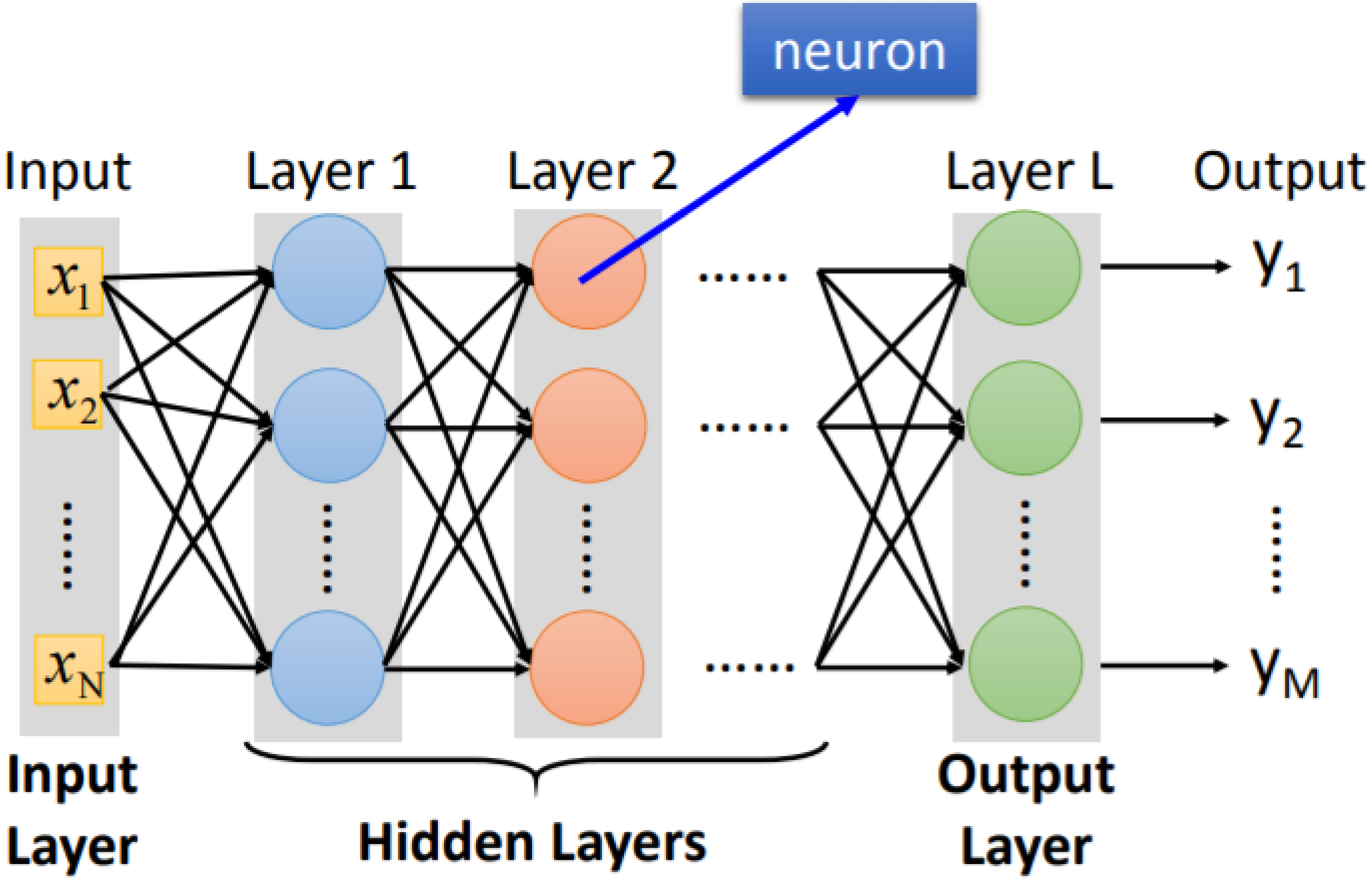
$$w_{7y} \leftarrow w_{7y} - \eta \frac{\partial L}{\partial w_{7y}} \qquad \frac{\partial L}{\partial w_{7y}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial w_{7y}}$$

②

$$w_{57} \leftarrow w_{57} - \eta \frac{\partial L}{\partial w_{57}} \qquad \frac{\partial L}{\partial w_{57}} = \frac{\partial L}{\partial y}\frac{\partial y}{\partial n_7}\frac{\partial n_7}{\partial w_{57}}$$

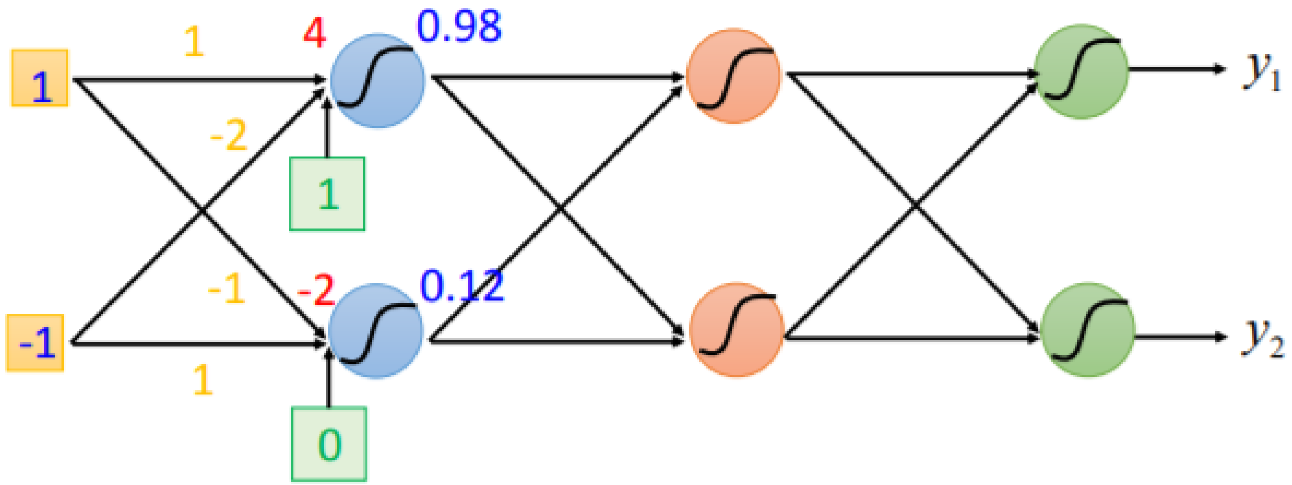
$$n_7 = f(n_3 * w_{37} + n_4 * w_{47} + n_5 * w_{57} + b_7)$$

MLP is a fully connected feedforward network



Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw

Fully connected feed forward network is implemented as matrix operation



$$y = \sigma(w \cdot x + b)$$

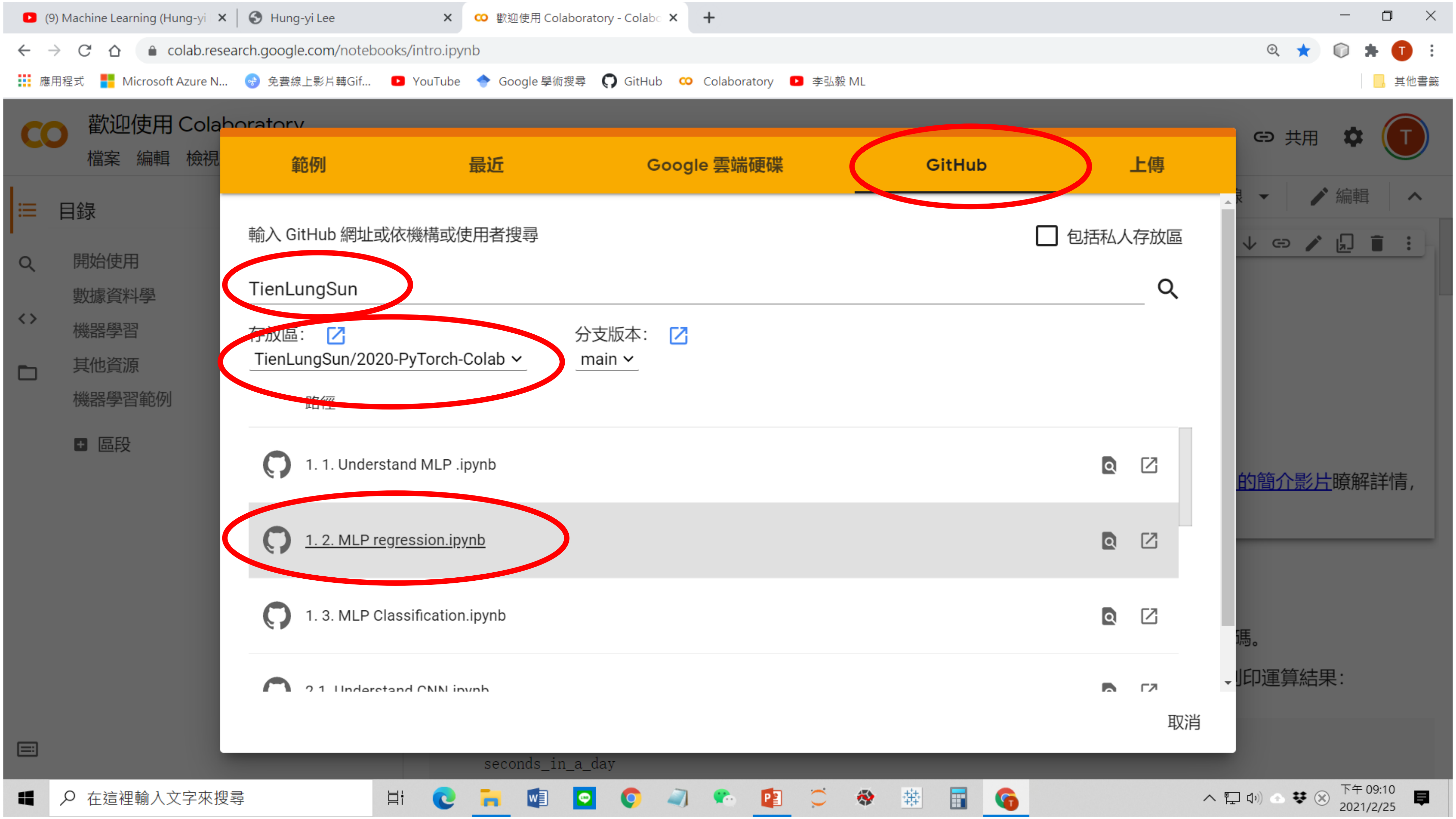
$$\sigma\Bigg(\underbrace{\begin{bmatrix} 1 & -2 \\ -1 & 1 \end{bmatrix}\begin{bmatrix} 1 \\ -1 \end{bmatrix} + \begin{bmatrix} 1 \\ 0 \end{bmatrix}}_{\begin{bmatrix} 4 \\ -2 \end{bmatrix}}\Bigg) = \begin{bmatrix} 0.98 \\ 0.12 \end{bmatrix}$$

Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw

Practice

- Run “1.1 Matrix operation.ipynb"

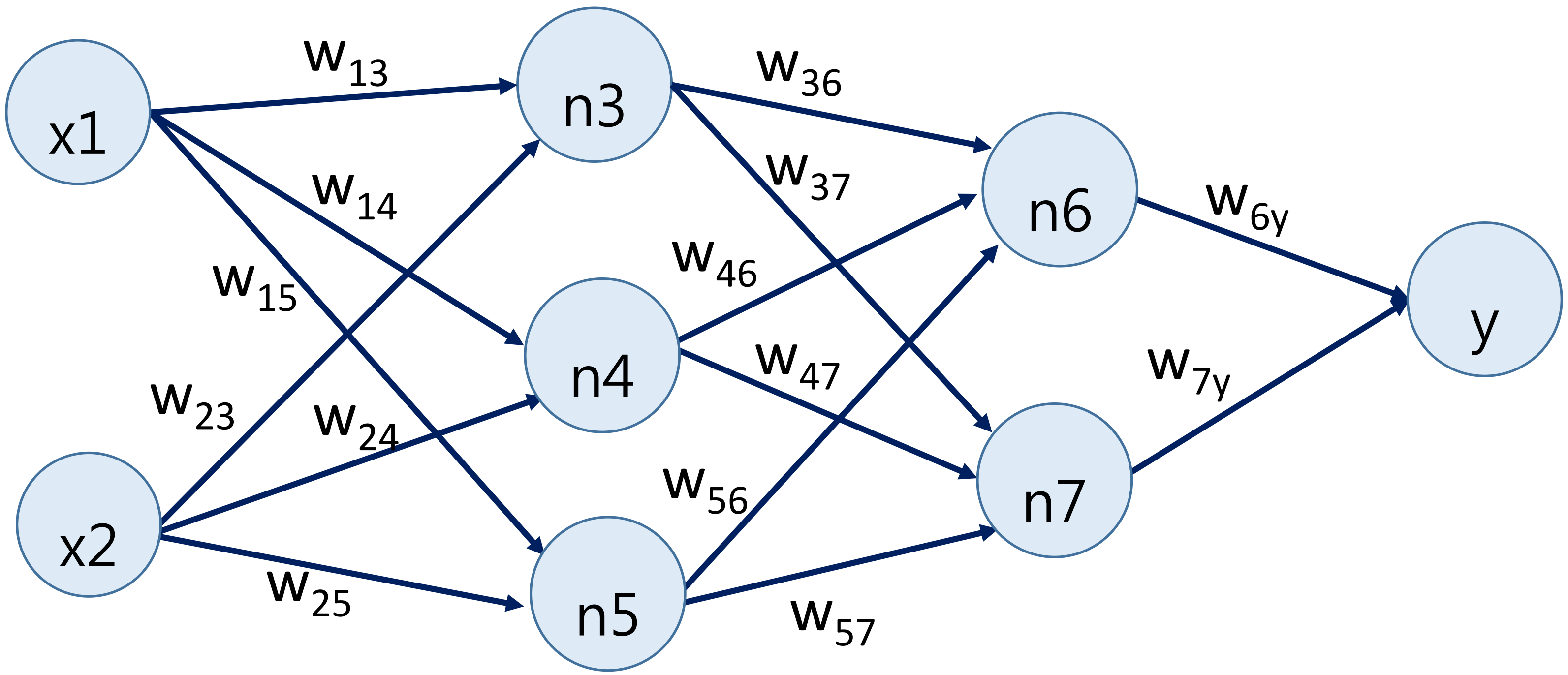
Run GitHub PyTorch code from Colab



Matrix operation

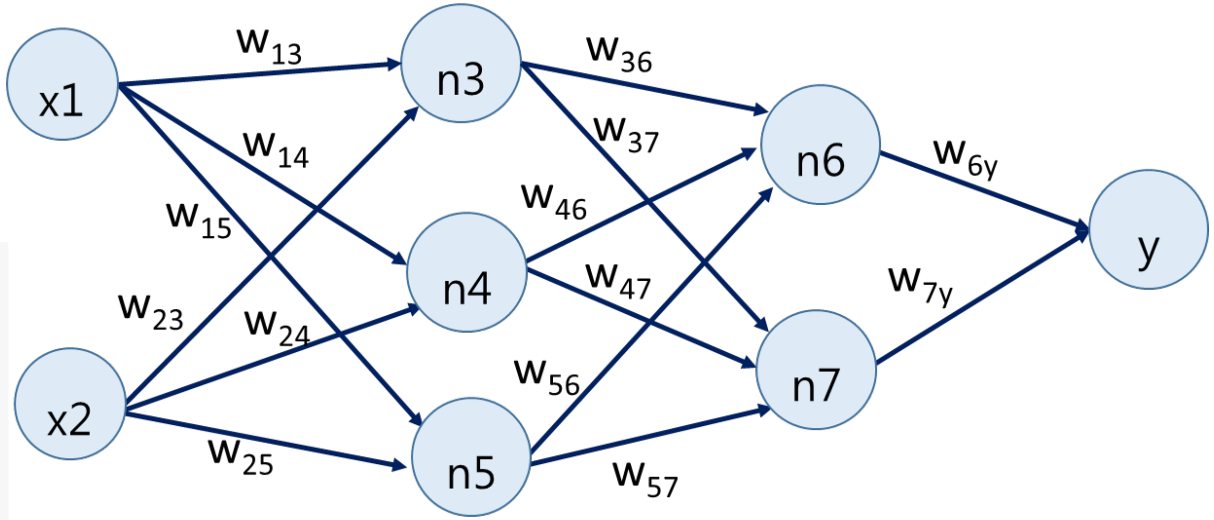
```
MyNet  =  nn.Sequential(
    nn.Linear(2,  3),
    nn.Linear(3,  2),
    nn.Linear(2,  1)
)
```

$$\begin{array}{cc} & \begin{array}{cc} x_1 & x_2 \end{array} \\ \vec{X} = & \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix} \end{array}$$



$$\hat{y} = \begin{bmatrix} 7 \\ 12 \\ 40 \end{bmatrix}$$

Matrix operation




```
for  param  in  MyNet.parameters():
        if  param.requires_grad:
                print(param.data)
```

```
tensor([[ 0.4727, -0.5188],
        [-0.5681, -0.6032],
        [-0.0252, -0.3011]])
tensor([-0.6986, -0.6602, -0.4860])
tensor([[-0.5549,  0.2550,  0.4584],
        [ 0.2930,  0.0849, -0.3146]])
tensor([0.1677, 0.0736])
tensor([[ 0.4106, -0.3618]])
tensor([-0.2270])
```

$$\begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}$$

$$[b_3 \quad b_4 \quad b_5]$$

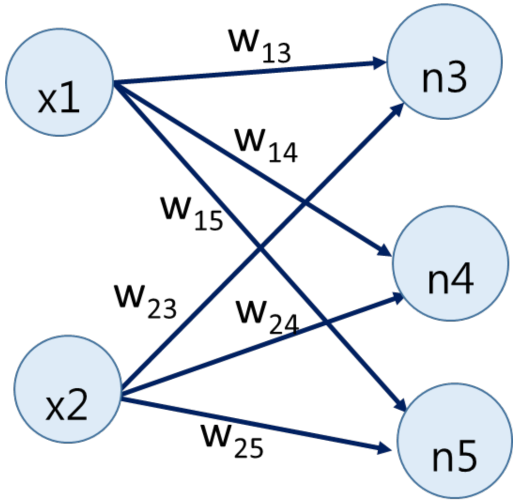
$$\begin{bmatrix} w_{36} & w_{46} & w_{56} \\ w_{37} & w_{47} & w_{57} \end{bmatrix}$$

$$[b_6 \quad b_7]$$

$$[w_{6y} \quad w_{7y}]$$

$$[b_y]$$

$$\vec{X} = \begin{array}{c} \begin{array}{cc} x_1 & x_2 \end{array} \\ \begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix} \end{array}$$



$$\underbrace{\begin{bmatrix} 1 & 2 \\ 2 & 3 \\ 10 & 5 \end{bmatrix} \begin{bmatrix} w_{13} & w_{14} & w_{15} \\ w_{23} & w_{24} & w_{25} \end{bmatrix}} + [b_3 \quad b_4 \quad b_5]$$

$$\underbrace{\begin{bmatrix} k_3^1 & k_4^1 & k_5^1 \\ k_3^2 & k_4^2 & k_5^2 \\ k_3^3 & k_4^3 & k_5^3 \end{bmatrix} + \begin{bmatrix} b_3 & b_4 & b_5 \\ b_3 & b_4 & b_5 \\ b_3 & b_4 & b_5 \end{bmatrix}}$$

$$\boxed{\begin{bmatrix} n_3^1 & n_4^1 & n_5^1 \\ n_3^2 & n_4^2 & n_5^2 \\ n_3^3 & n_4^3 & n_5^3 \end{bmatrix}}$$

Use Excel to verify

```
W1 = MyNet[0].weight
b1 = MyNet[0].bias
print(W1, W1.shape, b1)
```

```
Parameter containing:
tensor([[ 0.4727, -0.5188],
        [-0.5681, -0.6032],
        [-0.0252, -0.3011]],
tensor([-0.6986, -0.6602, -0.4860], r
```

$$\begin{bmatrix} w_{13} & w_{23} \\ w_{14} & w_{24} \\ w_{15} & w_{25} \end{bmatrix}$$

```
#Calculate n3, n4, n5
HiddenLayer1 = MyNet[0](tensorX)
print(HiddenLayer1)
```

```
tensor([[-1.2635, -2.4348, -1.1135],
        [-1.3097, -3.6061, -1.4398],
        [ 1.4340, -9.3577, -2.2441]],
```

```
#Calculate n3, n4, n5 using Pytorch matrix operation
HiddenLayer1 = tensorX.mm(torch.transpose(W1, 1, 0)) + b1
print(HiddenLayer1)
```

```
tensor([[-1.2635, -2.4348, -1.1135],
        [-1.3097, -3.6061, -1.4398],
        [ 1.4340, -9.3577, -2.2441]], grad_fn=<AddBackward0>)
```

```
#Calculate n6, n7 using PyTorch matrix operation
W2 = MyNet[1].weight
b2 = MyNet[1].bias
HiddenLayer2 = HiddenLayer1.mm(torch.transpose(W2, 1, 0)) +b2
print(HiddenLayer2)
```

```
tensor([[-0.2625, -0.1530],
        [-0.6852, -0.1632],
        [-4.0429,  0.4054]], grad_fn=<AddBackward0>)
```

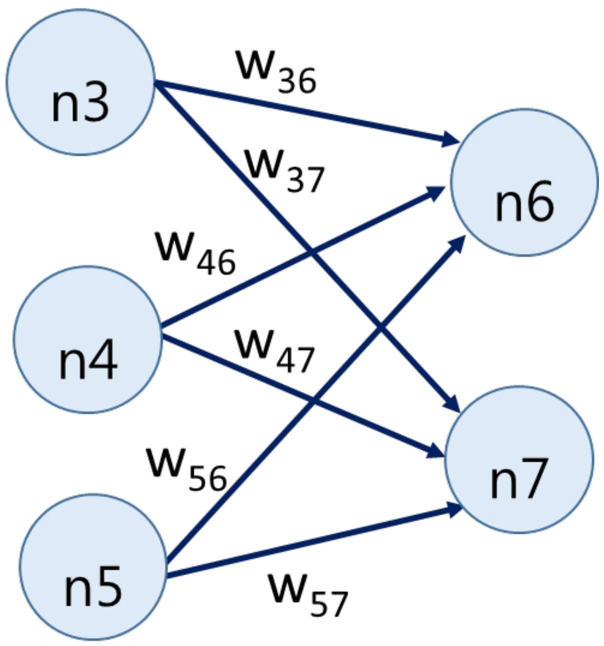
$$\begin{bmatrix} n_3^1 & n_4^1 & n_5^1 \\ n_3^2 & n_4^2 & n_5^2 \\ n_3^3 & n_4^3 & n_5^3 \end{bmatrix}$$

$$\underbrace{\begin{bmatrix} -1.2635 & -2.4348 & -1.1135 \\ -1.3097 & -3.6061 & -1.4398 \\ 1.4340 & -9.3577 & -2.2441 \end{bmatrix} \begin{bmatrix} w_{36} & w_{37} \\ w_{46} & w_{47} \\ w_{56} & w_{57} \end{bmatrix}} + \begin{bmatrix} b_6 & b_7 \end{bmatrix}$$

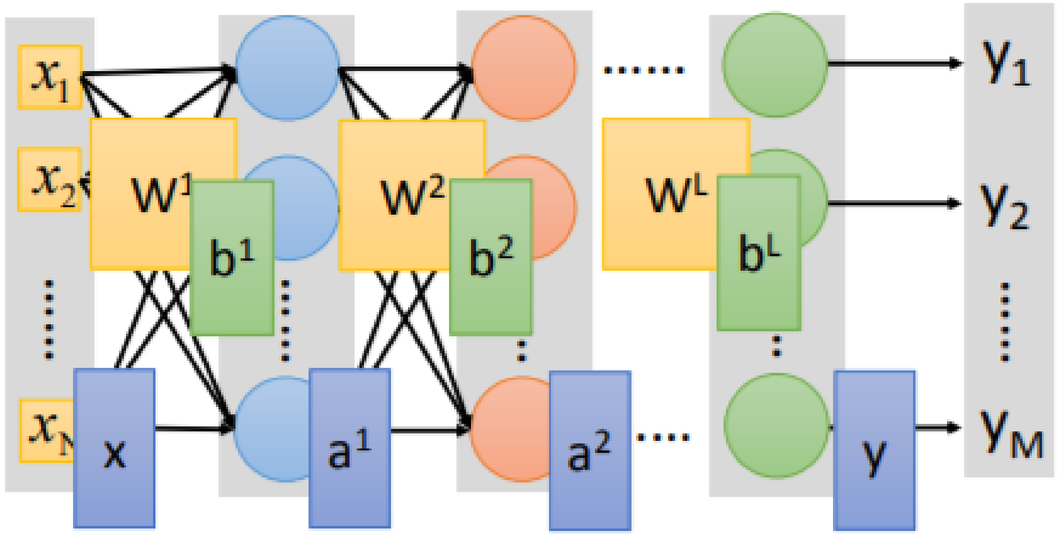
$$\underbrace{\begin{bmatrix} k_6^1 & k_7^1 \\ k_6^2 & k_7^2 \\ k_6^3 & k_7^3 \end{bmatrix} + \begin{bmatrix} b_6 & b_7 \\ b_6 & b_7 \\ b_6 & b_7 \end{bmatrix}}$$

$$\begin{bmatrix} n_6^1 & n_7^1 \\ n_6^2 & n_7^2 \\ n_6^3 & n_7^3 \end{bmatrix}$$

Use Excel to verify



Use parallel computing to speed up matrix operation



$$y = f(x)$$

Using parallel computing techniques to speed up matrix operation

$$= \sigma(W^L \cdots \sigma(W^2\, \sigma(W^1 x + b^1) + b^2) \cdots + b^L)$$

Reference: 李弘毅 ML Lecture 6 https://youtu.be/Dr-WRlEFefw

Use parallel computing to speed up matrix operation

In [2]:
```
if(torch.cuda.is_available()):
    device = torch.device("cuda")
    print(device, torch.cuda.get_device_name(0))
else:
    device= torch.device("cpu")
    print(device)
```

```
cuda Tesla P100-PCIE-16GB
```

```
tensorX = torch.FloatTensor(trainX).to(device)
tensorY_hat = torch.FloatTensor(trainY_hat).to(device)
print(tensorX.shape, tensorY_hat.shape)
```

```
torch.Size([128, 2]) torch.Size([128, 1])
```

```
conv1_out = conv1(imageTensor.to(device))
conv1_out.shape
#output image (feature map) has 64 channels
```

```
torch.Size([1, 64, 55, 55])
```