Days on Jupyter

政大應數蔡炎龍 yenlung

Taipei.py

環境設定

基本上用 Anaconda，裝 Python 3 的版本

Story 1 • Python

Apple][的時代

寫程式都是在玩耍

高中時代

第二關起就比較困難了

用的基本上都是 BASIC

所以用 BASIC 做所有事

當那美好的時代過了

我想找一個取代 BASIC 的語言

遇見。Python

- 212. Mr. Paolo G. Cantore, Ludwigshafen, Germany
- 213. Wilhelm G. Fitzpatrick, Seattle, USA
- 214. Dr. Douglas K. Cooper, Tigard, Oregon
- 215. Mr. Wolfgang H Feix, Hilzingen, Germany
- 216. Mr. Stuart M Ford, Grafton, WI
- 217. Mr. Sreeni R Nair, Parlin, New Jersey
- 218. Dr. Michael R Haggerty, Cambridge, MA
- 219. Mr. Thomas M. G. Bennett, Boone, North Carolina
- 220. Mr. Joseph T Bore, Jr, Hoboken, NJ
- 221. Mr. Yen-lung Tsai, Irvine, CA
- 222. Mr. Conrad Schneiker, Austin, TX
- 223. Mr. Ron West, ACT, Australia
- 224. Dr. Luby Liao, San Diego, CA
- 225. Jameson A Quinn, Seattle, wa
- 226. Peter Kropf, Sunnyvale, CA
- 227. Jonathan McLin, Tempe, AZ
- 228. Nils Fischbeck, Stralsund, Germany
- 229. Mike Howard, Cobleskill, NY
- 230. Dr. Alex Martelli, BO, Italia
- 231. Julio Carrera, Boston, MA
- 232. Mr. David Walter Schere, Annapolis , MD

交了 50 美元的年費!

2000 年時, Python Software Activity 只有 271 人

“Python is the second best language for anything.”

%pylab inline

簡直就是 Matlab 上身

雖然我們被警告
不要這樣

“No Pylab Thanks”

https://goo.gl/8i4nVb

一個問題是

這三段指令是一樣的。

```
> plot(randn(100))
```

```
> plt.plot(randn(100))
```

```
> plt.plot(np.random.randn(100))
```

重點是有方便的試驗場

[例子] 均數返還 (mean reversion)

我們令人驚訝的表現、或一時失手， 其實長期看來遲早會被「打回原型」？

分析洛杉磯天使隊

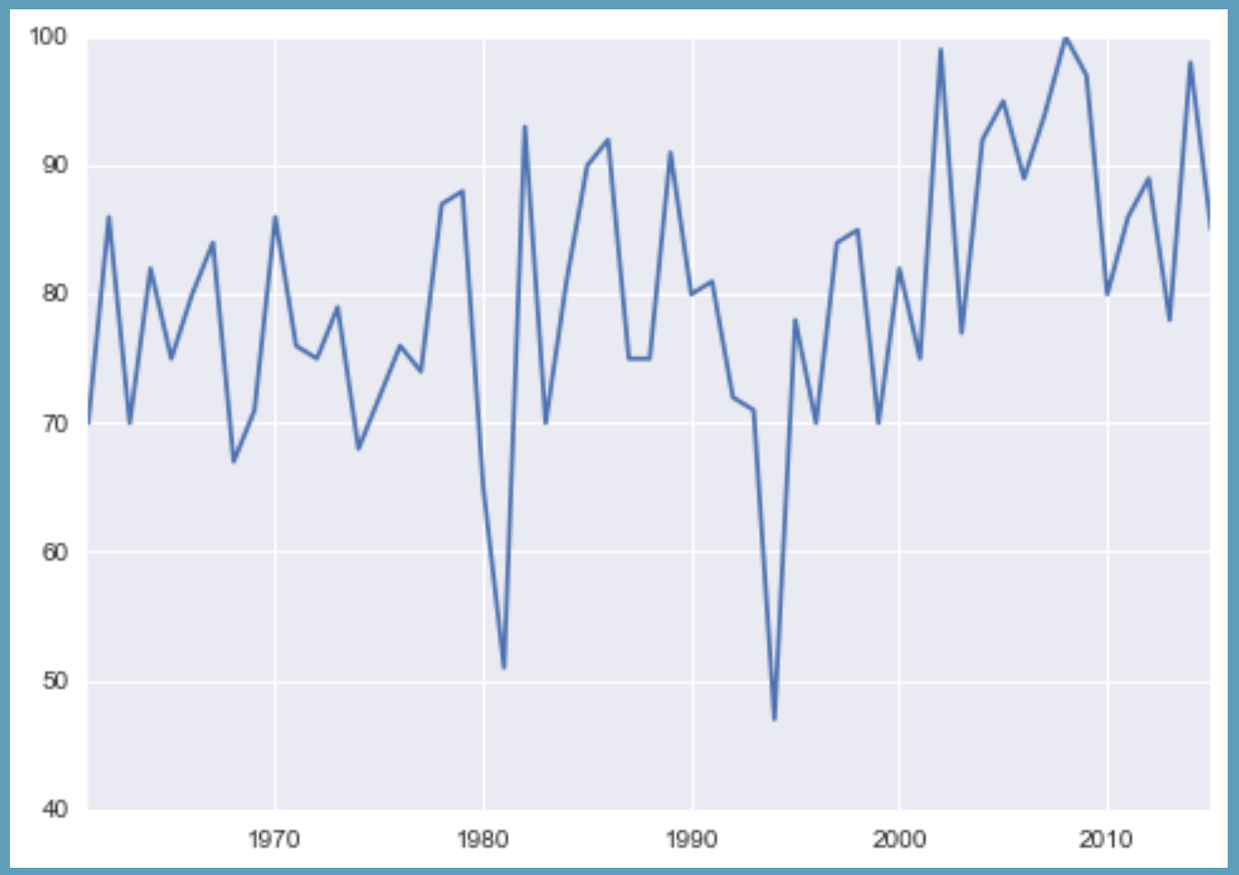
我們還是來個 `%pylab inline`，而 `seaborn` 套件只是讓圖漂亮。

```
%pylab inline
import pandas as pd
import seaborn as sns
```

天使隊 1961-2015 勝場數

```
x = arange(1961, 2016)

y = [70, 86, 70, 82, 75, 80, 84, 67, 71,
     86, 76, 75, 79, 68, 72, 76, 74, 87,
     88, 65, 51, 93, 70, 81, 90, 92, 75,
     75, 91, 80, 81, 72, 71, 47, 78, 70,
     84, 85, 70, 82, 75, 99, 77, 92, 95,
     89, 94, 100, 97, 80, 86, 89, 78, 98,
     85]
```

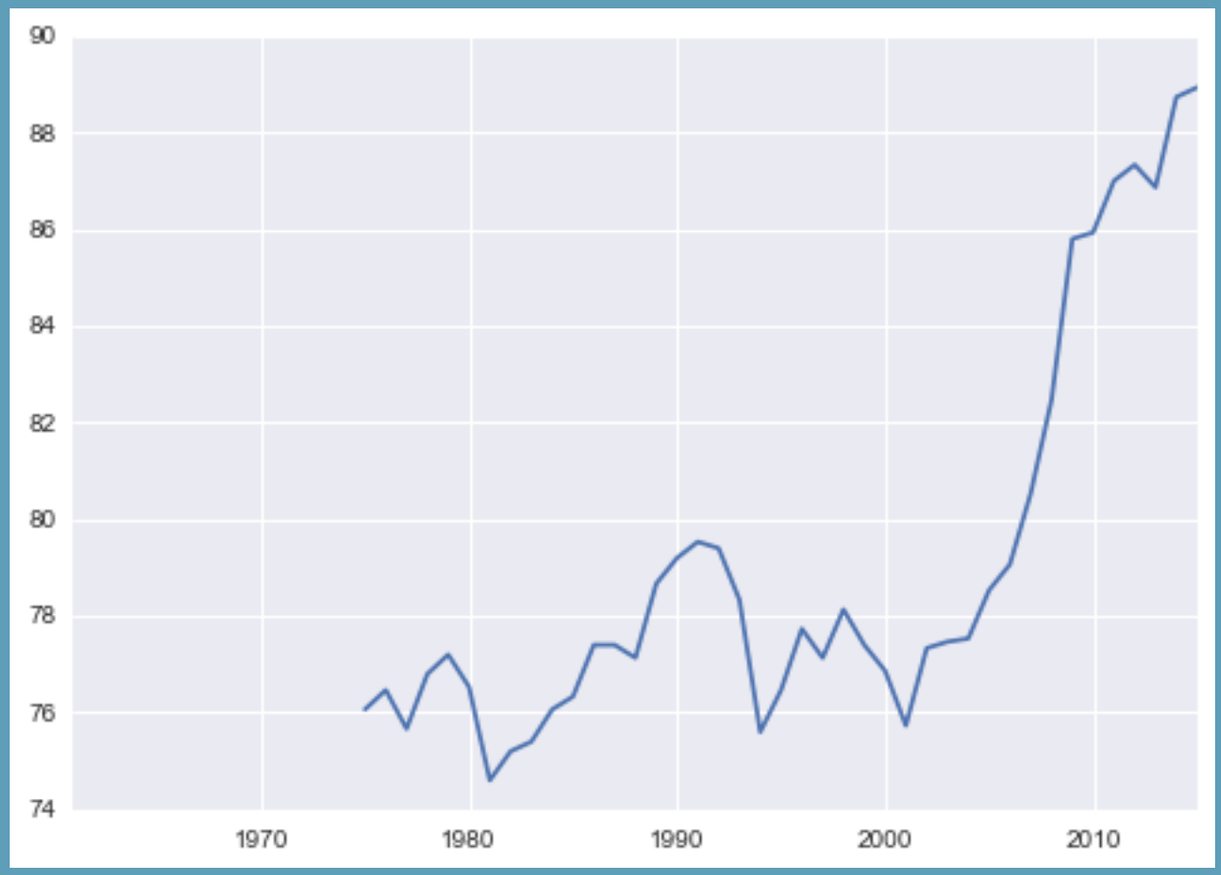



```
angels = pd.Series(y, index=x)

angels.plot()
```

怎麼看平均值的變動?

也許我們可以每 15 年算平均...




```
r = angels.rolling(window=15)

r.mean().plot()
```

Story 2 • Jupyter

事到如今...

大家可能都知道, Jupyter 之前是叫 IPython, 因為其實 IPython 不一定要是 Python 的前端, 用在別的語言也可以, 所以就用了 Julia-Python-R 三種語言, 命名為 Jupyter。

其實我很討厭網頁界面

而且還知道討厭的原因是 Sage

剛開始其實也不太喜歡 IPython

主要原因之一是（之前）有點難裝

後來發現真的很炫!

與 IPython, PyCUDA 作者認真 (?) 交流

Jupyter 互動模式

老手要注意有點變動

一切的開始

其實 interact, interact_manual 只要其中一個就好。

```
from ipywidgets import interact, interact_manual
```

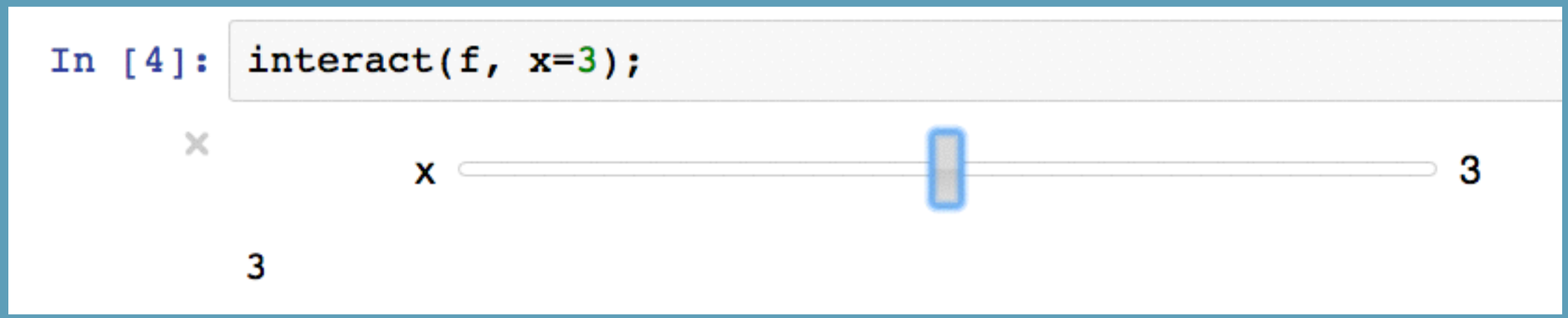
隨便來個函數

我就不信你想得出更簡單的。

```
def f(x):
    print(x)
```

來個數值滑桿

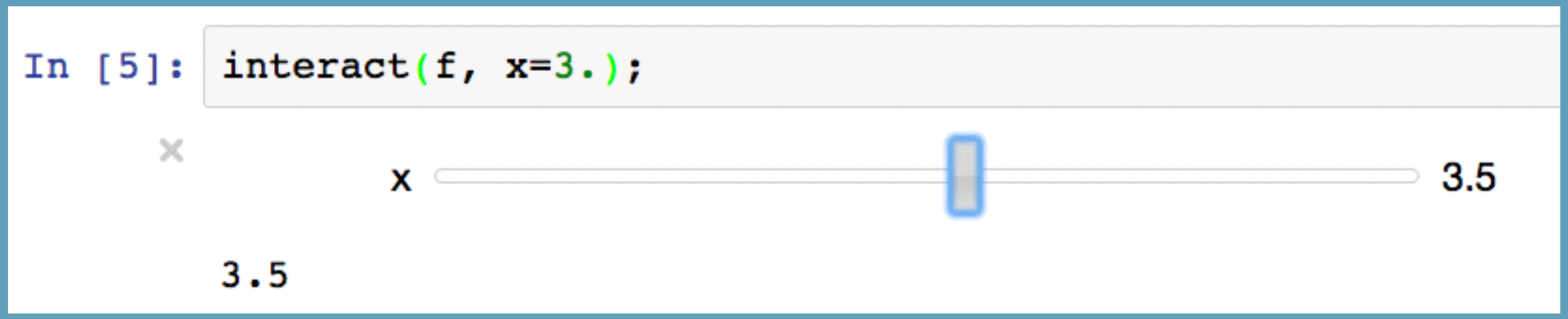
簡單到有點過份。




```
interact(f, x=3);
```

來個數值滑桿

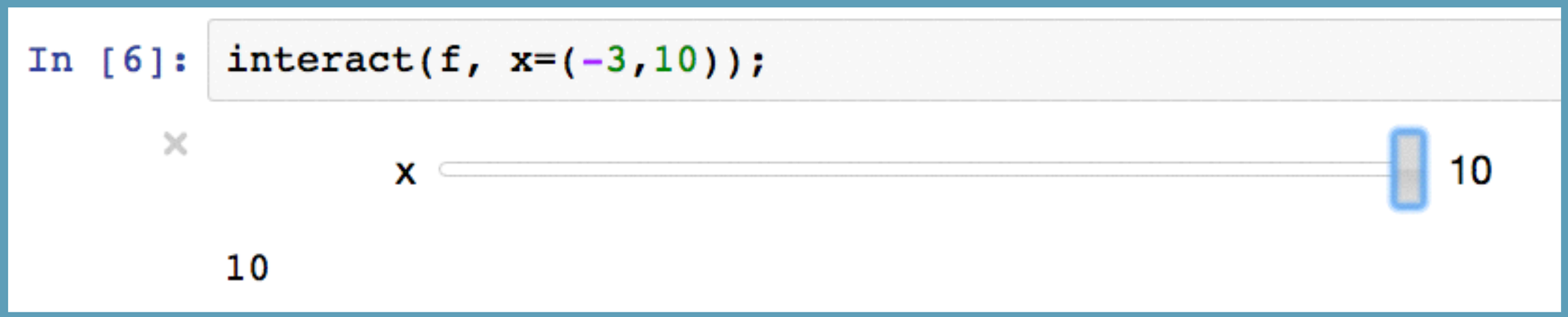
要浮點數版的。




```
interact(f, x=3.);
```

來個數值滑桿

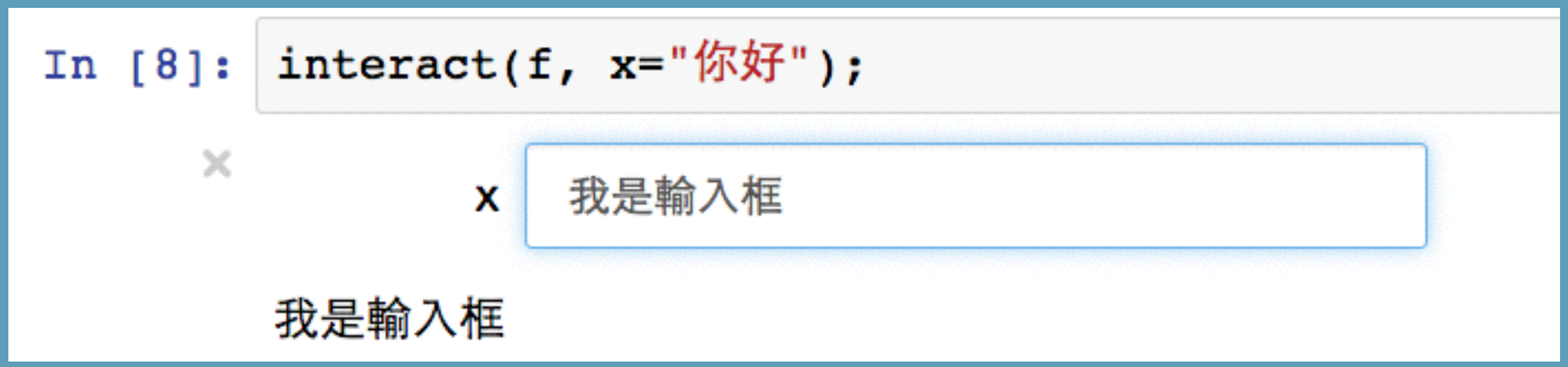
設定範圍也可以。




```
interact(f, x=(-3,10));
```

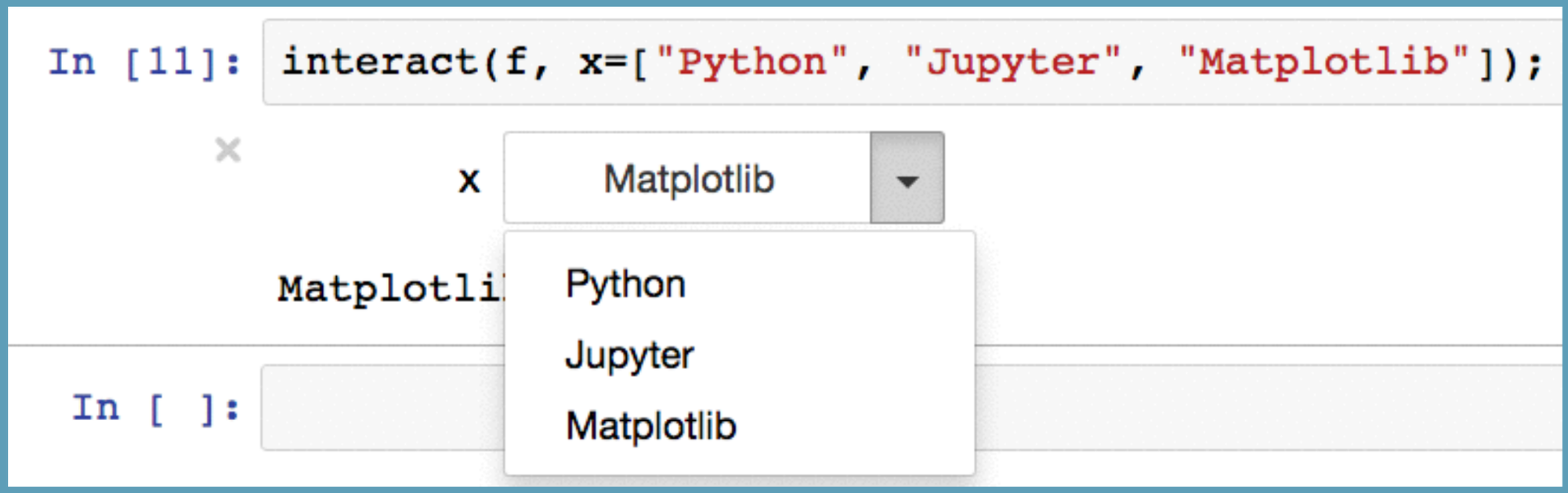
文字輸入框

相信你看出了端倪。



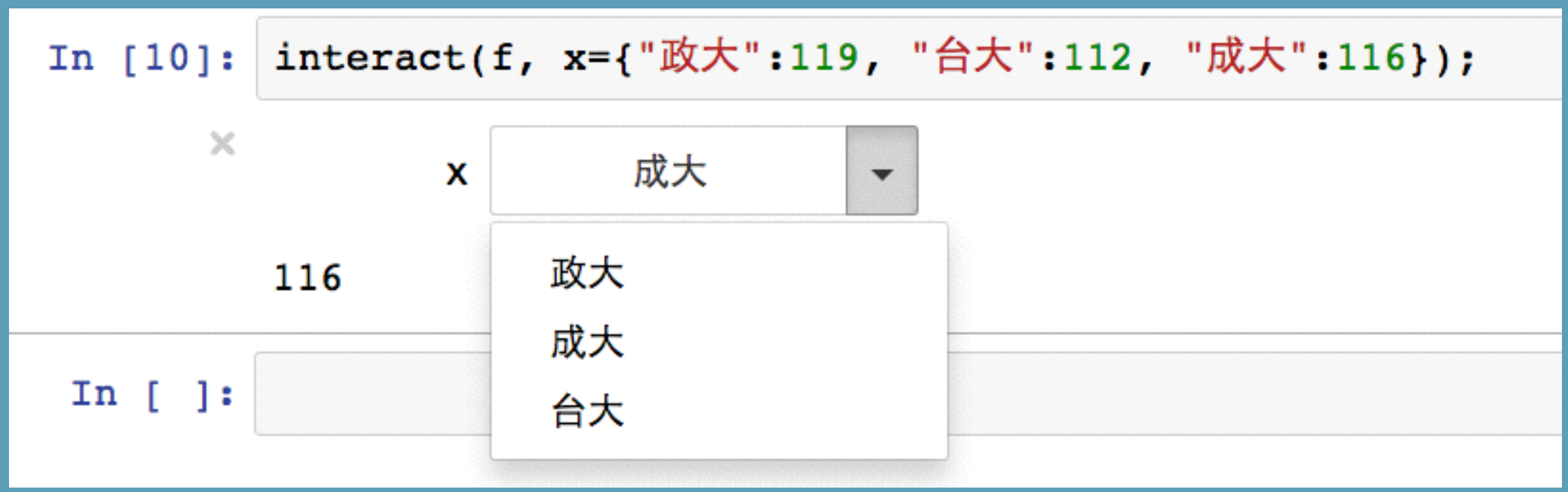

```
interact(f, x="你好");
```

下拉式選單!




```
interact(f, x=["Python", "Jupyter", "Matplotlib"]);
```

下拉式選單!




```
interact(f, x={"政大":119, "台大":112, "成大":116});
```

總之, 就是每個資料結構就對應一種互動模式!

interact_manual?

原本互動是「即時」的，有時我們希望調好、按個鈕再做，就要用到這（基本用法一樣）。

Python 3 很懂中文

函數、變數都可以用中文

匯率換算 GUI 程式!

想想怎麼做出這個 (用 `interact_manual`)?

動態圖形

圖形也可以互動，只是...

[例子] 找函數過指定點

在平面上給幾個任意點, 找一個函數儘可能通過最多點。

準備畫圖的標準動作

這才是「專業」在 Jupyter 下用 matplotlib 的方法。

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

隨便找 7 個點

我們隨便找七個點, 座標放入 x0, y0 當中。x 是為畫函數準備的, 做了 1000 個點。

```
x0 = np.linspace(0, 3.14, 7)
y0 = np.random.rand(7)
x = np.linspace(-0.5, 3.5, 1000)
```

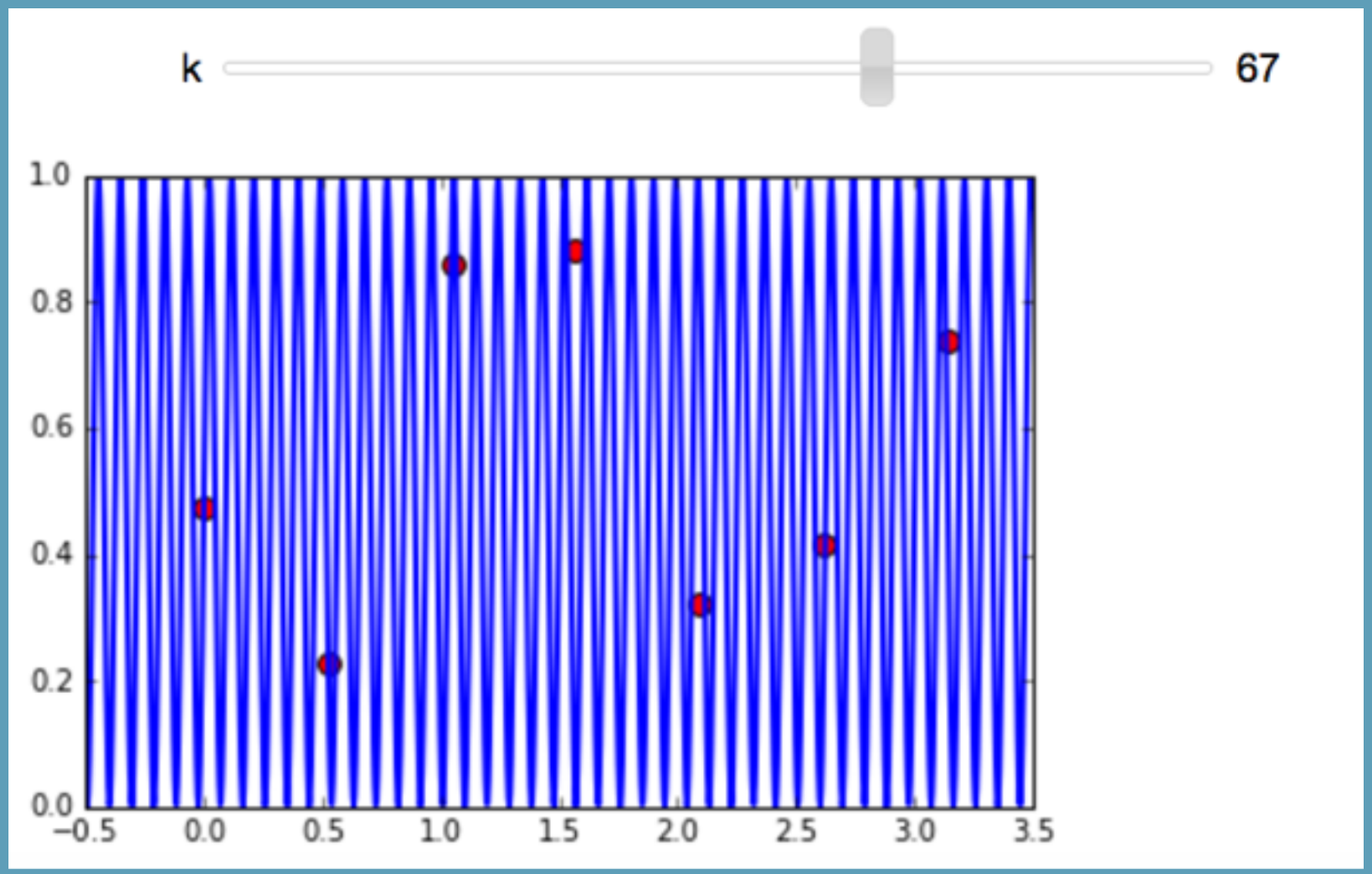
我們奸詐的使用...

$$f(x) = 0.5 \sin(k \cdot x) + 0.5$$

Fitting!

```
def fitting(k):
    plt.xlim(-0.5, 3.5)
    plt.ylim(0, 1)
    plt.scatter(x0,y0,s=60,c='r')
    plt.plot(x, 0.5*np.sin(k*x)+0.5,lw=2)
```

Jupyter 動態表演




```
interact(fitting, k=(1,100));
```

改用 Matplotlib 動畫

因為 Jupyter 做有點頓...

matplotlib 動畫三部曲

1. 畫個背景
2. 清空函數
3. 動畫函數

然後我們需要設定 matplotlib 用 HTML 5 輸出，就能在 Jupyter 中顯示！

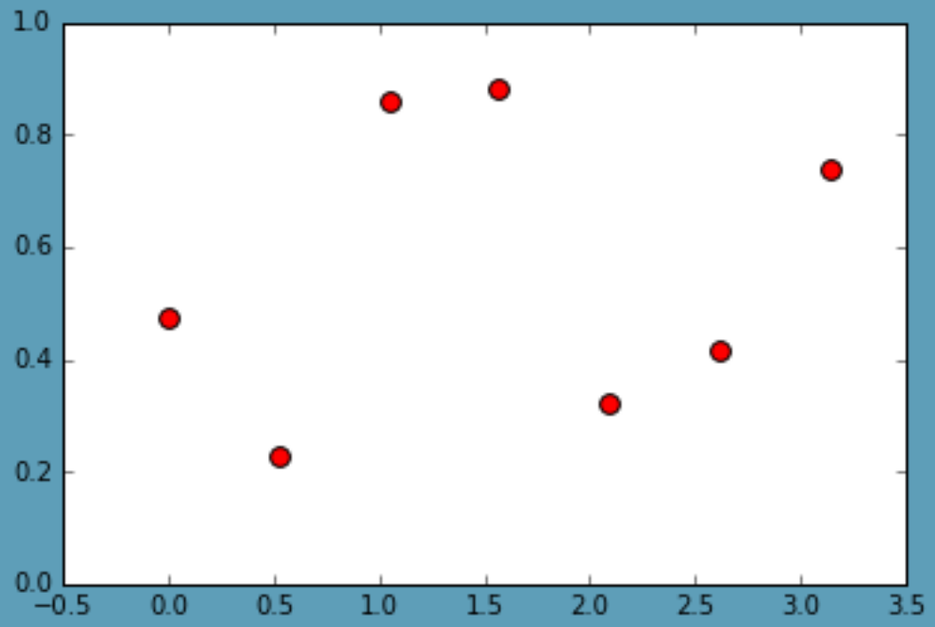
準備工作

使用 matplotlib 的 animation 套件, 還有準備改一點點參數。

```
from matplotlib import animation, rc
rc('animation', html='html5')
```

Step 1. 畫個背景

基本上就是畫不會動的部份。




```
fig, ax = plt.subplots()
ax.set_xlim((-0.5,3.5))
ax.set_ylim((0, 1))
ax.scatter(x0,y0,s=60,c='r')
line, = ax.plot([], [], lw=2)
```

Step 2. 清空函數

寫個 init 函數, 每次要畫新的圖清除舊的圖用的。

```
def init():
    line.set_data([], [])
    return (line,)
```

Step 3. 動畫函數

其實就是寫個帶參數、每個參數長不一樣的函數。

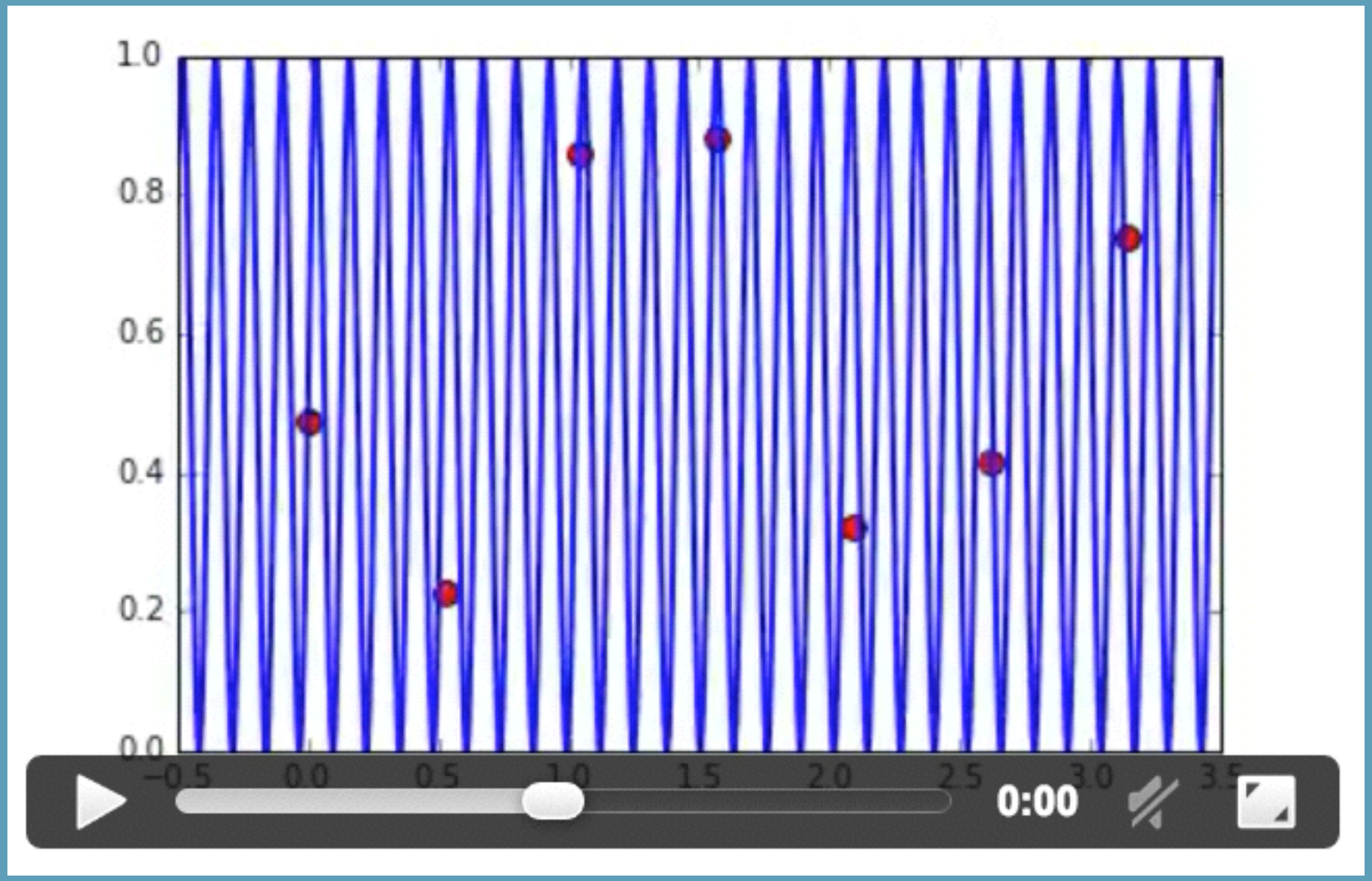
```
def animate(k):
    y = 0.5*np.sin(k*x)+0.5
    line.set_data(x, y)
    return (line,)
```

動起來!

```
anim = animation.FuncAnimation(fig, animate, init_func=init,
                               frames=100, interval=20,
                               blit=True)
```

frame	要幾張 frame。
interval	多久換一張, 以毫秒為單位, 所以這意思是 0.02 秒畫一張。
blit	這裡設 `True` 意思是不動的不用重畫。

然後真的可以動了！




```
anim
```

Story 3 • **Neural Networks**

暗黑學習法

真的有學任意函數的技法

[例子] 股票點數預測

f(日期 x) = 某股票 x 當天的收盤價

這可能不是最好的描述法。也許我們比較想要用之前的情況預測我們的想知道的那天。也就類似是下面的函數。

$$f(x_1, x_2, \ldots, x_{n-1}) = x_n$$

[例子] 你是那種類型的人？

假設我們把人分成若干型，例如 9 型 (九型人格)，16 型 (Myers-Briggs) 等等。於是我們想看這是哪一型的人，就是找這個函數：

f(某人 x) = x 這個人的人格型式

你怎麼「輸入」一個人呢？ 在此例通常是經過心理測驗，得到一串數字，那些數字我們就可以代表這個人。

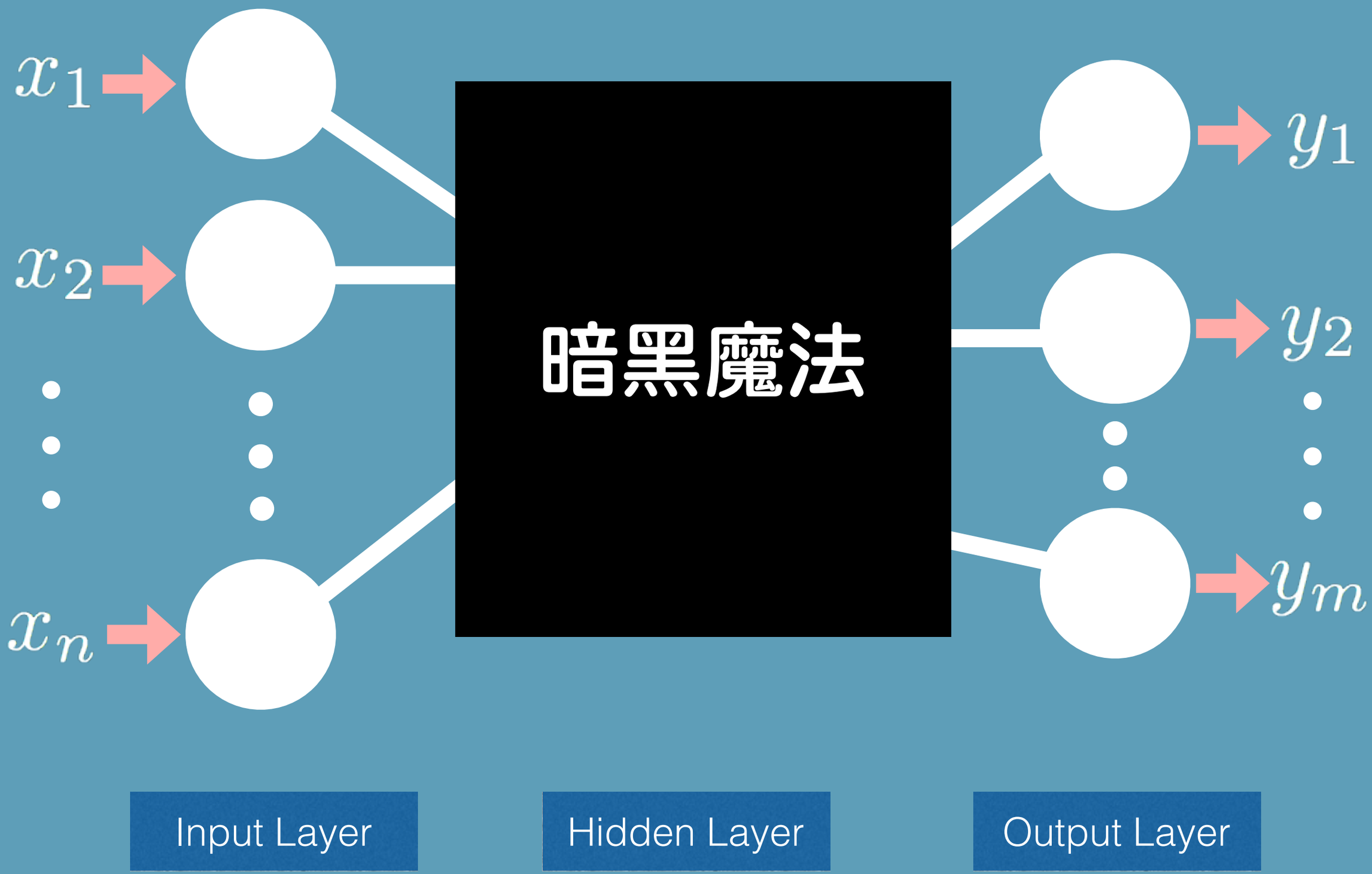
[例子] 速配指數

f(x, y) = x, y 這兩個人的速配指數

我們再度碰到要把人變成一串描述他的數字。很多可以用，比如星座、血型、興趣、學歷等等我們都可想辦法轉數字。

總之我們就是要函數

$$f : \mathbb{R}^n \to \mathbb{R}^m$$


x_1
x_2
x_n
暗黑魔法
y_1
y_2
y_m
Input Layer
Hidden Layer
Output Layer

厲害的是神經網路什麼都學得會!

而且你完全不用告訴它函數應該長什麼樣子: 線性啦、二次多項式啦等等。

神經網路是 Universal Approximator!

設 $\varphi: X \to Y$ 是一個有界、非常數連續函數, 其中 $X \subset \mathbb{R}^n$, 且 $Y \subset \mathbb{R}^m$ (φ 將是我們的 activation fuction)。設 I_n 是一個 n 維超立方體, 且 $\mathcal{C}(I_n)$ 為所有定義在 I_n 上的連續函數。則對於任意的 $f \in \mathcal{C}(I_n)$, 與任意的 $\varepsilon > 0$, 都存在 $N > 0$, 及

$$\alpha_i, \mathbf{w} = \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix}, b_i, i = 1, 2, \ldots, N$$

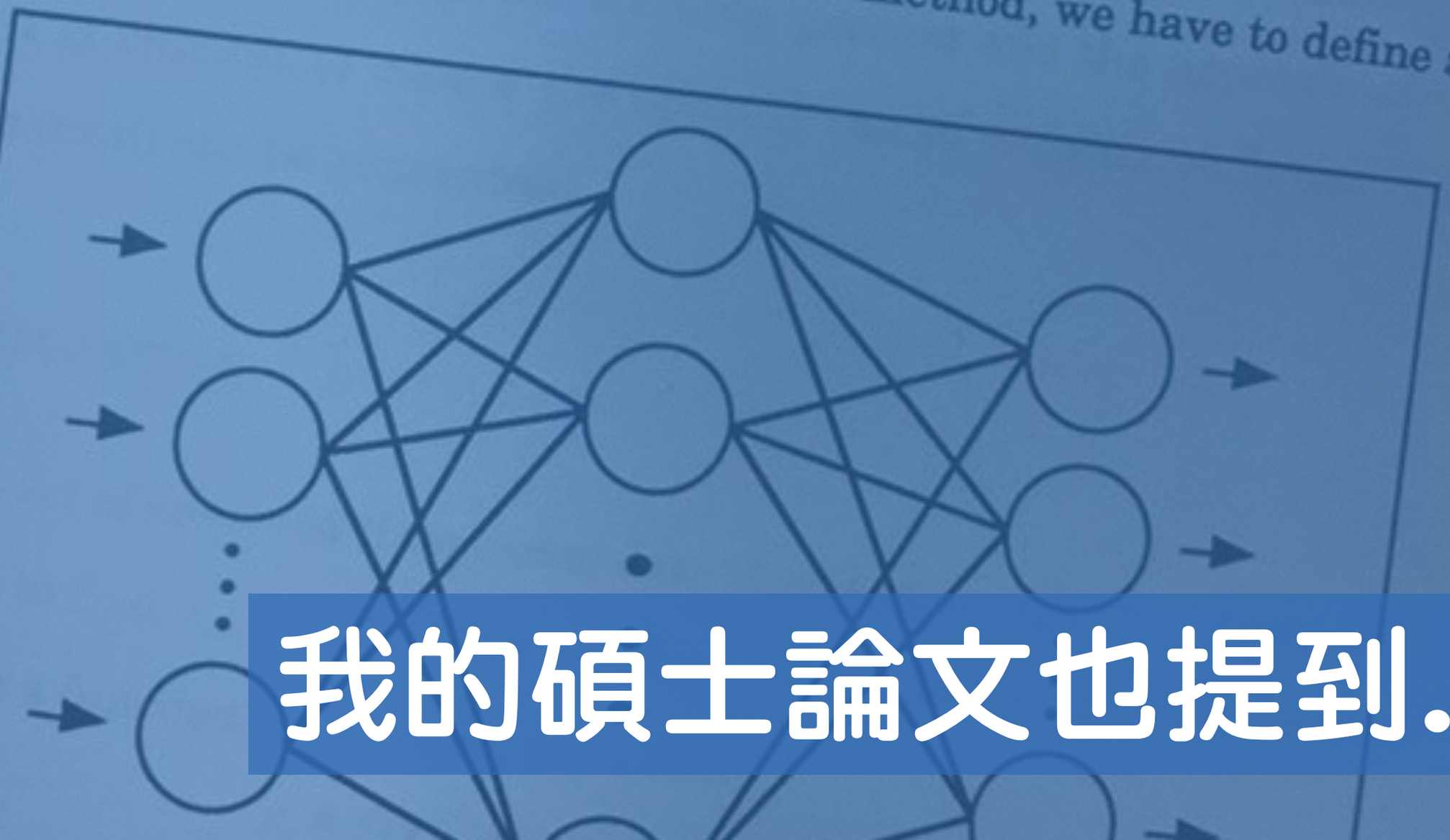
使得

$$F(\mathbf{x}) = \sum_{i \leq N} \alpha_i \varphi(\mathbf{w}^t \mathbf{x} + b_i) \quad \sup_{\mathbf{x} \in I_n} |f(x) - F(x)| < \varepsilon$$

所以在 1980-1990 變成很潮的東西

2.3 Back Propagation Networks

The back propagation network is a kind of the feedforward networks. The network uses "gradient descent" (Rumelhart and McClelland, 1986) method as the learning algorithm. Using this method, we have to define a



我的碩士論文也提到...

然後突然大家沒興趣了

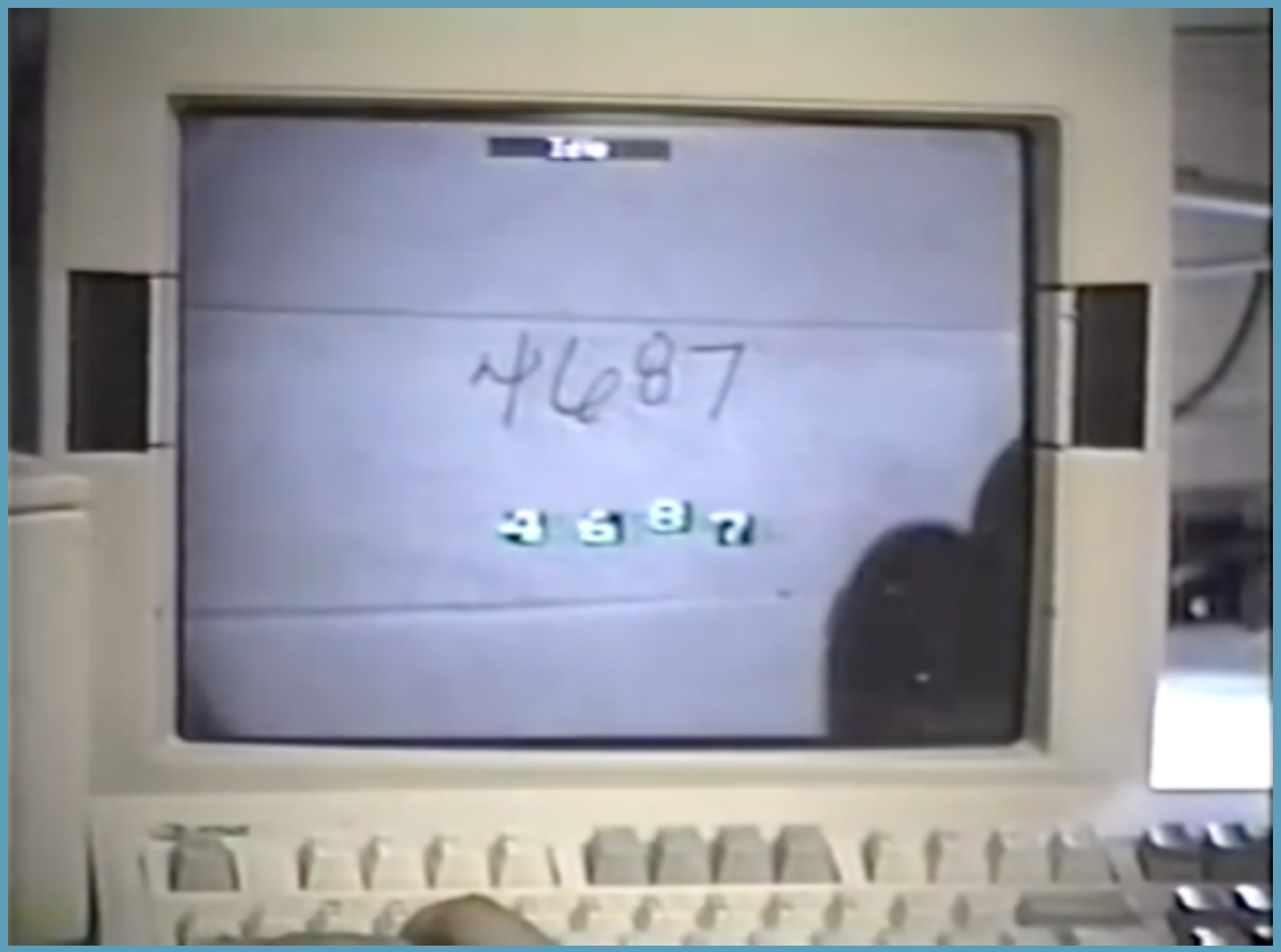
差不多近 2000 年

Deep Learning

神經網路再現!

關鍵

CNN / RNN / Deep



看到這 1993 第一代 CNN 都快哭了

https://youtu.be/FwFduRA_L6Q

李宏毅老師很棒的介紹

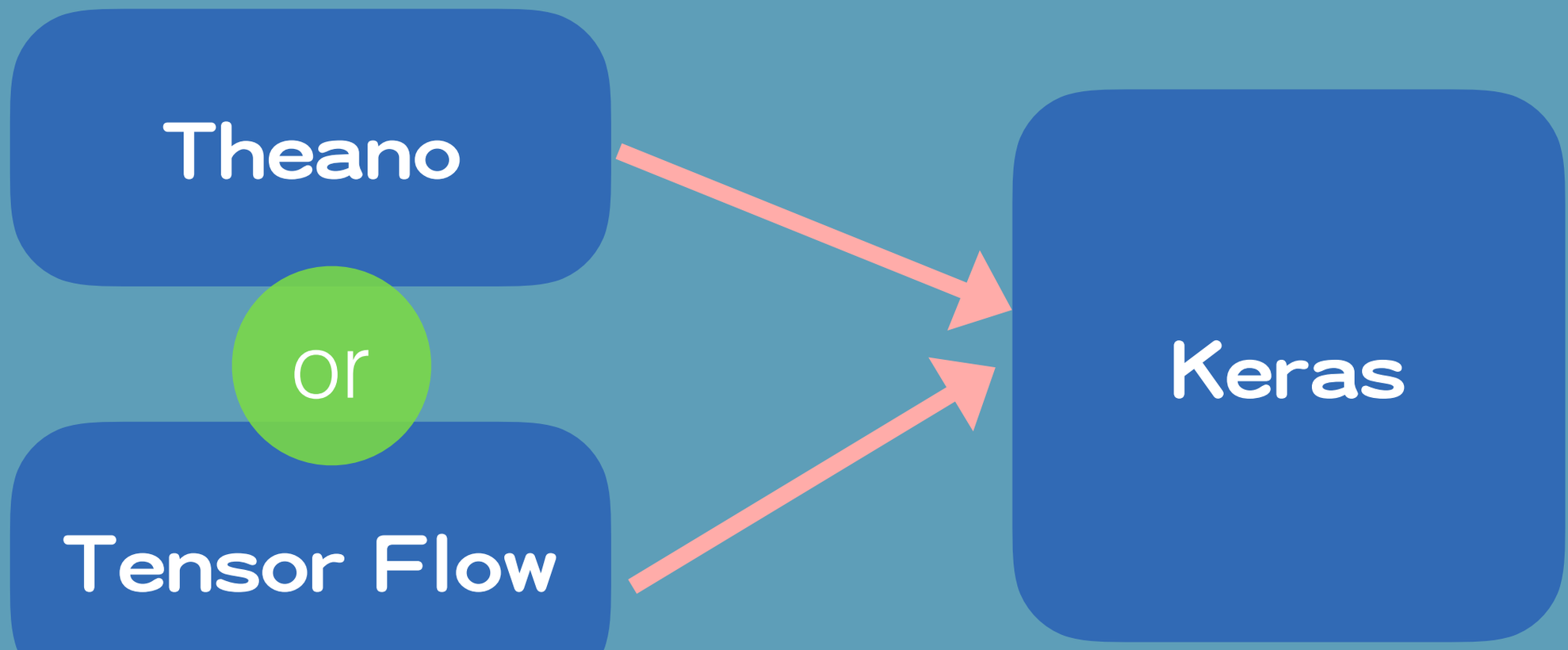
一天搞懂深度學習

http://goo.gl/appfNa

用 Python 做很簡單

現在有個 Keras

Keras 瞬間完成神經網路



Keras 很方便、快速完成神經網路。

在 Mac 裝很容易

http://goo.gl/Z1RS2K

我就說我沒要認真介紹

麻煩看李宏毅老師的投影片

這裡只秀給你看很簡單

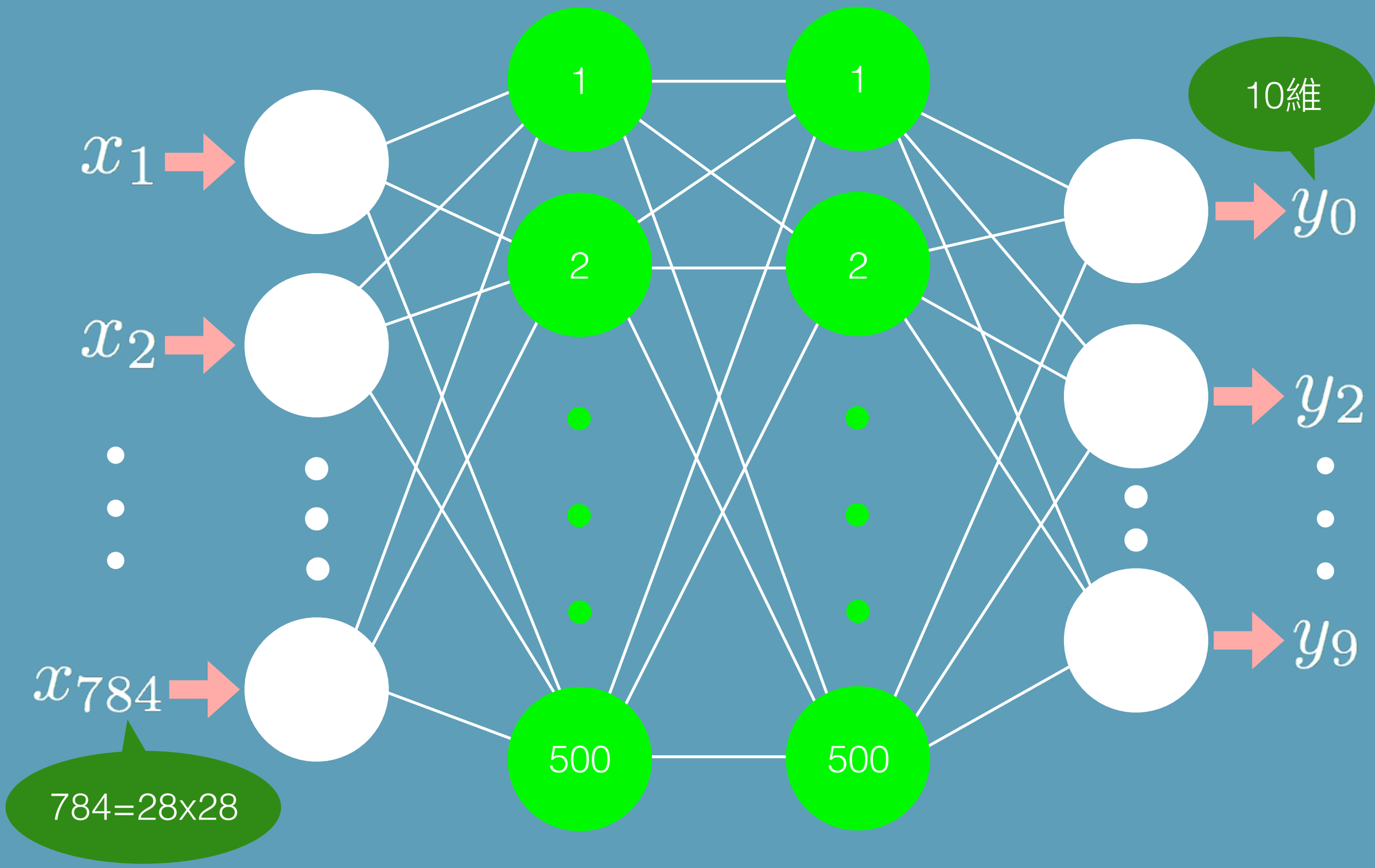
如果你有寫過神經網路的程式會哭

例如我們要做手寫辨識

雖然現在大家都用 CNN, 但我們借李宏毅
老師第一個例子用最傳統的

架構是這樣

- 輸入層有 28x28 = 784 個節點
- 兩個隱藏層, 各有 500 個節點
- 10 個輸出節點


x_1
x_2
x_{784}
784=28x28
1
2
500
1
2
500
10維
y_0
y_2
y_9
Input Layer
Hidden Layer
Output Layer

開始

和很多機器學習一樣, 弄個機器出來, 「正常」的 feedforward 神經網路就用 Sequential。

```
model = Sequential()
```

輸入層+隱藏層1

是不是很簡單, 基本上只要指出有幾個節點, 選哪個 activation function 就好。

```
model.add(Dense(input_dim=28*28, output_dim=500))
model.add(Activation('sigmoid'))
```

隱藏層2

輸入就是 500 個不用說, 直接說要輸出多少個, 也就是多少個節點。

```
model.add(Dense(output_dim=500))
model.add(Activation('sigmoid'))
```

輸出層

10 個輸出, 分別代表辨識為 0, 1, 2, ⋯, 9 的機率。

```
model.add(Dense(output_dim=10))
model.add(Activation('softmax'))
```

然後就組裝

說一下要用什麼當 loss function, 還有學習法。

```
model.compile(loss='mse', optimizer=SGD(lr=0.1),
metrics=['accuracy'])
```

基本上就好了

當然這只有架構部份，詳請麻煩看 code

檢測學習成果

又可以用 Jupyter 的互動功能, 快速做出一個 GUI 界面!