機器學習概論

劉昭麟

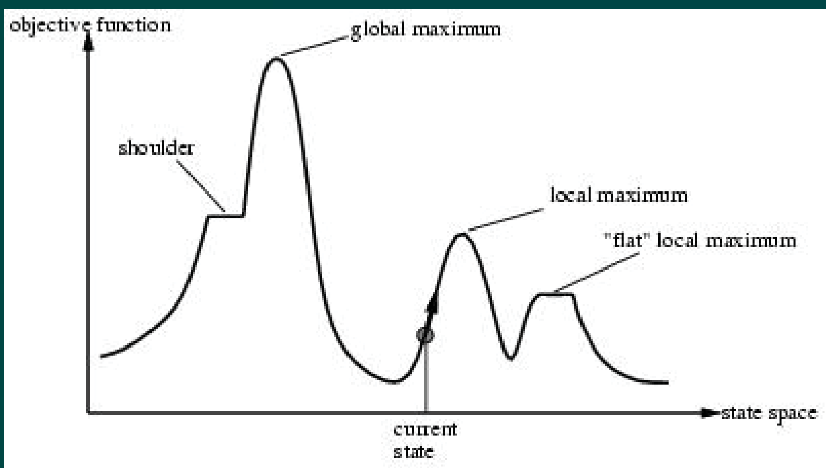
國立政治大學資訊科學系

ARTIFICIAL NEURAL NETWORKS
STOCHASTIC GRADIENT SEARCH

找尋函數的極值
Hill Climbing Search

Local Search

- Pure optimization problems: paths to the solution do not matter.



Source: Russel AIMA

函數曲面的起伏

- 我們不總是能夠想像或者看到函數曲面的起伏
  - $f(x,y) = 3x^3 + 2x^2y - 8xy^2 + 3$
  - $f(1,3) = 3 + 6 - 72 + 3 = -60$
- 如果所欲處理的函數沒有精確(或者說很難有)的公式解的話，就可能可以靠搜尋(search)的技術來找極值

Ridge

- 進行 local search 的時候，如何定義“鄰居”
- 如果鄰居定義得不好，可能會造成問題
- 視覺化：
  - https://academo.org/demos/3d-surface-plotter/?expression=x*x-y*y&xRange=-60%2C%2B60&yRange=-60%2C%2B60&resolution=40 (check (x,y)=(54,0))

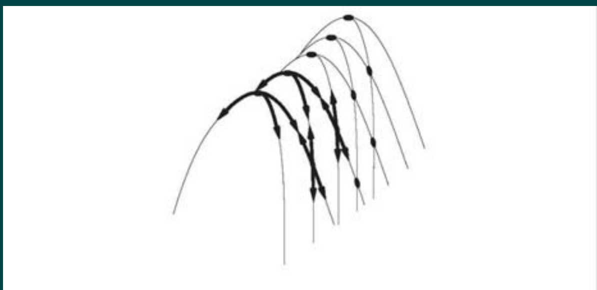


Figure 4.4 Illustration of why ridges cause difficulties for hill climbing. The grid of states (dark circles) is superimposed on a ridge rising from left to right, creating a sequence of local maxima that are not directly connected to each other. From each local maximum, all the available actions point downhill.

Source: Russel AIMA

程式操作

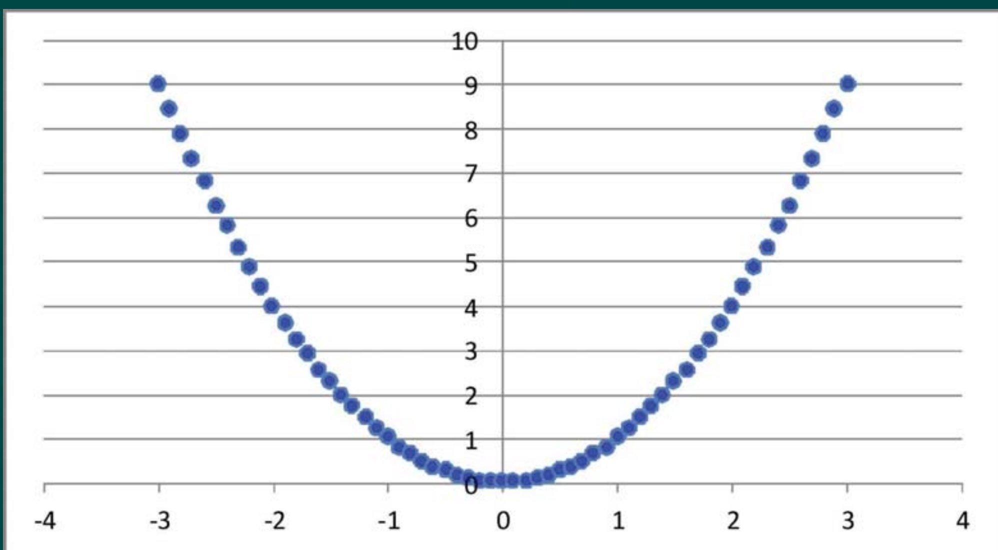
Illustration

- introAI.hill.climbing.ipynb

一個極度簡單的例子

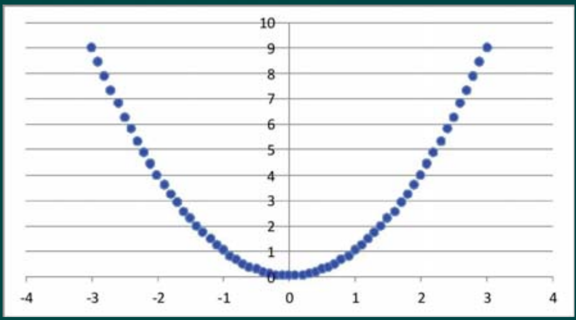
A Very Simple Case

$$y = f(x) = x^2$$



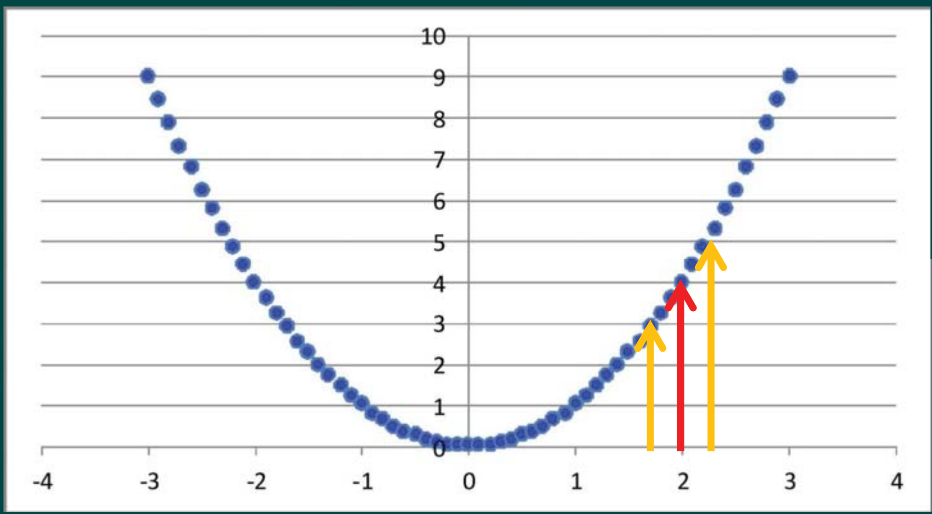
Search for Solution

- Why
  - 並不是所有的問題都有簡單的公式解
- 試試看search
  - 問問數學家？Data scientists? AI experts?



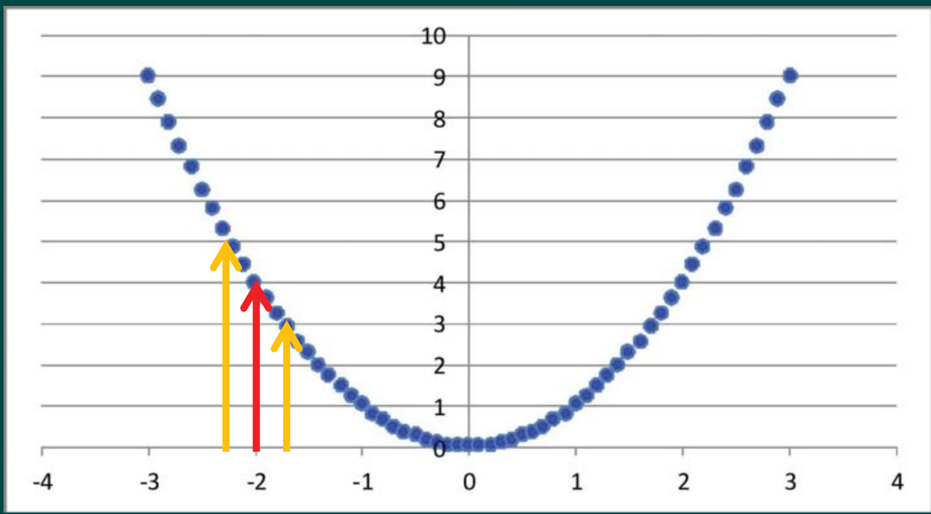
A Basic Search Idea

- 如果先亂猜一個數字；然後尋找下一步



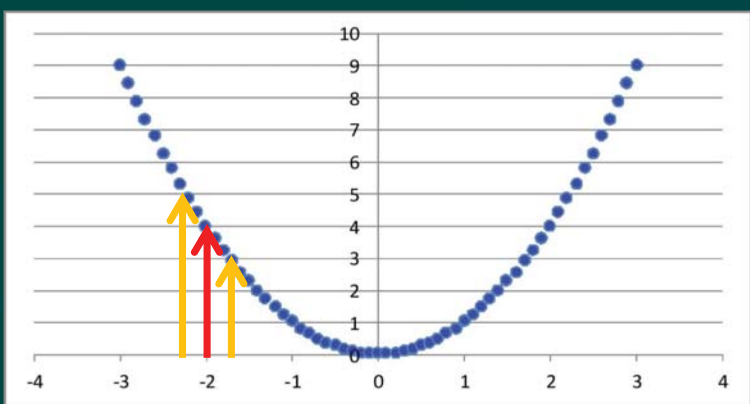
A Basic Search Idea

- 你當然可能先猜一個負數
- $x^{(i+1)} = x^i + \Delta x; \Delta x = ?$



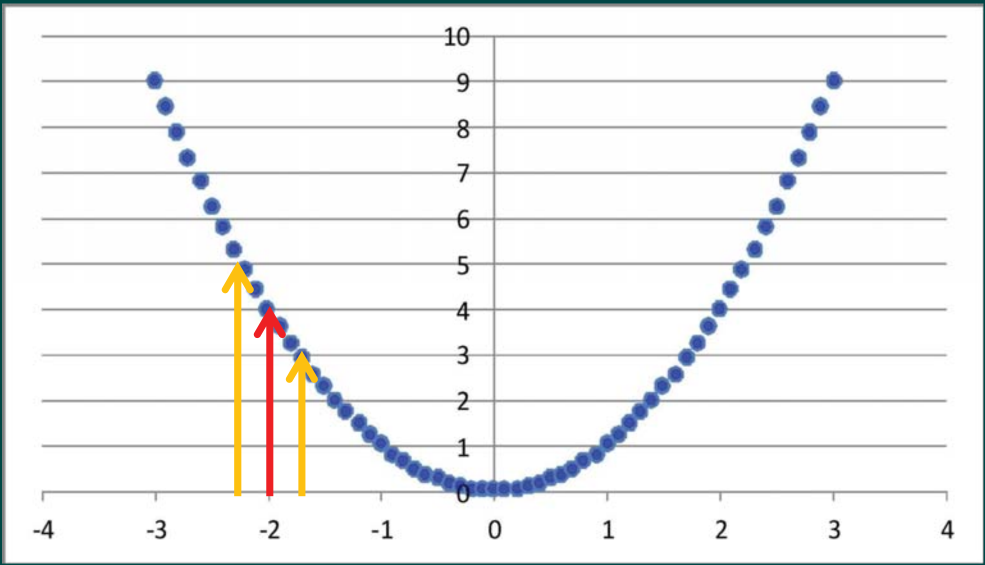
調整的方向和幅度

- $x^{(i+1)} = x^i + \Delta x; \Delta x = ?$
- $\frac{dy}{dx} = \frac{d(x^2)}{dx} = 2x$
- $\Delta x \propto \frac{dy}{dx}$
- $\Delta x = -\lambda f'(x)$



一個可能的方案

$$x^{(i+1)} = x^i - \lambda f'(x^i)$$



找尋函數的極值依賴切線斜率

斜率和切線

- 斜率和切線提供了數值變化趨勢的訊息
- 斜率
  - https://www.khanacademy.org/math/algebra/two-var-linear-equations/slope/v/positive-and-negative-slope

$$m = \frac{\Delta y}{\Delta x}$$

- 切線
  - https://www.khanacademy.org/math/ap-calculus-ab/ab-differentiation-1-new/ab-2-1/v/secant-lines-and-average-rate-of-change

公式解(求最小值)

$$y = f(x) = x^2$$

$$\frac{dy}{dx} = \frac{d(x^2)}{dx} = 2x$$

令 $\frac{dy}{dx} = 2x = 0 \Rightarrow x = 0$

簡單的程序

$$y = f(x)$$

$$\frac{dy}{dx} = \frac{df(x)}{dx}$$

$$guess = guess - \lambda \frac{dy}{dx}$$

$$guess = guess - \lambda y'$$

簡單的程序(2)

$$y = f(x_1, x_2)$$

$$\frac{\partial y}{\partial x_1} = \frac{\partial f(x_1, x_2)}{\partial x_1} \qquad \frac{\partial y}{\partial x_2} = \frac{\partial f(x_1, x_2)}{\partial x_2}$$

$$\hat{x}_1 = \hat{x}_1 - \lambda \frac{\partial y}{\partial x_1} \qquad \hat{x}_2 = \hat{x}_2 - \lambda \frac{\partial y}{\partial x_2}$$

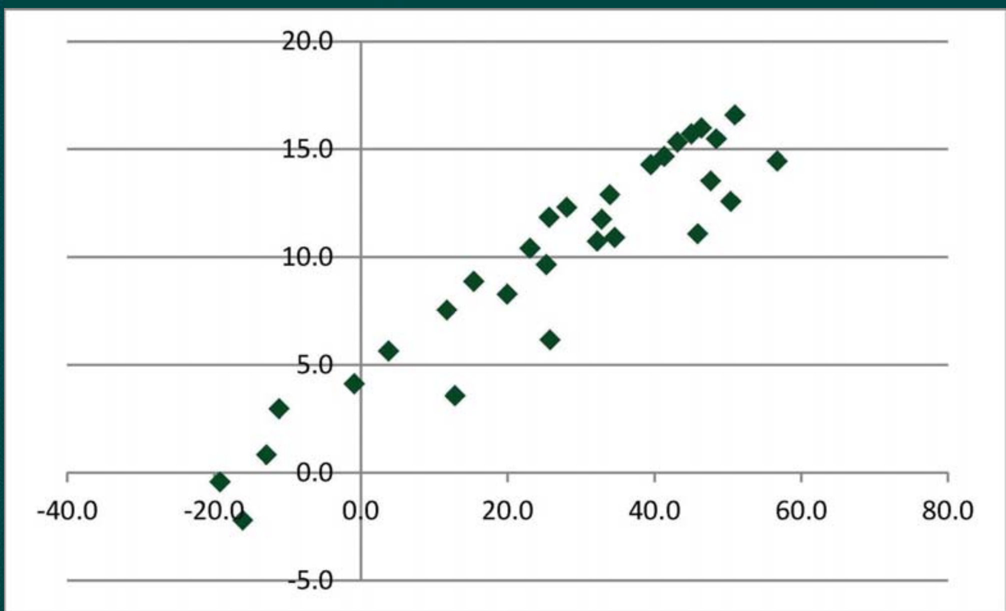
程式操作

Python Practice

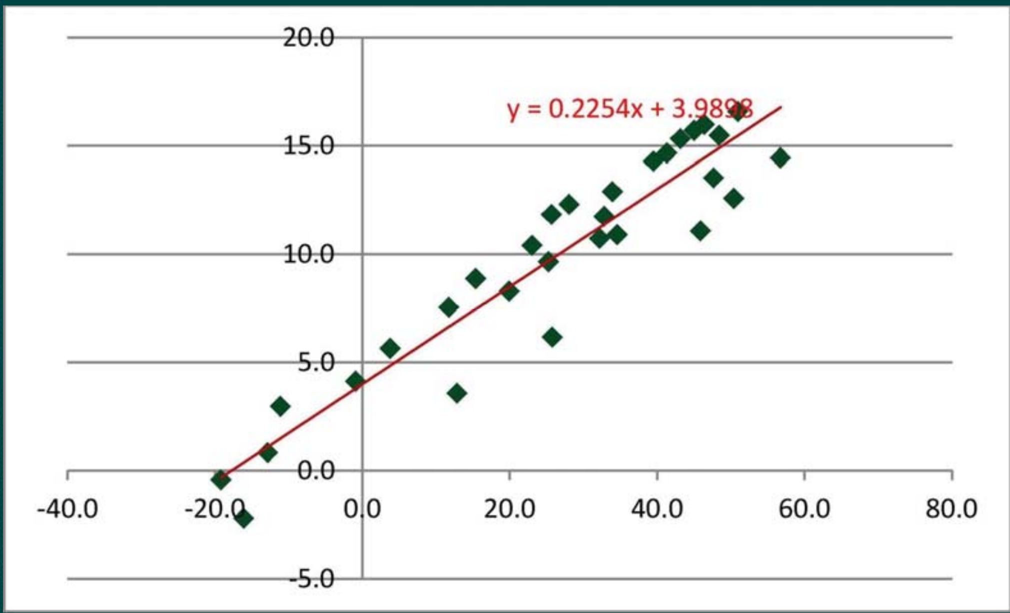
- simple.case.ipynb
- simple.case.two.variables.ipynb

LINEAR REGRESSION

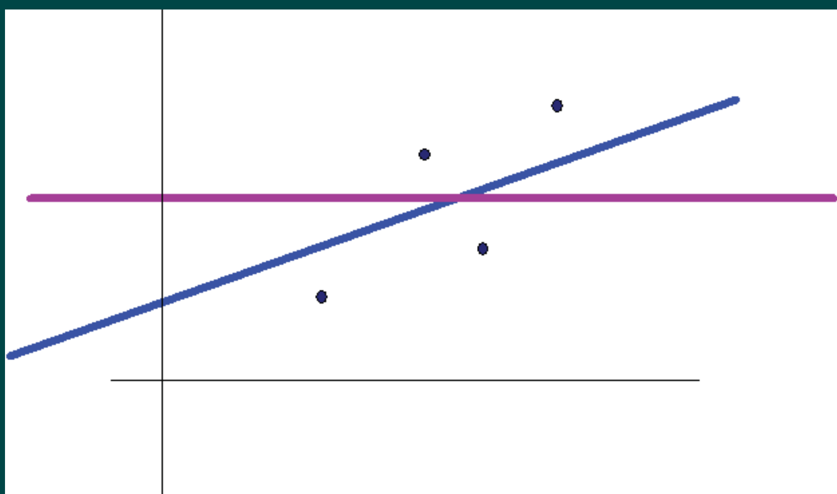
Randomized – Strong Noise



Learning from Real Data



哪一條線比較好？



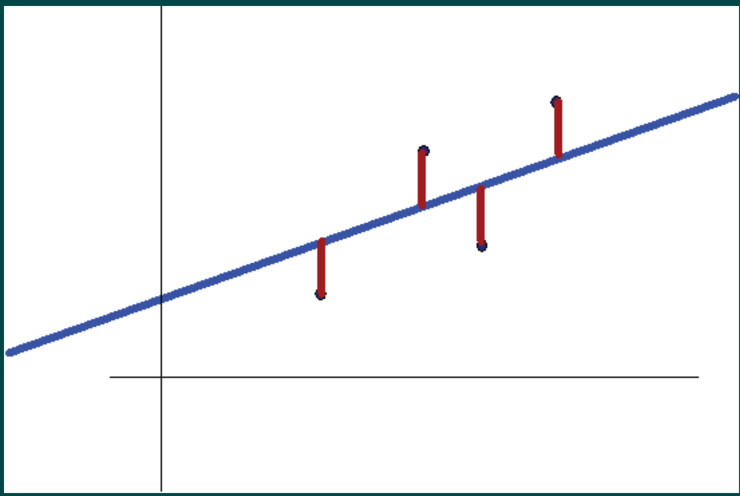
LINEAR REGRESSION 的公式解

我們投資過很多時間

LINEAR REGRESSION 的非公式解

誤差(殘差)

- 使用E_1會正負互相抵消



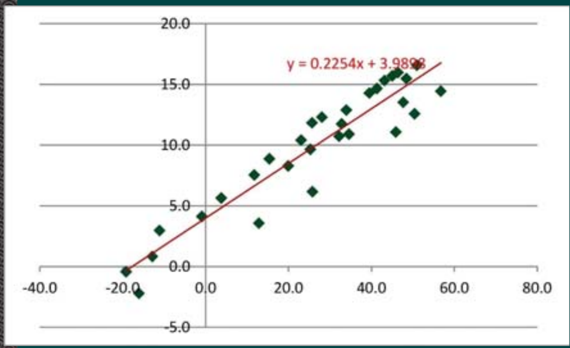
回顧先前關於LR的討論

- 我們尋找一條直線，希望能夠將總和誤差極小化
- 下星期我們延續函數極小化的議題

X	Y	$\hat{Y}$	E_1	E_2	E_3
X_0	Y_0	$\hat{Y}_0$	$Y_0-\hat{Y}_0$	$\lvert Y_0-\hat{Y}_0\rvert$	$(Y_0-\hat{Y}_0)^2$
X_1	Y_1	$\hat{Y}_1$	$Y_1-\hat{Y}_1$	$\lvert Y_1-\hat{Y}_1\rvert$	$(Y_1-\hat{Y}_1)^2$
X_2	Y_2	$\hat{Y}_2$	$Y_2-\hat{Y}_2$	$\lvert Y_2-\hat{Y}_2\rvert$	$(Y_2-\hat{Y}_2)^2$
X_3	Y_3	$\hat{Y}_3$	$Y_3-\hat{Y}_3$	$\lvert Y_3-\hat{Y}_3\rvert$	$(Y_3-\hat{Y}_3)^2$

重整一些先前的基礎

$$\hat{y} = \sum_{i=0}^{k} w_i a_i \qquad \hat{y}_j = \sum_{i=0}^{k} w_i a_i^j \qquad E = \sum_{j=1}^{n} \left(\hat{y}_j - y_j\right)^2$$



X	Y	$\hat{Y}$	E_1	E_2	E_3
X_0	Y_0	$\hat{Y}_0$	Y_0-$\hat{Y}_0$	$Y_0-\hat{Y}_0$	$\vert(Y_0-\hat{Y}_0)^2$
X_1	Y_1	$\hat{Y}_1$	Y_1-$\hat{Y}_1$	$Y_1-\hat{Y}_1$	$\vert(Y_1-\hat{Y}_1)^2$
X_2	Y_2	$\hat{Y}_2$	Y_2-$\hat{Y}_2$	$Y_2-\hat{Y}_2$	$\vert(Y_2-\hat{Y}_2)^2$
X_3	Y_3	$\hat{Y}_3$	Y_3-$\hat{Y}_3$	$Y_3-\hat{Y}_3$	$\vert(Y_3-\hat{Y}_3)^2$

工作目標

- 找尋一組$w_0, w_1, w_2, \ldots, w_k$，使得MSE為最小

$$\mathrm{MSE} = \frac{1}{n} \sum_{j=1}^{n} \left(\hat{y}_j - y_j\right)^2$$

- 可以利用找尋函數最小值的方法

一些微分的回顧

複習一些微分

- 多項式微分

$$y = f(x) = ax^n$$

$$\frac{dy}{dx} = f'(x) = anx^{n-1}$$

$$z = f(x) = 3x^2 - 2x + 8$$

$$\frac{dz}{dx} = f'(x) = 6x - 2$$

- 特殊函數的微分

$$y = f(x) = e^x$$

$$\frac{dy}{dx} = f'(x) = e^x$$

多項式的微分

- 如果 x 和 y 是獨立的變數的時候，我們將一個 $f(x,y)$ 的函數對於 x 做微分的時候，是把 y 當作一個常數來看待。

$$z = f(x,y) = 3x^3 + 3x^2y - 2xy + 8$$

$$\frac{dz}{dx} = f'(x,y) = 9x^2 + 6xy - 2y$$

- 如果 x 和 y 有關係的話，則微分會比較複雜。

$$y = f(x) = 3x^2 - 2x \qquad z = f(x,y) = 3x^2y - 2y + 8$$

$$\frac{dz}{dx} = f'(x,y) = 6xy + 3x^2\frac{dy}{dx} - 2\frac{dy}{dx}$$

$$\frac{dz}{dx} = f'(x,y) = 6x(3x^2 - 2x) + 3x^2(6x - 2) - 2(6x - 2) = 36x^3 - 18x^2 - 12x + 4$$

Sigmoid 的微分(複雜版本)

$$y = f(x) = \frac{1}{1 + e^{-x}} \qquad \frac{dy}{dx} = f'^{(x)} = \frac{d\left(\frac{1}{1+e^{-x}}\right)}{dx} = \frac{d(1+e^{-x})^{-1}}{dx}$$

$$let\ z = 1 + e^{-x} \qquad \frac{dy}{dx} = \frac{d(z^{-1})}{dx} = \frac{d(z^{-1})}{dz}\frac{dz}{dx} = \frac{d(z^{-1})}{dz}\frac{d(1+e^{-x})}{dx}$$

$$let\ w = e^{x} \qquad \frac{dy}{dx} = \frac{d(z^{-1})}{dz}\frac{d(1+e^{-x})}{dx} = \frac{d(z^{-1})}{dz}\frac{d(w^{-1})}{dx} = \frac{d(z^{-1})}{dz}\frac{d(w^{-1})}{dw}\frac{dw}{dx}$$

$$\frac{dy}{dx} = \frac{d(z^{-1})}{dz}\frac{d(w^{-1})}{dw}\frac{dw}{dx} = (-z^{-2})(-w^{-2})e^{x}$$

$$\frac{dy}{dx} = \frac{-1}{(1+e^{-x})^2}\left(\frac{-1}{(e^{x})^2}\right)e^{x} = \frac{1}{(1+e^{-x})^2}\left(\frac{1}{e^{x}}\right) = \frac{1}{(1+e^{-x})^2}e^{-x}$$

$$\frac{dy}{dx} = \frac{1}{(1+e^{-x})^2}e^{-x} = \frac{1}{1+e^{-x}}\frac{e^{-x}}{1+e^{-x}} = \frac{1}{1+e^{-x}}\left(1 - \frac{1}{1+e^{-x}}\right) = y(1-y)$$

更多微分的回顧
CHAIN RULE

Illustrations

$$y = f(x) = 2x + 1$$

$$z = f(y) = y^2$$

$$z = f(y) = y^2 = (2x+1)^2 = 4x^2 + 4x + 1$$

$$\frac{dz}{dx} = \frac{d(4x^2 + 4x + 1\,)}{dx} = \frac{d(4x^2 + 4x + 1\,)}{dx} = 8x + 4$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = \frac{d(y^2)}{dy}\frac{d(2x+1)}{dx} = 2y(2) = 8x + 4$$

Illustrations (2)

$$y = f(x) = (2x+1)^2 = 4x^2 + 4x + 1$$

$$z = f(y) = 2y + 1$$

$$z = f(y) = 2y + 1 = 2(2x+1)^2 + 1$$

$$\frac{dz}{dx} = \frac{d(2(2x+1)^2 + 1)}{dx} = \frac{d(8x^2 + 8x + 3)}{dx} = 16x + 8$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = \frac{d(2y+1)}{dy}\frac{d\left((2x+1)^2\right)}{dx} = 2(8x+4) = 16x + 8$$

Illustration(3, 變數代換)

$$y = f(x) = (2x+1)^2 = 4x^2 + 4x + 1$$

$$z = f(y) = 2y + 1$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = \frac{d(2y+1)}{dy}\frac{d\left((2x+1)^2\right)}{dx} = 2(8x+4) = 16x + 8$$

$$let\ m = 2x + 1$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = \frac{d(2y+1)}{dy}\frac{d\left((2x+1)^2\right)}{dx} = \frac{d(2y+1)}{dy}\frac{d(m^2)}{dm}\frac{dm}{dx}$$

$$\frac{dz}{dx} = \frac{dz}{dy}\frac{dy}{dx} = \frac{d(2y+1)}{dy}\frac{d(m^2)}{dm}\frac{dm}{dx} = 2(2m)(2) = 8m = 16x + 8$$

Multiple dependency

$$z = f(x,y) = 3x + 2y$$

$$x = g(s,t) = 5s + 6t$$

$$y = h(s,t) = 8s - 7t$$

$$z = f(x,y) = 3(5s + 6t) + 2(8s - 7t) = 31s + 4t$$

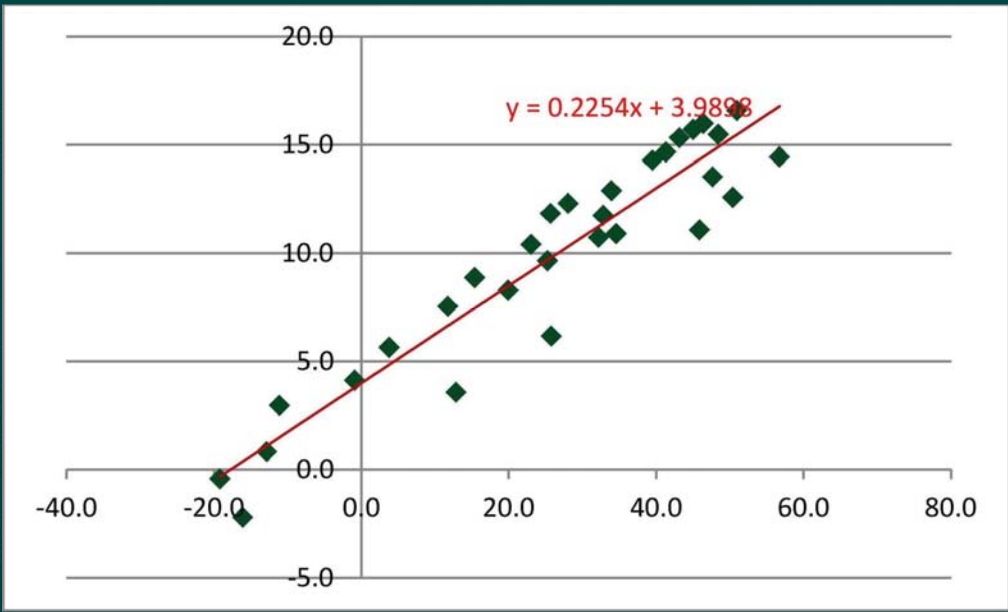
$$\frac{\partial z}{\partial s} = 31 \quad \frac{\partial z}{\partial t} = 4$$

$$\frac{\partial z}{\partial s} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial s} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial s} = 3 \times 5 + 2 \times 8 = 31$$

$$\frac{\partial z}{\partial t} = \frac{\partial z}{\partial x}\frac{\partial x}{\partial t} + \frac{\partial z}{\partial y}\frac{\partial y}{\partial t} = 3 \times 6 + 2 \times (-7) = 4$$

推導 MSE 的微分 (GRADIENT)

Learning from Real Data



回顧先前關於LR的討論

- 我們尋找一條直線，希望能夠將總和誤差極小化
- 下星期我們延續函數極小化的議題

X	Y	$\hat{Y}$	E_1	E_2	E_3
X_0	Y_0	$\hat{Y}_0$	$Y_0-\hat{Y}_0$	$\lvert Y_0-\hat{Y}_0\rvert$	$(Y_0-\hat{Y}_0)^2$
X_1	Y_1	$\hat{Y}_1$	$Y_1-\hat{Y}_1$	$\lvert Y_1-\hat{Y}_1\rvert$	$(Y_1-\hat{Y}_1)^2$
X_2	Y_2	$\hat{Y}_2$	$Y_2-\hat{Y}_2$	$\lvert Y_2-\hat{Y}_2\rvert$	$(Y_2-\hat{Y}_2)^2$
X_3	Y_3	$\hat{Y}_3$	$Y_3-\hat{Y}_3$	$\lvert Y_3-\hat{Y}_3\rvert$	$(Y_3-\hat{Y}_3)^2$

逐步推導

$$\text{MSE} = \frac{1}{n}\sum_{j=1}^{n}(\hat{y}_j - y_j)^2$$

$$\hat{y}_j = \sum_{i=0}^{k} w_i a_i^j$$

$$E = \frac{1}{n}\sum_{j=1}^{n}\left(\sum_{i=0}^{k} w_i a_i^j - y_j\right)^2$$

$$E = \frac{1}{n}\sum_{j=1}^{n}\left(w_0 a_0^j + w_1 a_1^j + w_2 a_2^j + \cdots + w_k a_k^j - y_j\right)^2$$

看起來有點困難的微分

$$E = \frac{1}{n}\sum_{j=1}^{n}\left(w_0 a_0^j + w_1 a_1^j + w_2 a_2^j + \cdots + w_k a_k^j - y_j\right)^2$$

$$\frac{\partial E}{\partial w_i} = \frac{\partial\left(\frac{1}{n}\sum_{j=1}^{n}\left(w_0 a_0^j + w_1 a_1^j + w_2 a_2^j + \cdots + w_i a_i^j + \cdots + w_k a_k^j - y_j\right)^2\right)}{\partial w_i}$$

$$\frac{\partial E}{\partial w_i} = \frac{2}{n}\sum_{j=1}^{n} a_i^j\left(w_0 a_0^j + w_1 a_1^j + w_2 a_2^j + \cdots + w_i a_i^j + \cdots + w_k a_k^j - y_j\right)$$

$$\frac{\partial E}{\partial w_i} = \frac{2}{n}\sum_{j=1}^{n} a_i^j(\hat{y}_j - y_j)$$

更新步驟

$$E = \frac{1}{n}\sum_{j=1}^{n}\left(w_0 a_0^j + w_1 a_1^j + w_2 a_2^j + \cdots + w_k a_k^j - y_j\right)^2$$

$$\frac{\partial E}{\partial w_i} = \frac{2}{n}\sum_{j=1}^{n} a_i^j\left(\hat{y}_j - y_j\right)$$

$$\widehat{w}_i = \widehat{w}_i - \lambda\frac{\partial E}{\partial w_i}$$

程式操作

Python Practice

- Linear Regression using Gradient Descent.ipynb
- Linear Regression using Gradient Descent Multivariates.ipynb

比較公式解的推導 (講過很多次)

尋求參數使整體誤差的最小化

- 如果我們選擇E_3

$$e_i = (y_i - \hat{y}_i)^2$$

$$e_i = (y_i - f(x_i)\,)^2$$

$$e_i = (y_i - ax_i - b)^2$$

$$\text{Error} = \textstyle\sum_{i=1}^{i=n} e_i$$

$$\text{Error} = \textstyle\sum_{i=1}^{i=n} (y_i - ax_i - b)^2$$

$$\text{E}= \textstyle\sum_{i=1}^{i=n} (y_i - ax_i - b)^2$$

$$\frac{\partial E}{\partial a} = -\sum_{i=1}^{i=n} 2x_i(y_i - ax_i - b) = 0$$

$$\frac{\partial E}{\partial b} = -\sum_{i=1}^{i=n} 2(y_i - ax_i - b) = 0$$

多一些推導工作

$$\frac{\partial E}{\partial a} = -\sum_{i=1}^{i=n} 2x_i(y_i - ax_i - b) = 0 \longrightarrow a\sum_{i=1}^{i=n} x_i^2 + b\sum_{i=1}^{i=n} x_i - \sum_{i=1}^{i=n} x_i y_i = 0$$

$$\frac{\partial E}{\partial b} = -\sum_{i=1}^{i=n} 2(y_i - ax_i - b) = 0 \longrightarrow a\sum_{i=1}^{i=n} x_i + nb - \sum_{i=1}^{i=n} y_i = 0$$

$$\downarrow$$

$$b = \frac{1}{n}\sum_{i=1}^{i=n} y_i - \frac{a}{n}\sum_{i=1}^{i=n} x_i$$

$$\downarrow$$

$$b = \frac{1}{n}\sum_{i=1}^{i=n} y_i - \frac{a}{n}\sum_{i=1}^{i=n} x_i$$

$$\downarrow$$

$$b = \bar{Y} - a\bar{X}$$

你要自己完成剩下的部分嗎？

$$-\sum_{i=1}^{i=n} 2x_i(y_i - ax_i - b) = 0 \longrightarrow \sum_{i=1}^{i=n} (ax_i + b - y_i)x_i = 0$$

$$b = \bar{Y} - a\bar{X}$$

$$\Big\downarrow ?$$

$$a = \frac{\sum_{i=1}^{i=n}(y_i - \bar{Y})(x_i - \bar{X})}{\sum_{i=1}^{i=n}(x_i - \bar{X})^2}$$

Logistic Regression

基本原理

- Witten 第4.6節
- $y = \Pr(1|a_1, \ldots, a_k) = \sum_{i=0}^{i=k} w_i a_i$
  - 注意我們令$a_0 = 1$, w_0 就是原先的 intercept
  - 不容易限制 y 的範圍在[0,1]
- 改用下面這樣的關係，y的數值就相當自由

$$y = log\left(\frac{\Pr(1|a_1, \ldots, a_k)}{\Pr(0|a_1, \ldots, a_k)}\right) = log\left(\frac{\Pr(1|a_1, \ldots, a_k)}{1 - \Pr(1|a_1, \ldots, a_k)}\right) = \sum_{i=0}^{i=k} w_i a_i$$

- $Pr(1|a_1, \cdots, a_k) = \frac{1}{1+e^{-\sum_{i=0}^{i=k} w_i a_i}}$

公式推導的一些細節

$$log\left(\frac{\Pr(1|a_1, \ldots, a_k)}{1 - \Pr(1|a_1, \ldots, a_k)}\right) = \sum_{i=0}^{i=k} w_i a_i$$

$$\frac{\Pr(1|a_1, \ldots, a_k)}{1 - \Pr(1|a_1, \ldots, a_k)} = e^{\sum_{i=0}^{i=k} w_i a_i}$$

$$\Pr(1|a_1, \ldots, a_k) = (1 - \Pr(1|a_1, \ldots, a_k))e^{\sum_{i=0}^{i=k} w_i a_i}$$

$$\Pr(1|a_1, \ldots, a_k) = e^{\sum_{i=0}^{i=k} w_i a_i} - \Pr(1|a_1, \ldots, a_k)\, e^{\sum_{i=0}^{i=k} w_i a_i}$$

$$\Pr(1|a_1, \ldots, a_k)\left(e^{\sum_{i=0}^{i=k} w_i a_i} + 1\right) = e^{\sum_{i=0}^{i=k} w_i a_i}$$

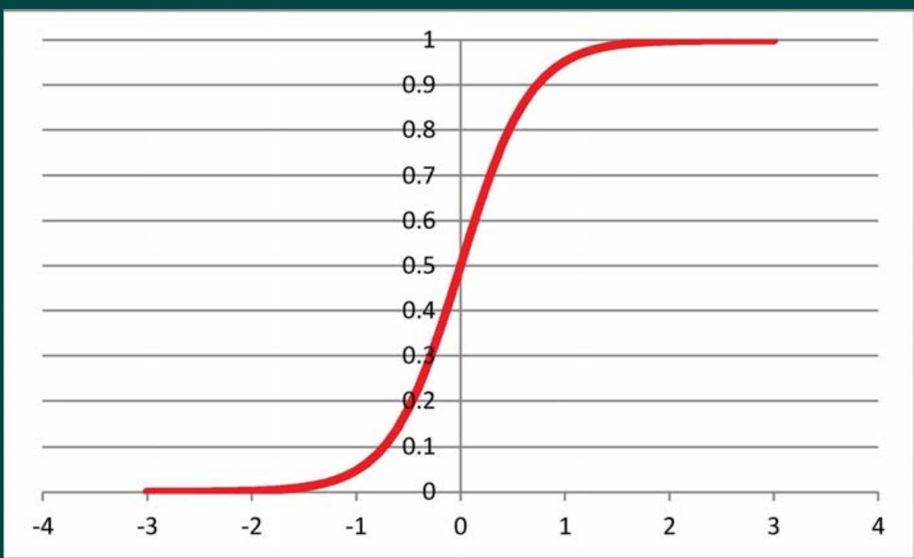
$$\Pr(1|a_1, \ldots, a_k) = \frac{e^{\sum_{i=0}^{i=k} w_i a_i}}{\left(e^{\sum_{i=0}^{i=k} w_i a_i} + 1\right)} = \frac{1}{\left(1 + e^{-\sum_{i=0}^{i=k} w_i a_i}\right)}$$

觀察數據趨勢

- $Pr(1|a_1, \cdots, a_k) = \frac{1}{1+e^{-\sum_{i=0}^{i=k} w_i a_i}}$
- Excel demonstration
- Application
  - Educational Measurement
  - Item Response Theory (IRT)

觀察基本圖形

- $\Pr(1|w_0 + w_1 a_1)\,; w_0 = 0, w_1 = 3$



Cross Entropy

KL Divergence

- KL divergence measures the "difference" between two probability distributions, p and q.

$$D(p\|q) = \sum_{i=1}^{n} p_i \log \frac{p_i}{q_i}$$
$$= \sum_{i=1}^{n} p_i (\log p_i - \log q_i)$$
$$= \sum_{i=1}^{n} (p_i \log p_i - p_i \log q_i)$$
$$= \sum_{i=1}^{n} p_i \log p_i - \sum_{i=1}^{n} p_i \log q_i$$

Therefore, the best q will minimize this quantity.

Let p be the "true" probability distribution. This quantity would be a constant for any q.

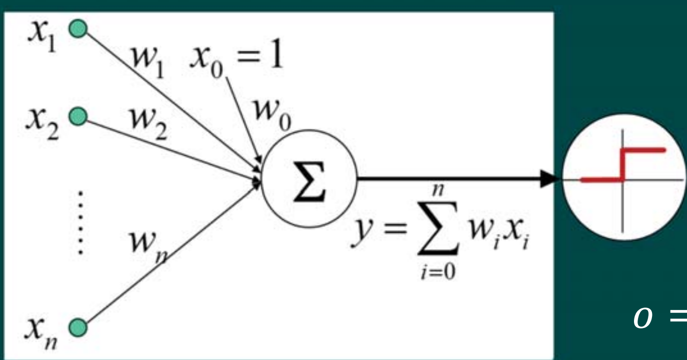
程式操作

資料分析二

- http://moodle.nccu.edu.tw/mod/forum/discuss.php?d=132365
- numpy.array.basic.intro.ipynb
- 說明學期程式作業的解答
  - assignment.ipynb
  - 上課說明，不可以拍照
  - 回家自己再做一次

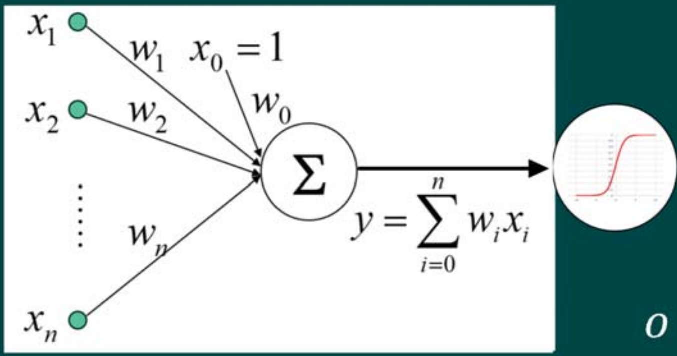
PERCEPTRON

One Simple Design



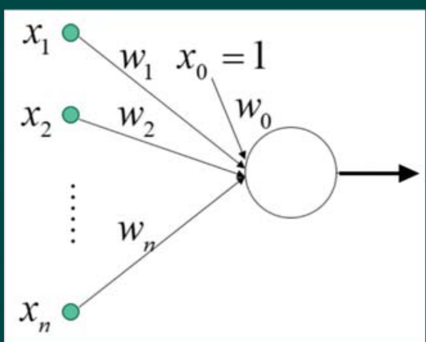
$$o = \begin{cases} 1 & if\ y \geq 0 \\ 0 & else \end{cases}$$

A Better Choice

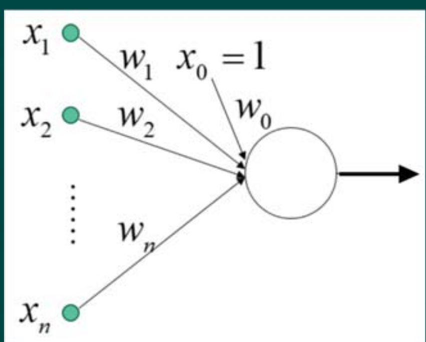


$$o = \frac{1}{1 + e^{-y}}$$

Simplified Representation

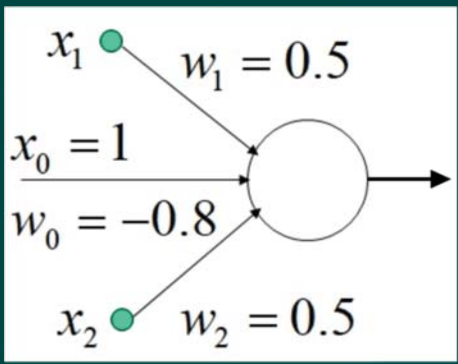


$$o = \begin{cases} 1 & if\ y \geq 0 \\ 0 & else \end{cases}$$



$$o = \frac{1}{1 + e^{-y}}$$

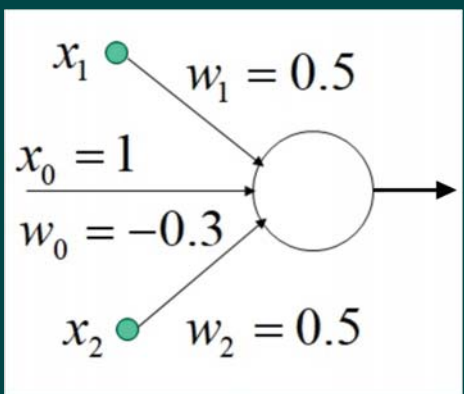
An AND Gate



$$o = \begin{cases} 1 & if\ y \geq 0 \\ 0 & else \end{cases}$$

x_1	x_2	output
0	0	0
0	1	0
1	0	0
1	1	1

An OR Gate

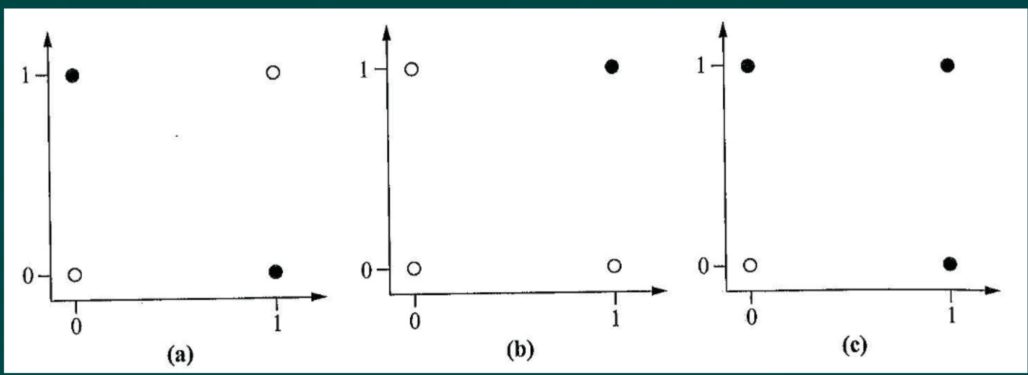


$$o = \begin{cases} 1 & if\ y \geq 0 \\ 0 & else \end{cases}$$

x_1	x_2	output
0	0	0
0	1	1
1	0	1
1	1	1

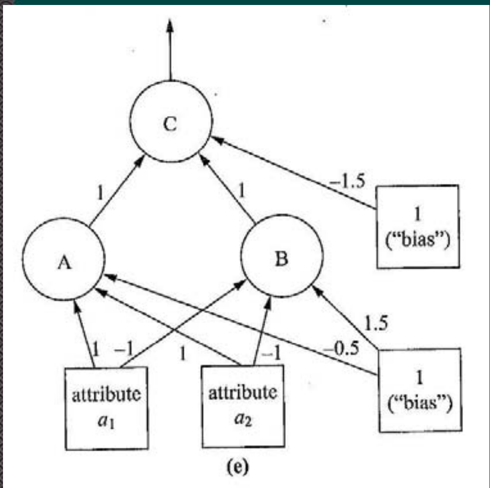
LINEARLY SEPARABE

Witten Figure 6.11 (Ed. 4)



An XOR Gate

- Witten Figure 6.11 (ed. 4)

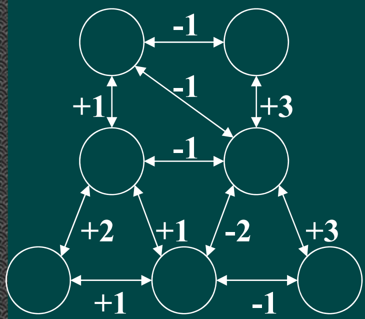


$$o = \begin{cases} 1 & if\ y \geq 0 \\ 0 & else \end{cases}$$

a_1	a_2	output
0	0	0
0	1	1
1	0	1
1	1	0

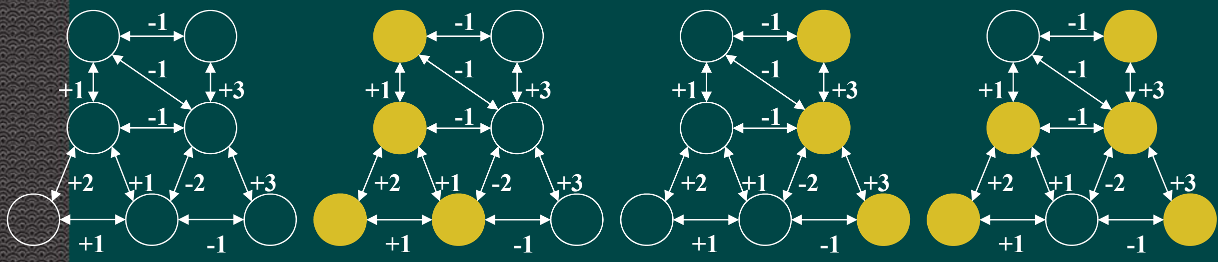
Hopfield Networks

A Simple Hopfield Network



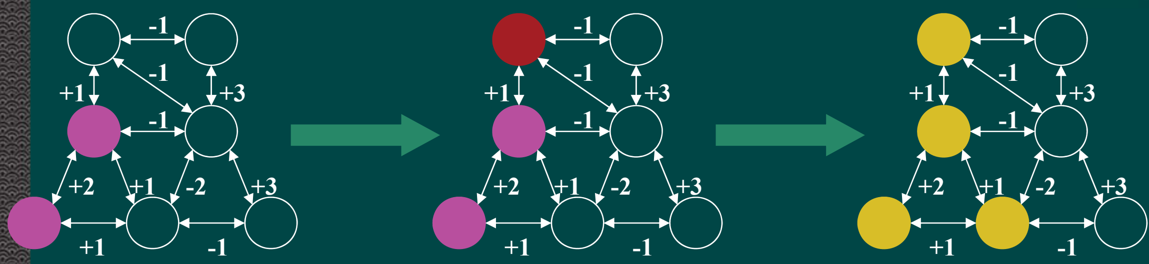
$$output = \begin{cases} 1 & \text{if } \sum w_i x_i > 0 \\ -1 & \text{otherwise} \end{cases}$$

four stable states

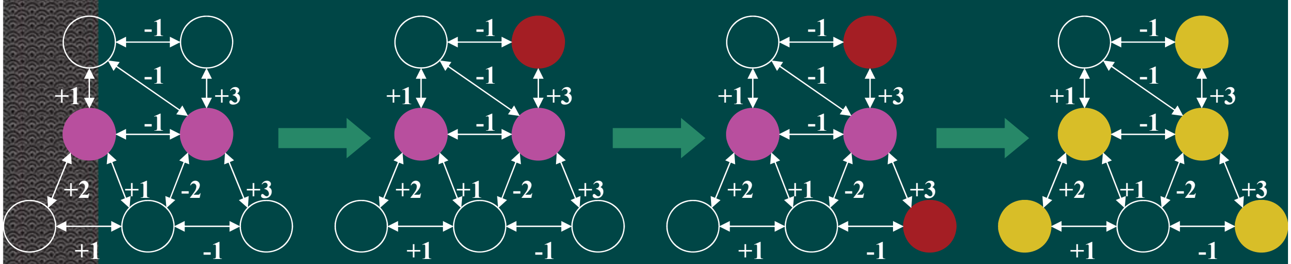


Rich and Knight, *Artificial Intelligence*, 2nd Ed., McGraw Hill

Converging to a Local Optimum

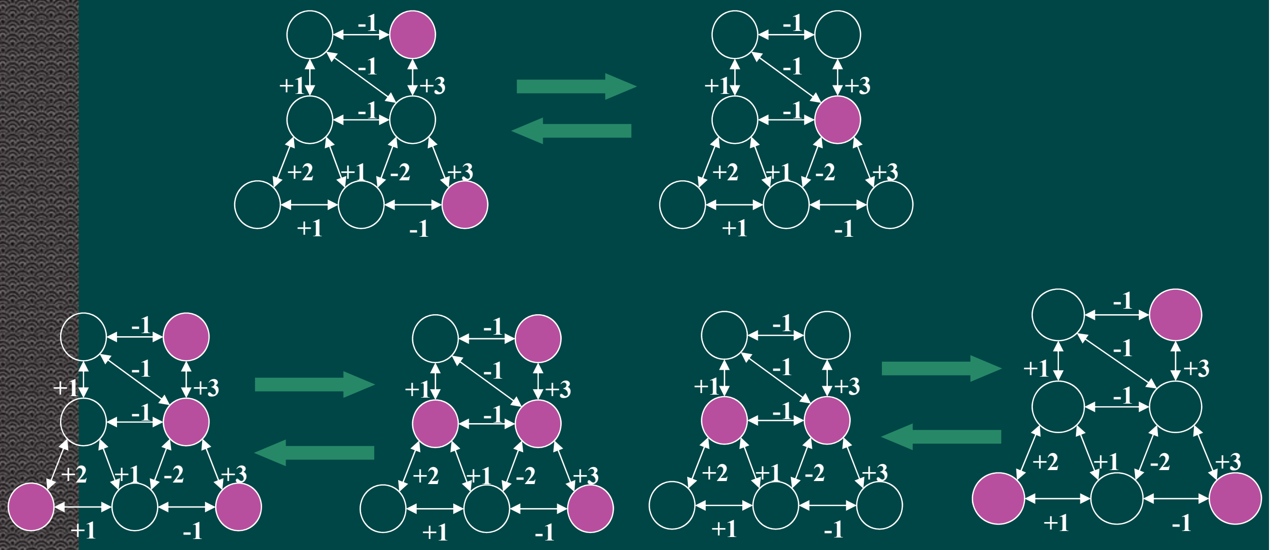


Notice that the steps are imaginary, each unit changes states in parallel.



Unstable Cases

Notice that the steps are imaginary, each unit changes states in parallel.



Simple Examples

Artificial Neural Networks

- https://machinelearningmastery.com/tutorial-first-neural-network-python-keras/
- https://www.tensorflow.org/tutorials/keras/basic_classification
- https://www.kaggle.com/miggle/my-first-neural-network

BACKPROPAGATION

Gradually Challenging Introductions

- Conceptual introduction
  - https://medium.com/datathings/neural-networks-and-backpropagation-explained-in-a-simple-way-f540a3611f5e
- A little bit more details
  - https://missinglink.ai/guides/neural-network-concepts/backpropagation-neural-networks-process-examples-code-minus-math/
- A Relatively Simple Illustration
  - https://hmkcode.github.io/ai/backpropagation-step-by-step/

Gradually Challenging Introductions(2)

- Becoming formal
  - https://google-developers.appspot.com/machine-learning/crash-course/backprop-scroll/
- More formal steps
  - https://www.kamperh.com/notes/kamper_backprop17.pdf
- The algorithm
  - Mitchell, p. 98
- Here is a sample Python program
  - https://www.kaggle.com/romaintha/an-introduction-to-backpropagation