Perceptron與多層感知器(Multi-Layer Perceptrons)

林郁修

Yu-Hsiu Lin
yhlin@ntut.edu.tw
https://sites.google.com/view/aici

Graduate Institute of Automation Technology,
National Taipei University of Technology

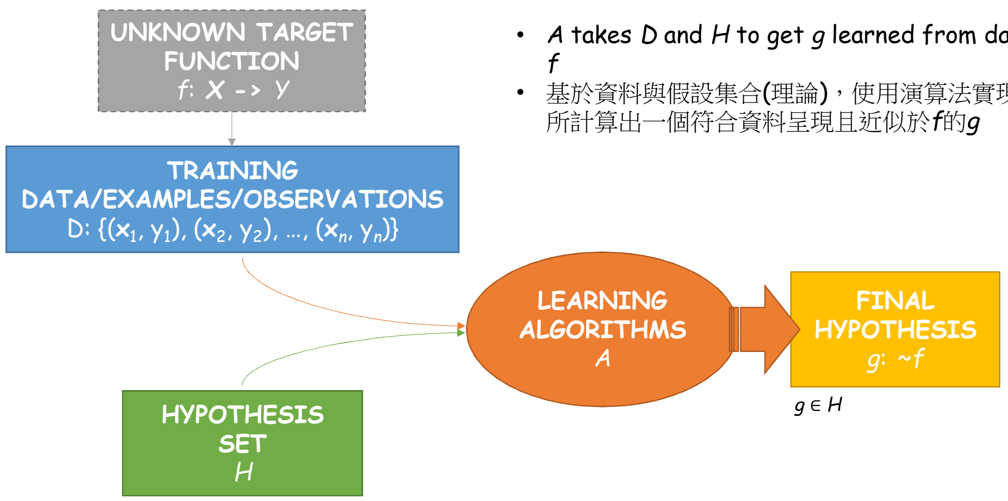
Outline

- 機器學習的架構
- 感知器 (Perceptron)
- 感知器的學習
- 感知器 VS. 多層感知器 (Multi-Layer Perceptrons)
- 多層感知器的學習

Outline

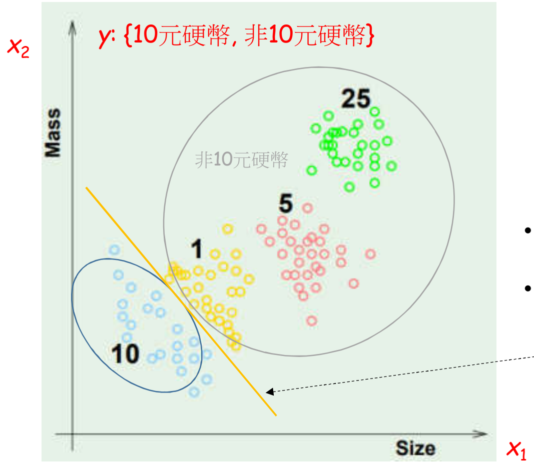
- 機器學習的架構
- 感知器 (Perceptron)
- 感知器的學習
- 感知器 VS. 多層感知器 (Multi-Layer Perceptrons)
- 多層感知器的學習

機器學習的架構



- *A* takes D and *H* to get *g* learned from data for *f*
- 基於資料與假設集合(理論)，使用演算法實現假設所計算出一個符合資料呈現且近似於*f*的*g*

- coin recognition



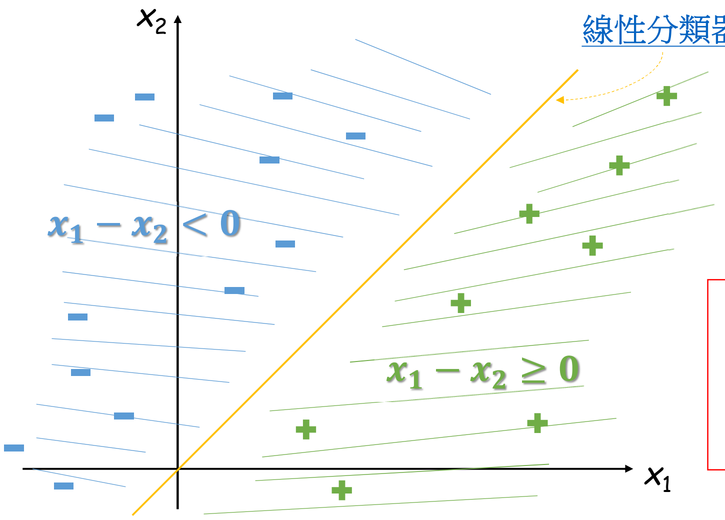
- 資料集*D*（每一筆資料由2維特徵/屬性所組成）作為(訓練)資料
- 自**機器學習理論*H*實踐演算法*A***找出*g*

the perceptron that can also be used to Approve/Deny credits … for **Credit Analysis**

Outline

- 機器學習的架構
- 感知器 (Perceptron)
- 感知器的學習
- 感知器 VS. 多層感知器 (Multi-Layer Perceptrons)
- 多層感知器的學習

感知器 (Perceptron)

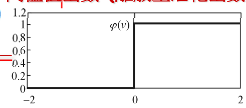


$$y = \begin{cases} \text{"類別 +"}, & if\ x_1 - x_2 \geq 0 \\ \text{"類別 −"}, & if\ x_1 - x_2 < 0 \end{cases}$$

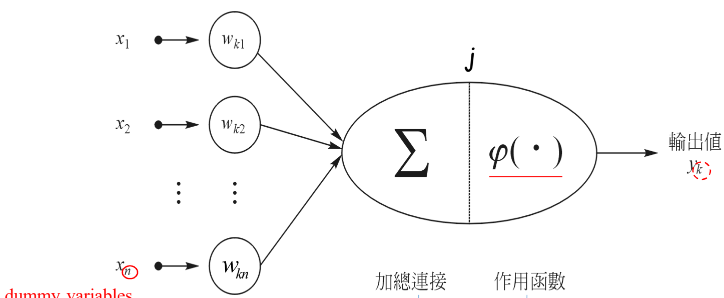
$$net_j = \sum_{i=1}^{m} w_{ji} x_i + b_j$$

$$y_j = F(net_j)$$

門檻值函數 (離散型活化函數)

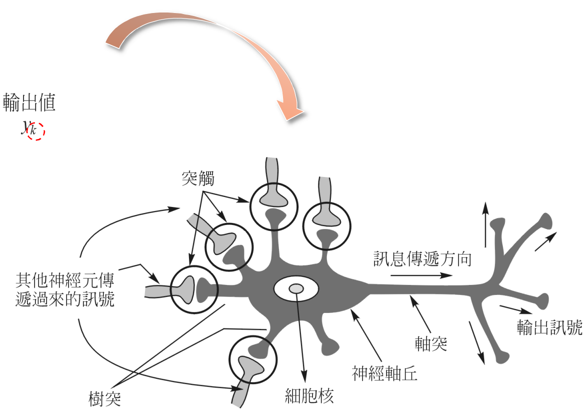


- 感知器/類神經元(neuron)數學模型 -- Rosenblatt（1958）



$$net_j = \sum_{i=1}^{m} w_{ji} x_i + b_j$$

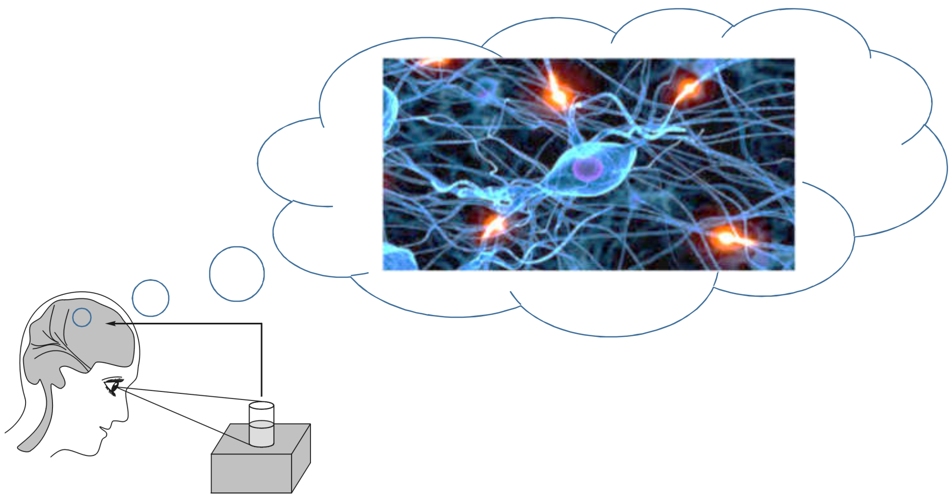
$$y_j = F(net_j)$$



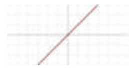
- 生物大腦中約**有數十億個到數佰億個神經元**，有的神經元間沒有相互連結，有的則以非常錯綜複雜的方式相連結，甚至有環狀或有回饋(**feedback**)式的連結

- **生物對新事物的學習(learning)，基本上是造成神經元間連結(weights)強弱的改變(*weighting*)**，或是使原本沒有連結的神經元間產生新的連結
  - 連結權重w_{ji}即模擬不同生物神經元之間的連結強弱
  - 連結權重值可為**正**亦可為*負*；分別代表**刺激**或是*抑制*反應

- 神經元太少無法處理較複雜的問題
  - 深度學習的發展



- Activations functions
  - Identity



  - TanH

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

  - Rectified Linear Unit (ReLU)

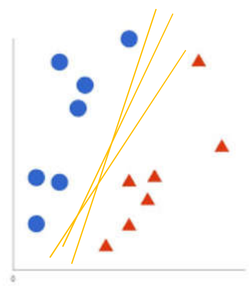
  - …

Outline

- 機器學習的架構
- 感知器 (Perceptron)
- 感知器的學習
- 感知器 VS. 多層感知器 (Multi-Layer Perceptrons)
- 多層感知器的學習

感知器的學習 (Perceptron Learning Algorithm)

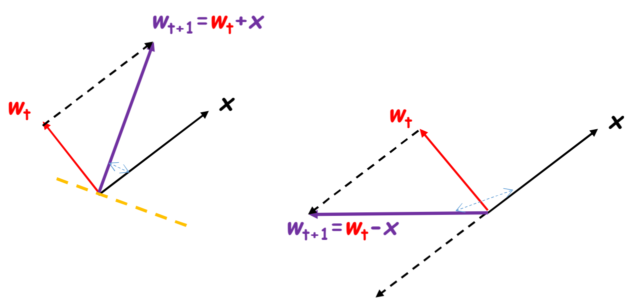
- 為何需要“學習”？
  - 平面中可以有無限多條的決策線 (超平面(hyper-plane)/決策邊界(decision boundaries))…
  - 高維度的特徵空間 (feature space)
  - …



1. 隨機決定一條決策線
2. 分類錯誤
3. “修正”參數

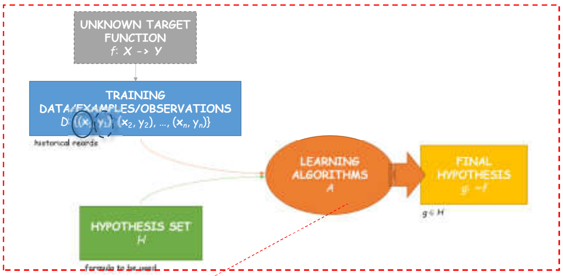
- PLA
  - 決策超平面: 兩個多維度的向量做內積 $\boldsymbol{w}^{\mathsf{T}}\boldsymbol{x}$
  - 向量加、減法

$$y = \begin{cases} \text{“類別” } +1\text{”}, & if\ \ x_1 - x_2 \geq 0 \\ \text{“類別” } -1\text{”}, & if\ \ x_1 - x_2 < 0 \end{cases}$$



$$\Delta \boldsymbol{w}_j = \eta\left[d - sgn\left(\boldsymbol{w}_j^{\mathsf{T}}\boldsymbol{x}\right)\right]\boldsymbol{x}$$

$$\Delta \boldsymbol{w}_j = \begin{cases} \pm 2\eta\boldsymbol{x}, & \text{目標值與類神經元輸出值不一致} \\ 0\ , & \text{目標值與類神經元輸出值一致} \end{cases}$$



$$\Delta \mathbf{w}_j = \eta \left[d - sgn\left(\mathbf{w}_j^T \mathbf{x} \right) \right] \mathbf{x}$$

$$\Delta \mathbf{w}_j = \begin{cases} \pm 2\eta \mathbf{x}, & \text{目標值與類神經元輸出值不一致} \\ 0\ , & \text{目標值與類神經元輸出值一致} \end{cases}$$

PLA pseudo-code (參考)

- If a mis-classification exists, a perceptron that aims to classify all patterns correctly is trained.

initialize weight vector $\vec{w}$ and bias weight w_0 arbitrarily
while exist misclassified pattern $\vec{x} \in \mathcal{P} \cup \mathcal{N}$ **do**
  if $\vec{x} \in \mathcal{P}$ **then**
    $\vec{w} \leftarrow \vec{w} + \vec{x}$
    $w_0 \leftarrow w_0 + 1$
  else
    $\vec{w} \leftarrow \vec{w} - \vec{x}$
    $w_0 \leftarrow w_0 - 1$
  end if
end while
return $\vec{w}$ and w_0

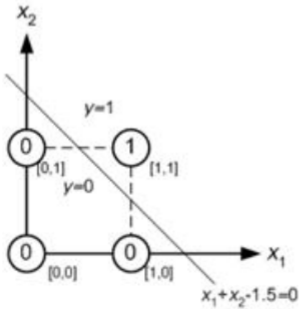
P: positive training patterns

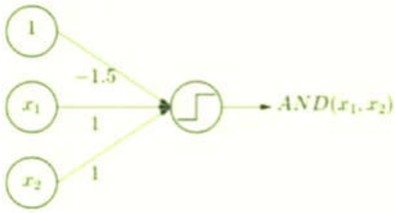
N: negative training patterns

Assignment #1 (Part I)

- 解決**AND gate分類**問題
- 利用**Perceptron**學習法**(PLA)**更新感知器的權重值

x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

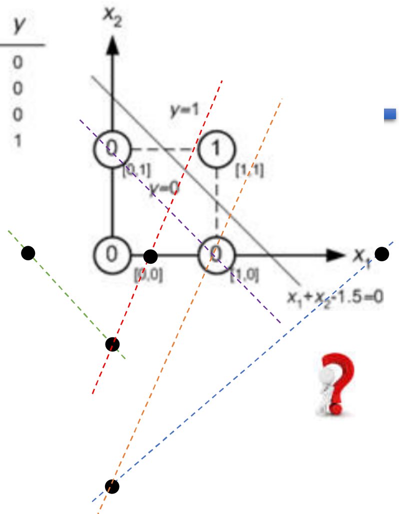




x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1

epoch

- $\boldsymbol{w}_{t=0}$ = (0.5, 0.5, 0.5) ← $0.5x_1+0.5x_2+0.5$ (the initial decision boundary)
- $\boldsymbol{w}_1=\boldsymbol{w}_0-\boldsymbol{x}$ = (0.5, 0.5, 0.5)-(0, 0, 1) = (0.5, 0.5, -0.5)
  $0.5x_1+0.5x_2-0.5$
- $\boldsymbol{w}_2=\boldsymbol{w}_1-\boldsymbol{x}$ = (0.5, 0.5, -0.5)-(0, 1, 1) = (0.5, -0.5, -1.5)
  $0.5x_1-0.5x_2-1.5$
- $\boldsymbol{w}_3=\boldsymbol{w}_2$
- $\boldsymbol{w}_4=\boldsymbol{w}_3+\boldsymbol{x}$ = (0.5, -0.5, -1.5)+(1, 1, 1) = (1.5, 0.5, -0.5)
  $1.5x_1+0.5x_2-0.5$
- $\boldsymbol{w}_5=\boldsymbol{w}_4$
- $\boldsymbol{w}_6=\boldsymbol{w}_5-\boldsymbol{x}$ = (1.5, 0.5, -0.5)-(0, 1, 1) = (1.5, -0.5, -1.5)
  $1.5x_1-0.5x_2-1.5$
- …



$$\Delta \boldsymbol{w}_j = \eta\left[d - sgn\left(\boldsymbol{w}_j^T \boldsymbol{x}\right)\right]\boldsymbol{x}$$

$$\Delta \boldsymbol{w}_j = \begin{cases} \pm 2\eta \boldsymbol{x}, & \text{目標值與類神經元輸出值不一致} \\ 0\ , & \text{目標值與類神經元輸出值一致} \end{cases}$$

- the code

```python
import numpy as np

inputs = []
inputs.append(np.array([1, 1, 1]))
inputs.append(np.array([1, 0, 1]))
inputs.append(np.array([0, 1, 1]))
inputs.append(np.array([0, 0, 1]))

labels = np.array([1, 0, 0, 0])

Iters = 10

no_of_inputs = 2
weights = np.random.randn(no_of_inputs + 1)
print("initial: " + str(weights))

learning_rate = 0.15
```

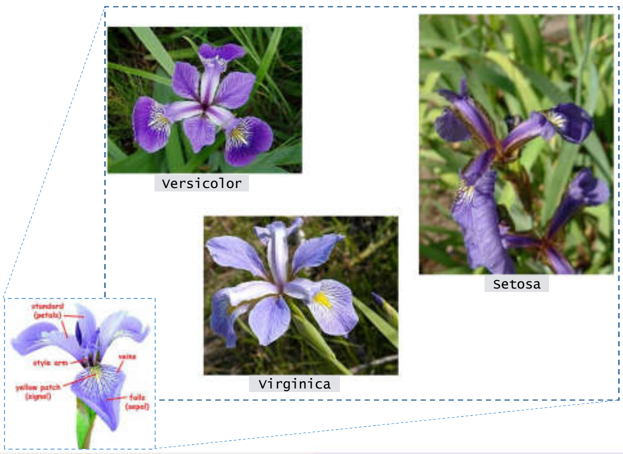


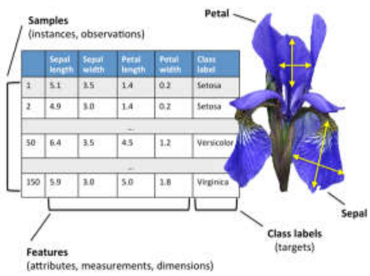
```python
# cont'd
for _ in range(Iters):
    for _input, label in zip(inputs, labels):

        summation = np.dot(_input, weights)  # dot product

        if summation > 0:  # the step activation function
            predicted = 1
        else:
            predicted = 0

        weights += learning_rate * (label - predicted) * _input

print("trained: " + str(weights))
```

- initial:
  [0.73421793 -0.38731 0.85634143]
- trained:
  [0.13421793 0.06269 -0.19365857]

Assignment #1 (Part II)

- 解決**Iris鳶尾花分類**問題
- 利用**Perceptron學習法(PLA)**更新感知器（階層式）的權重值





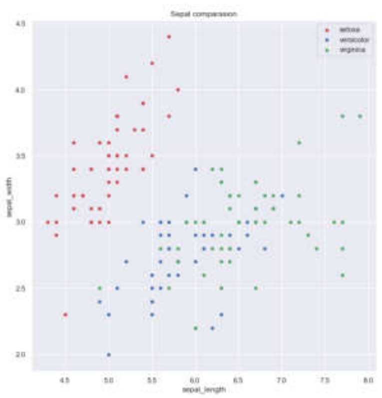
Assignment #1 (Part II) (Cont'd)

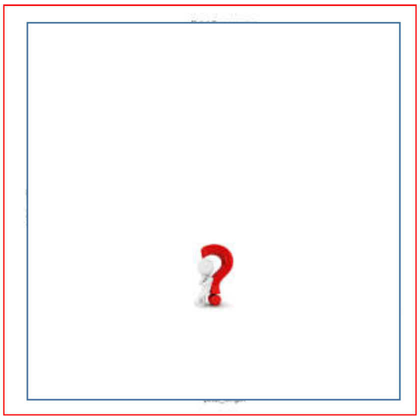
- Hints
  - import pandas as pd
  - iris = pd.read_csv('https://raw.githubusercontent.com/mwaskom/seaborn-data/master/iris.csv')
  - iris.head()

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

Assignment #1 (Part II) (Cont'd)

- Hints (Cont'd)
  - 特徵選取於變量使用：sepal length vs. sepal width, sepal length vs. petal width, or …
  - https://ithelp.ithome.com.tw/articles/10264416



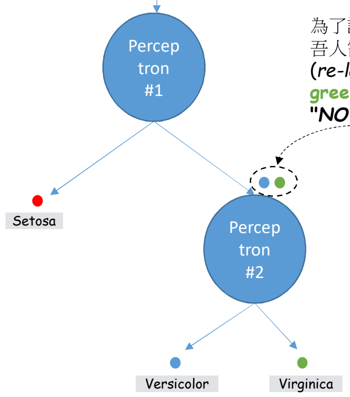


…

• Hints (Cont'd)
 – https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html
 – https://scikit-learn.org/stable/modules/generated/sklearn.utils.resample.html

• 若某一筆資料輸入至Perceptron #1且被分類為"非"Setosa，則該筆資料被接著輸入至Perceptron #2。
• 其中，該筆資料的類別可能為"Versicolor"或"Virginica"。

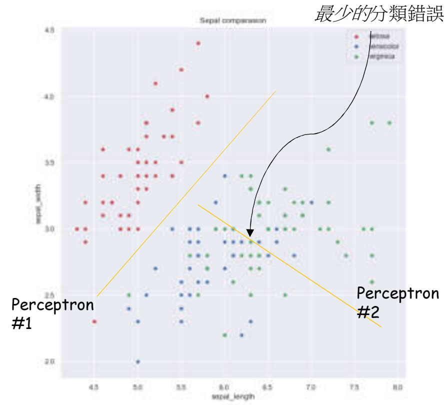
• Hints (Cont'd)
 – 階層式二元分類**Perceptrons**



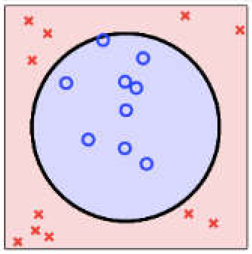
為了訓練Perceptron #1，吾人需要對資料*重新標註*(*re-labeling*)：**blues**跟**greens** 需要被標註為"***NOT* reds**"。

此分類器運作如同一決策樹

於試誤中，取得*最少的*分類錯誤。



- Hints (Cont'd)
- 什麼情況PLA不會停止？
  - 資料分佈為非線性可分離 **(non-linearly separable)**
  - 有限的**PLA**學習迭代次數



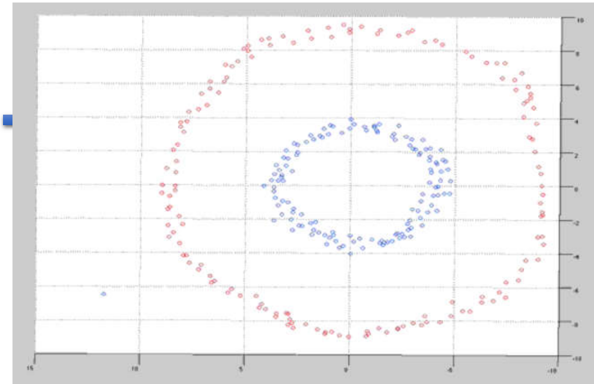
- POCKET ALGORITHM
  - 雜訊資料 (真實世界一定存在著分類錯誤情況 (***nothing is perfect***))
  - 口袋名單 ($g \in H$)

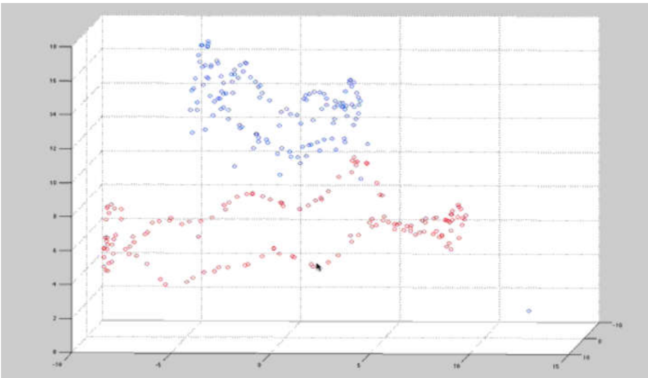
- Hints (Cont'd)
  - https://scikit-learn.org/stable/modules/generated/sklearn.metrics.accuracy_score.html

- 需說明特徵選取(考量哪兩個特徵作為分類器的輸入變量)
- 需視覺化呈現決策邊界

- 佔本課程學期總成績15%

- Due：4/5(三), 2023 23:59前上傳PPT檔至i學園作業繳交區
- Oral: 4/6(四), 2023 約10分鐘/人





Perceptron Convergence Theorem (參考)

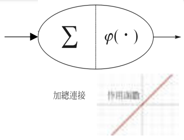
- http://www.cs.columbia.edu/~mcollins/courses/6998-2012/notes/perc.converge.pdf
- http://www.cems.uvm.edu/~rsnapp/teaching/cs295ml/notes/perceptron.pdf

(參考)

學習演算法	Δw_{ji}	初始權重	學習法	神經元特性
Hebbian	$\eta y_j x_i(t)$	0	非監督式	任意
Perceptron	$\eta\ [d_j - sgn(w_j x)]x_i$	任意	監督式	**binary/bi-polar**
Delta	$\eta\ (d_j - y_j)f'(net_j)x$	任意	監督式	連續
Widrow-Hoff	$\eta\ (d_j - w_j x)x$	任意	監督式	任意
Correlation	$\eta\ d_j x$	0	監督式	任意
Winner-takes-all	$\alpha\ (x_i - w_{mi})$	任意（正規化）	非監督式	連續
Grossberg	$\beta\ (d_m - w_{mi})$	0	監督式	連續

連續型活化函數

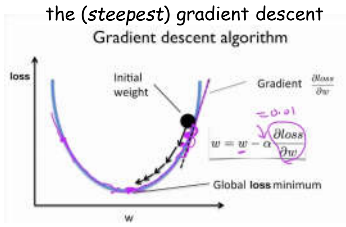
- Perceptron: $y=f(x)=w \cdot x$
- Activation function: Identity



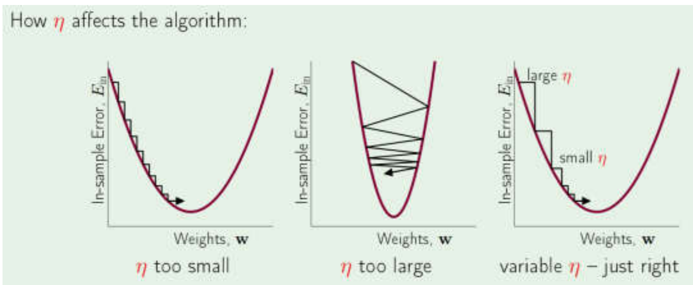
- Error function/cost function:

$$E(w)=\frac{1}{2}(\underbrace{d}_{期望}-\underbrace{y}_{預測})^2$$

- Goal: Learn the output towards the value of 50
  - $x=5$
- Derivatives: $[(d-w \cdot x)(-x)]$
- Learning: $\Delta w=-\eta[(d-w \cdot x)(-x)]$



- η: 0.01 ($\in$ (0, 1))
- Results ($\Delta w=+\eta[(d-w \cdot x)(x)]$):
  - 7 * 5 = 35
  - 8.5 * 5 = 42.5
  - 9.25 * 5 = 46.25
  - 9.625 * 5 = 48.125
  - 9.8125 * 5 = 49.0625
  - 9.90625 * 5 = 49.5312
  - …
  - 9.98828 * 5 = 49.9414
  - 9.99414 * 5 = 49.9707

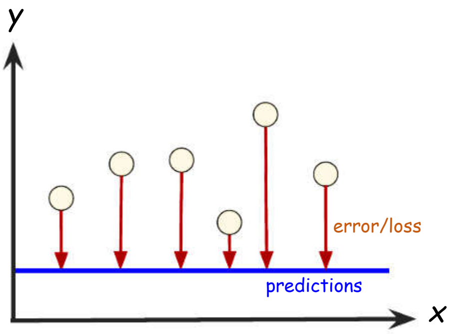


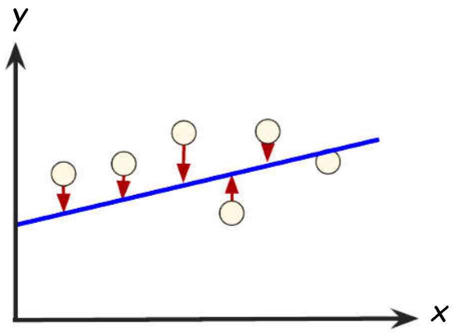
$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$

期望

預測

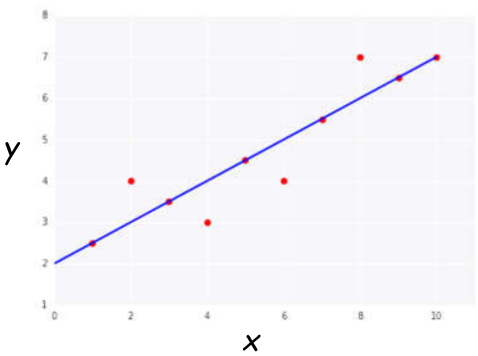
(Mean Squared Error)



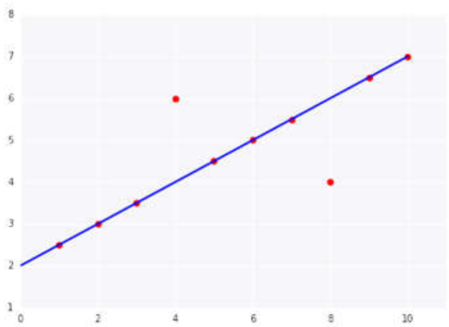


L2損失函數/最小化平方誤差
(Least Square Error)

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2$$



?



?

- import numpy as np

- inputs = [] # x
- inputs.append(np.array([0.922]))
- inputs.append(np.array([0.459]))
- inputs.append(np.array([0.984]))
- inputs.append(np.array([0.794]))
- inputs.append(np.array([0.119]))
- inputs.append(np.array([0.258]))
- inputs.append(np.array([0.734]))
- inputs.append(np.array([0.123]))
- inputs.append(np.array([0.713]))
- inputs.append(np.array([0.943]))

- labels = np.array([0.559, 0.298, 0.639, 0.516, 0.077, 0.167, 0.477, 0.079, 0.463, 0.612]) # y

- Iters = 10

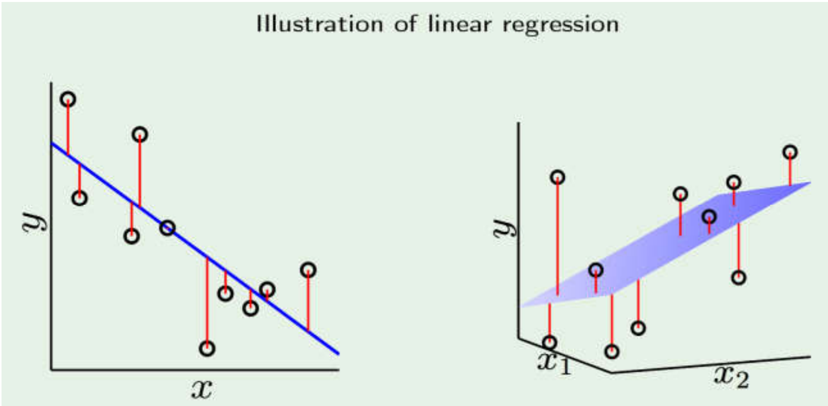
- no_of_inputs = 1
- np.random.seed(55)
- weights = np.random.randn(no_of_inputs)
- print("initial: " + str(weights))

- learning_rate = 0.1

- Err = []
- #_W = []
- for _ in range(Iters):
-     err = 0
-     W = []
-     for _input, label in zip(inputs, labels):
-         predicted = np.dot(_input, weights) # dot product
-         weights -= learning_rate * (label - predicted) * (-1) * _input
-         err += (label - predicted) ** 2
-         #W.append(weights)
-     Err.append(err/len(inputs))
-     #_W.append(np.std(W))

- print("trained: " + str(weights))

- import matplotlib.pyplot as plt
- %pylab inline
- plt.plot(range(0, len(Err)),Err)
- plt.xlabel('Iteration')
- plt.ylabel('loss/error')



Hyper-plane!?