Linear Models
- Classification and Regression

Jen-Ing Hwang
Department of Computer Science and
Information Engineering
Fu Jen Catholic University

Outline

- Introduction
- Linear Classification
- Linear Regression

Introduction (1/2)

- Supervised learning 監督式學習
  - The most popular paradigm for machine learning
  - Given a dataset with input attribute(s) and output label(s), discovering a relationship between the input and the output
  - Can be divided into classification and regression problems
    - Classification(分類): with nominal (名目式) output label(s)
    - Regression(迴歸): with numeric (數值式) output label(s)

Introduction (2/2)

- Linear models
  - Algorithms searching a hypothesis over the hyperplane(超平面) hypothesis space
- Hyperplane
  - The set of all points that are in R^M and satisfy a linear equation
  - In R^2, the hyperplanes are straight lines (直線，一維(2-1))
  - In R^3, hyperplanes are planes (平面，二維(3-1))
  - More generally, if the input instance space is *M*-dimensional then its hyperplanes are (*M*-1)-dimensional
  - 超過三度空間之後我們就沒有辦法畫了，所以超過三度空間的我們叫做超平面
  - 狹義的說法：在R4、R5、R6
  - 廣義的說法：在R2、R3、R4 、R5、R6

Outline

- Introduction
- Linear Classification
- Linear Regression

Illustrative Example for Linear Classification

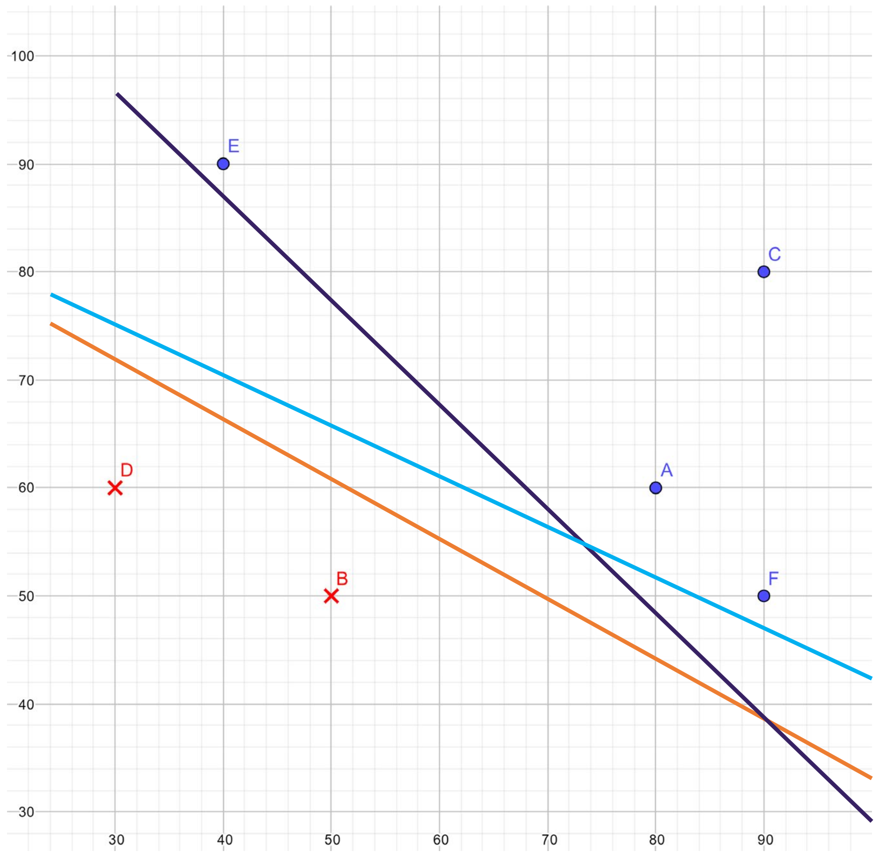
- Dataset *D*
  - – Students take a class on a **Pass/Fail** basis

Students	x_1 (Midterm)	x_2 (Final)	y (Pass/Fail)
A	80	60	+1 (Pass)
B	50	50	–1 (Fail)
C	90	80	+1 (Pass)
D	30	60	–1 (Fail)
E	40	90	+1 (Pass)
F	90	50	+1 (Pass)

Data Representation

- The dataset *D*
  - Training Examples: $(\mathbf{x}_A, y_A), (\mathbf{x}_B, y_B), \ldots, (\mathbf{x}_F, y_F)$
  - 註：粗體表示向量 ex: $\mathbf{x}_{A}$(第一個同學的兩個分量)
  - $\mathbf{x}_A, \mathbf{x}_B, \ldots, \mathbf{x}_F$: input
    - Each **x** is a vector with two attributes
    - $\mathbf{x}$ = (Midterm, Final) = $(x_1, x_2) = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$
  - $y_A, y_B, \ldots, y_F$: output (+1 or –1)

Hypothesis Space and Version Space



Hypothesis Space (假說空間) H = The set of straight lines

<註> 所有的假說形成的集合。是我認為一條直線可以把它分開，但是這條直線可能可以分開，可能不能分開，對於那些可以分開的我們叫它Version Space

Version Space $VS_{H,D}$ =
{All straight lines that separate positive and negative examples.}
註記： Version Space並不唯一
->如果點(data)越多，能找到正確的直線的可能性就變高了

We can apply the Perceptron Learning Algorithm (PLA) to find a consistent hypothesis(一致假設)

Perceptron Learning Algorithm (PLA) (1/2)

透過不斷的內積、修正錯誤，進而找到一條可以將資料分成兩類的演算法

- For input **x** = (x_1, x_2) //grades of a students
- Pass if $w_1 x_1 + w_2 x_2$ > threshold

  (($w_1 x_1 + w_2 x_2$)− threshold) > 0
- Fail if $w_1 x_1 + w_2 x_2$ < threshold

  (($w_1 x_1 + w_2 x_2$)− threshold) < 0
- This linear formula $h \in H$ can be written as
- h (**x**) = sign (($w_1 x_1 + w_2 x_2$) − threshold)

Perceptron Learning Algorithm (PLA) (2/2)

- $h(\mathbf{x})$ = sign $((w_1 x_1 + w_2 x_2) - \text{threshold})$

  Let $w_0 = -\text{threshold}$

  $\Rightarrow h(\mathbf{x})$ = sign $((w_1 x_1 + w_2 x_2) + w_0)$

  Introduce an artificial coordinate $x_0 = 1$ (杜撰的，為了讓式子變好看，可以讓它寫成內積)

  $\Rightarrow h(\mathbf{x})$ = sign $(w_0 x_0 + w_1 x_1 + w_2 x_2)$

- We can write $h(\mathbf{x})$ in vector form:

  $h(\mathbf{x})$ = sign $(\mathbf{w} \bullet \mathbf{x})$ // • inner product

  where $\mathbf{w} = (w_0, w_1, w_2)$ and $\mathbf{x} = (x_0, x_1, x_2)$ // $x_0 = 1$

What PLA Does? (1/4)

Given the training examples: $(\mathbf{x}_A, y_A), (\mathbf{x}_B, y_B), \ldots, (\mathbf{x}_F, y_F)$

if an input example, $\mathbf{x}_k$, is misclassified (它該+1時，你算出來是-1，它該是-1時，你算出來是+1)

(i.e., sign $(\mathbf{w} \bullet \mathbf{x}_k) \neq y_k$, $k \in \{A, B, \ldots, F\}$)

then update the weight vector **w**

$\mathbf{w} \leftarrow \mathbf{w} + y_k \mathbf{x}_k$ // Note that y_k is +1 or –1

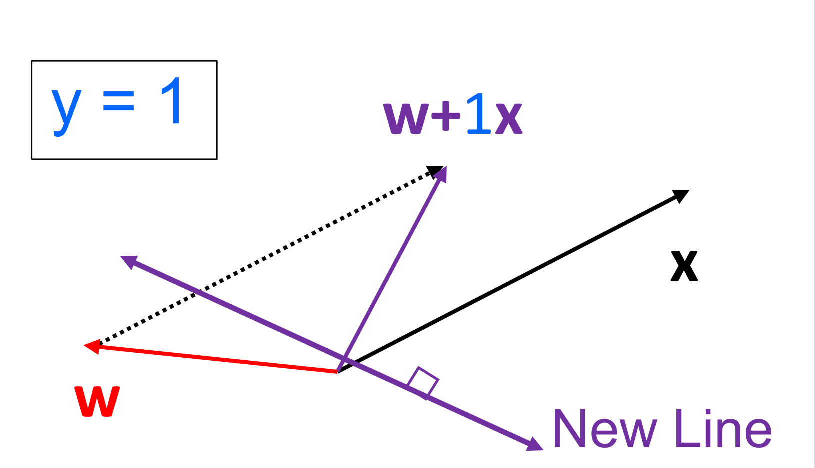
What PLA Does? (2/4)

- Two misclassified types for a training example ($\mathbf{x}$, y)
- Type 1: (答案應是1，我算出來的卻是-1)

  y = 1 (正解) but $h(\mathbf{x})$ = sign $(w_0 x_0 + w_1 x_1 + w_2 x_2)$ = –1 　(x_0、x_1、x_2和w_0、w_1、w_2在平面的兩側)

  $\Rightarrow$ **PLA: $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$**

  $\Rightarrow$ **PLA: $\mathbf{w} \leftarrow \mathbf{w} + 1\mathbf{x}$** 　// make **w** close to **x**
- W想成是法向量



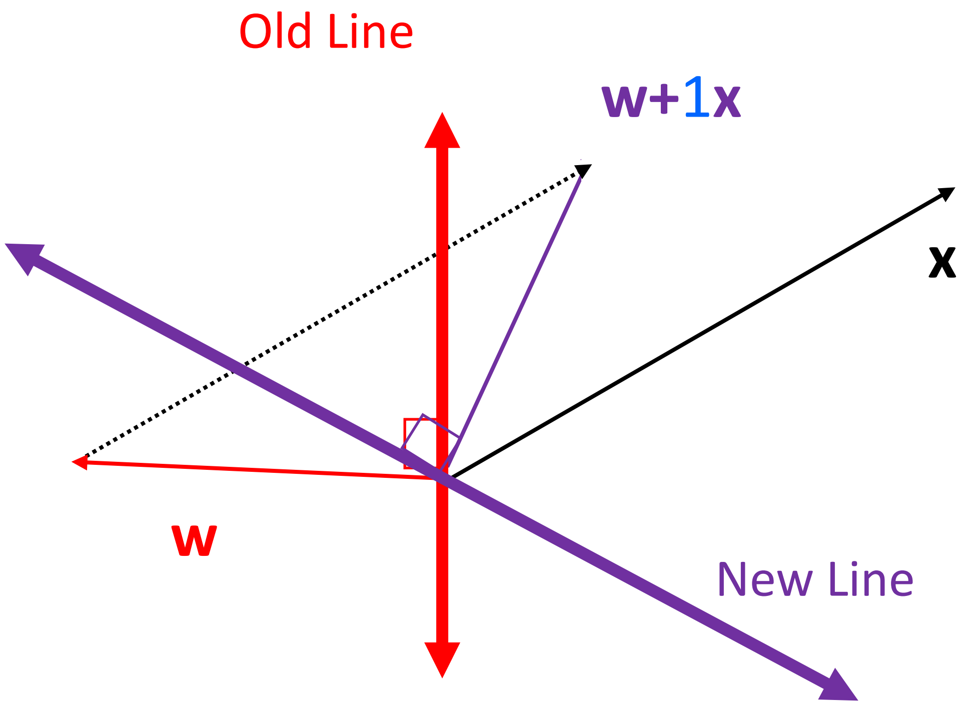
y = 1
我希望x_0、x_1、x_2和w_0、w_1、w_2在同一面

w : Normal vector of the line
x : The training example
w+1x : Normal vector of the new line

- w_0、w_1、w_2代表R_3空間的平面的法向量
- x_0、x_1、x_2代表R_3空間的任何一個點，只是有一個虛擬的x_0的X軸=1，所以我所有的x_0都是在1這個地方，但可以假想成自從引入了這個虛擬的X軸以後，好像data是在R_3空間
- 請問x_0、x_1、x_2和w_0、w_1、w_2的內積是多少？
- 若>0，表示W跟X是在同一個法向量的平面上
- 若<0，表示W跟X是在法向量的另外兩側

What PLA Does? (3/4)

y = 1



PLA:

w ← **w** + 1**x** // make **w** close to **x**

w : Normal vector of the line
x : The training example
w+1x : Normal vector of the new line
(怎麼加? Ans:平行四邊形的對角線)

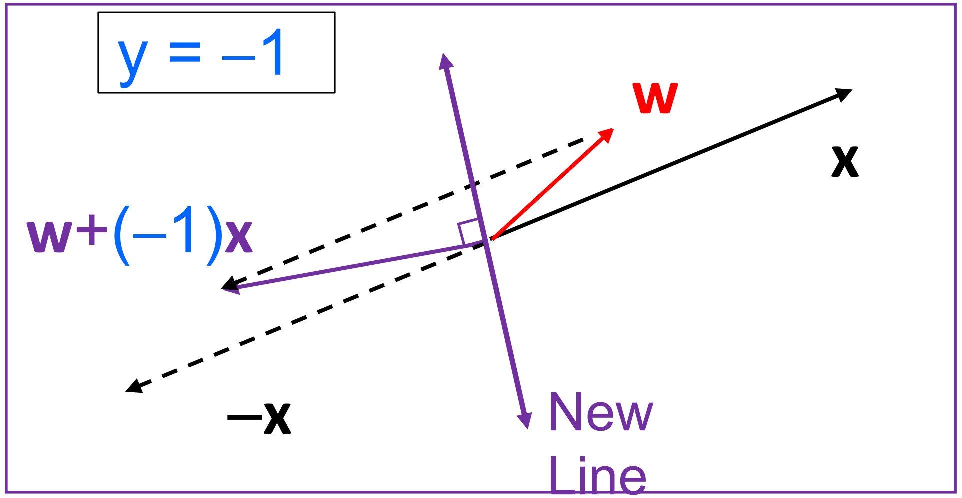
What PLA Does? (4/4)

- Type 2: (答案是-1，你卻算出來是+1，表示其實你的X應該要跟法向量是另外一邊)

  $y = -1$ but $h(\mathbf{x}) = \text{sign}\,(w_0 x_0 + w_1 x_1 + w_2 x_2) = 1$

  $\Rightarrow$ **PLA:** $\mathbf{w} \leftarrow \mathbf{w} + y\mathbf{x}$

  $\Rightarrow$ **PLA:** $\mathbf{w} \leftarrow \mathbf{w} + (-1)\mathbf{x}$ // move **w** away from **x**



w : Normal vector of the line

x : The training example

w+(–1)x : Normal vector of the new line

The PLA Algorithm

Epoch：scan所有的data一次

Given training dataset:

D = {($\mathbf{x}_1$, y_1), …, ($\mathbf{x}_N$, y_N)} // N training examples

Initialize all weights w_i to random values

// $\mathbf{w} = (w_0, w_1, \ldots, w_M)$; M+1 attributes

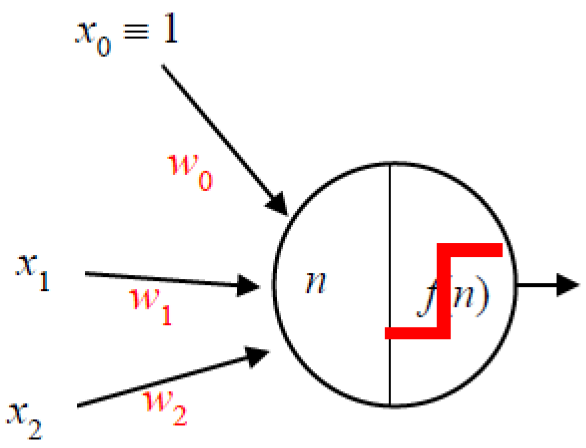
WHILE not all examples correctly predicted DO

FOR each training example $\mathbf{x}_k \in D$

If sign $(\mathbf{w} \bullet \mathbf{x}_k) \neq y_k$ then $\mathbf{w} \leftarrow \mathbf{w} + y_k\, \mathbf{x}_k$

Note: The PLA is guaranteed to converge if the training set is linearly separable.

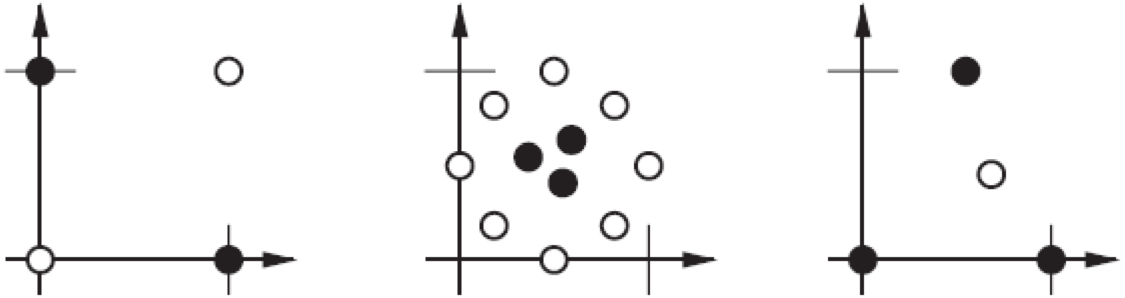
Perceptron Architecture



- Net input: $n = w_0 x_0 + w_1 x_1 + w_2 x_2$
- Activation Function:
  - $f(n)$ = sign (n)
  - Two different outputs
  - Binary Classification

Remarks on PLA (1/2)

- The decision boundary is always orthogonal (正交的) to the weight vector.
- Single-layer perceptrons can only classify linearly separable inputs(只能分類可線性分離的資料們).
- The PLA cannot solve linearly inseparable problems.
- Examples of linearly inseparable problems:



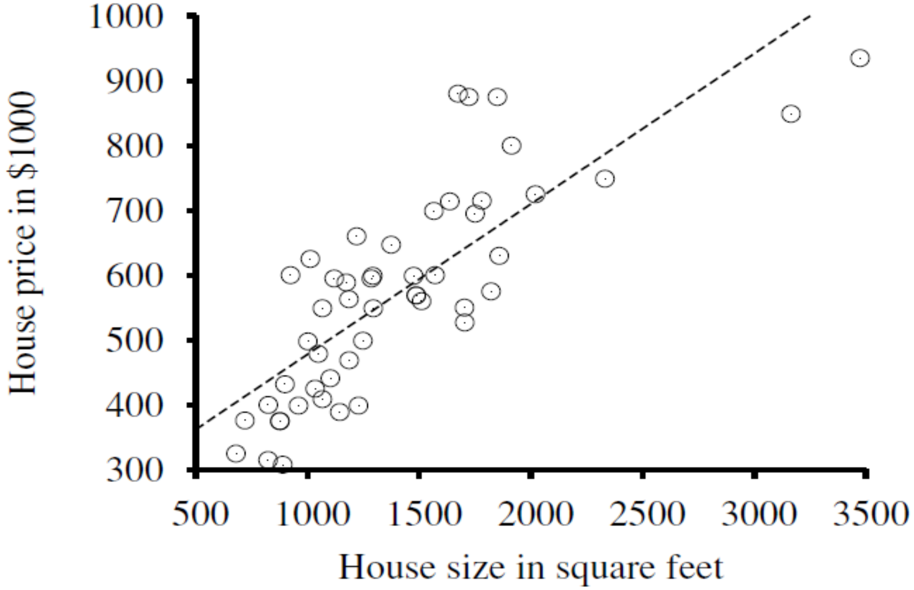
Remarks on PLA (2/2)

- Variants of PLA:
  - Activation Function: $a = f(n) = \begin{cases} 0 & if\ n < 0 \\ 1 & if\ n \geq 0 \end{cases}$
  - $\mathbf{w} \leftarrow \mathbf{w} + e\,\mathbf{x}$; e = (y – a)

Outline

- Introduction
- Linear Classification
- Linear Regression

Illustrative Example for Linear Regression



Simple Linear Regression: Finding a relationship between two continuous variables; x and $y = f(x)$

Data points of Price (= y) vs. House size (= x) for sale in Berkeley, CA, in July 2009, along with the linear regression that minimizes squared error: $\hat{y} = 0.232x + 246$.

Figure adapted from Fig 18.13 in Ref. [1]

Data Representation

- In general, a dataset has *N* examples with *M* input attributes

  N Training Examples: $(\mathbf{x}_1, y_1), (\mathbf{x}_2, y_2), \ldots, (\mathbf{x}_N, y_N)$

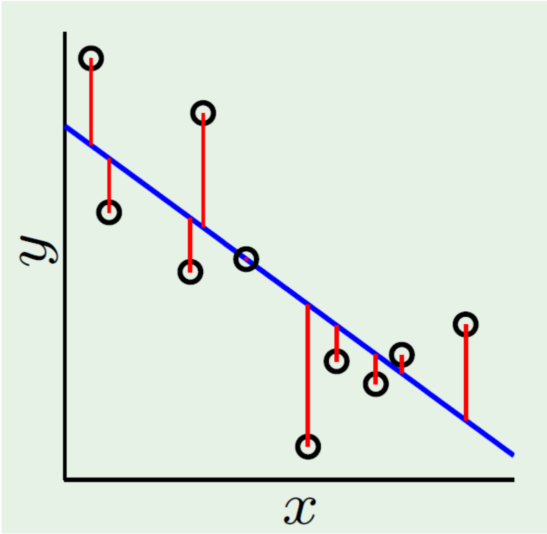
  $\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N$: input

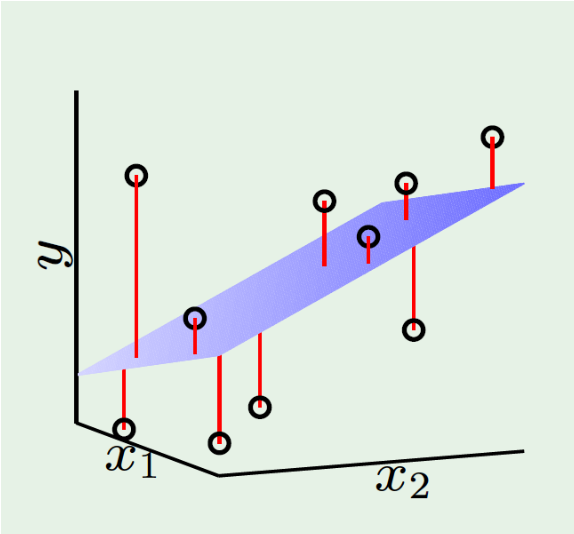
  Each **x** is a vector with M attributes

  $$\mathbf{x} = (x_1, x_2, \ldots, x_M) = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$$: *M* input attributes

  $y_1, y_2, \ldots, y_N$: output; each *y* is a real number

Illustration of Linear Regression





10 (= N) training examples
1 (= M) attribute; x

10 (= N) training examples
2 (= M) attributes; x_1, x_2

Figures adapted from Fig 3.3 in Ref. [2]

How to Measure the Error? (1/2)

- How well does a hypothesis function $h(\mathbf{x})$ approximate the target function $f(\mathbf{x})$?
- In linear regression, we can measure the accuracy of our hypothesis function using an error function (又稱 cost function or loss function).
- One common measure:

the squared error $(h(\mathbf{x}) - f(\mathbf{x}))^2$

How to Measure the Error? (2/2)

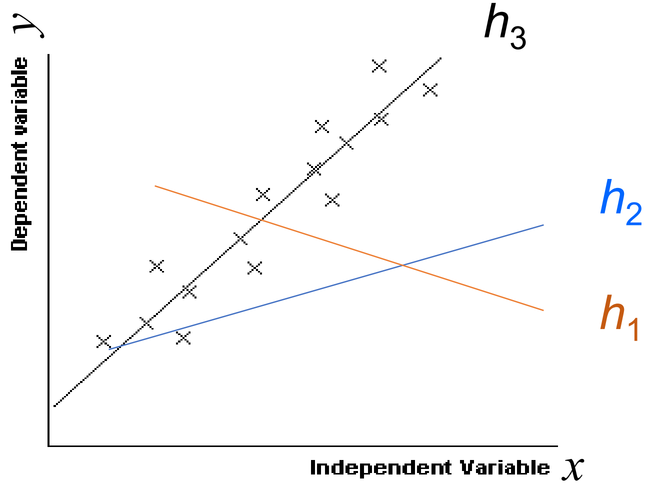
- Error of the training examples

: $E(h)=\frac{1}{2}\frac{1}{N}\sum_{k=1}^{N}(h(\mathbf{x}_k)-y_k)^2$ // Replace *f*(**x**) by y_k

  - $\frac{1}{N}\sum_{k=1}^{N}(h(\mathbf{x}_k)-y_k)^2$: Mean of the squares of $(h(\mathbf{x}_k)-y_k)$, *k* = 1,…, *N*
  - Half the mean (½) as a convenience (1/2是為了之後方便計算) for the later computation of the gradient descent

Minimizing Error Function

- Each h represents a line, $h(x) = w_0 + w_1 x$ if training examples have only one attribute x
  - Minimize $E(h) = E(w_0, w_1) = \frac{1}{2}\frac{1}{N}\sum_{k=1}^{N}(h(x_k) - y_k)^2$



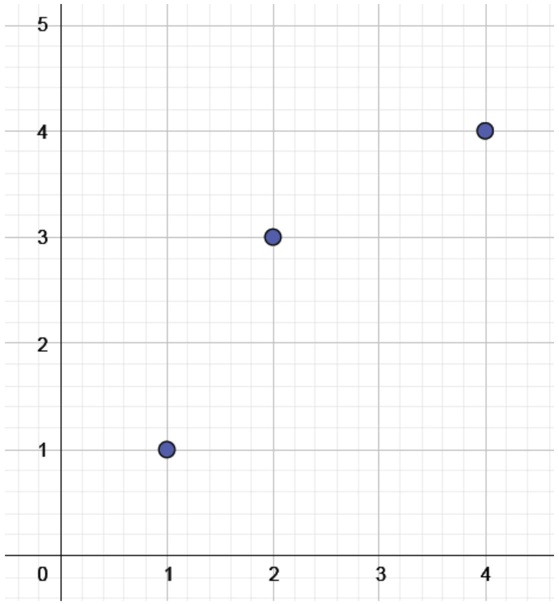
Note: $h(x) = w_0 + w_1 x$
w_0: y-intercept
w_1: the slope of line

Error Function – one weight (1/5)

Assume that $w_0 = 0$ 通過原點

// A line pass through the origin $h(x) = w_0 + w_1 x = w_1 x$

$\Rightarrow h(x) = wx$ // Let $w_1 = w$ for the sake of simplicity(為了簡單)



Training examples (x, y):
(1, 1), (2, 3), (4, 4)

$$E(h) = E(w) = \frac{1}{2}\frac{1}{3}\sum_{k=1}^{3}(h(x_k) - y_k)^2$$

Error Function – one weight (2/5)

Training examples: $(x_1, y_1) = (1, 1)$; $(x_2, y_2) = (2, 3)$; $(x_3, y_3) = (4, 4)$

$E(h) = E(w) = \frac{1}{2}\frac{1}{3}\sum_{k=1}^{3}(h(x_k) - y_k)^2$

If $w = \frac{1}{2}$ then $h1(x_1) = wx_1 = \frac{1}{2} * 1 = \frac{1}{2}$

$h1(x_2) = wx_2 = \frac{1}{2} * 2 = 1$

$h1(x_3) = wx_3 = \frac{1}{2} * 4 = 2$

$\Rightarrow E(w) = \frac{1}{2}\frac{1}{3}\sum_{k=1}^{3}(h1(x_k) - y_k)^2$

$= \frac{1}{2}\frac{1}{3}\{(\frac{1}{2} - 1)^2 + (1 - 3)^2 + (2 - 4)^2\} = \frac{33}{24}$

Error Function – one weight (3/5)

If w = 1 then $h2(x_1)= wx_1$ = 1* 1 = 1

$h2(x_2)= wx_2$ = 1* 2 = 2

$h2(x_3)= wx_3$ = 1* 4 = 4

$\Rightarrow\ E(w) = \frac{1}{2}\frac{1}{3}\sum_{k=1}^{3}(h2(x_k) - y_k)^2$

$= \frac{1}{2}\frac{1}{3}\{(1 - 1)^2 + (2 - 3)^2 + (4 - 4)^2\} = \frac{1}{6}$

If $w = \frac{3}{2}$ then $h3(x_1)= wx_1 = \frac{3}{2} * 1 = \frac{3}{2}$

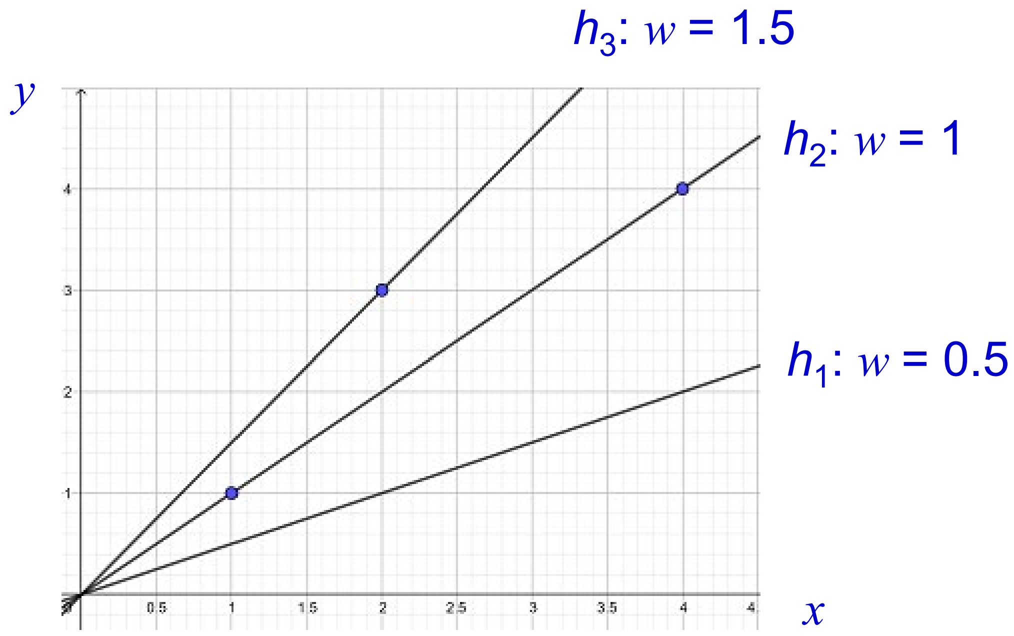
$h3(x_2)= wx_2 = \frac{3}{2} * 2 = 3$

$h3(x_3)= wx_3 = \frac{3}{2} * 4 = 6$

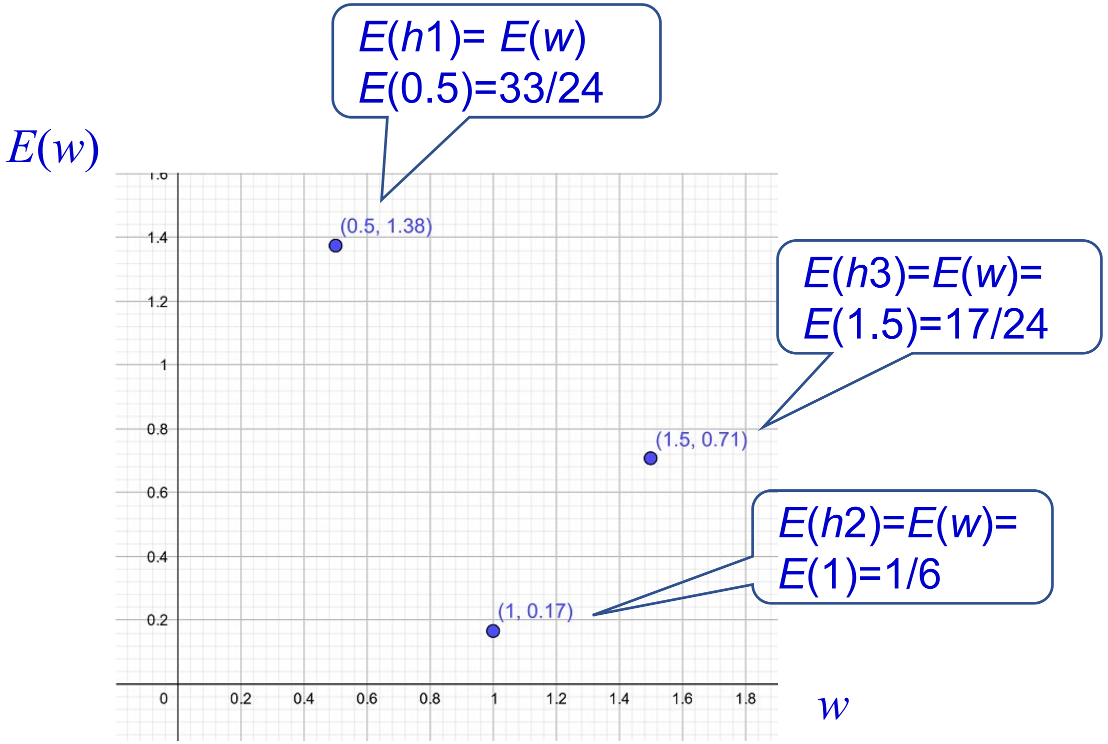
$\Rightarrow\ E(w) = \frac{1}{2}\frac{1}{3}\sum_{k=1}^{3}(h3(x_k) - y_k)^2$

$= \frac{1}{2}\frac{1}{3}\{(\frac{3}{2} - 1)^2 + (3 - 3)^2 + (6 - 4)^2\} = \frac{17}{24}$

Error Function – one weight (4/5)

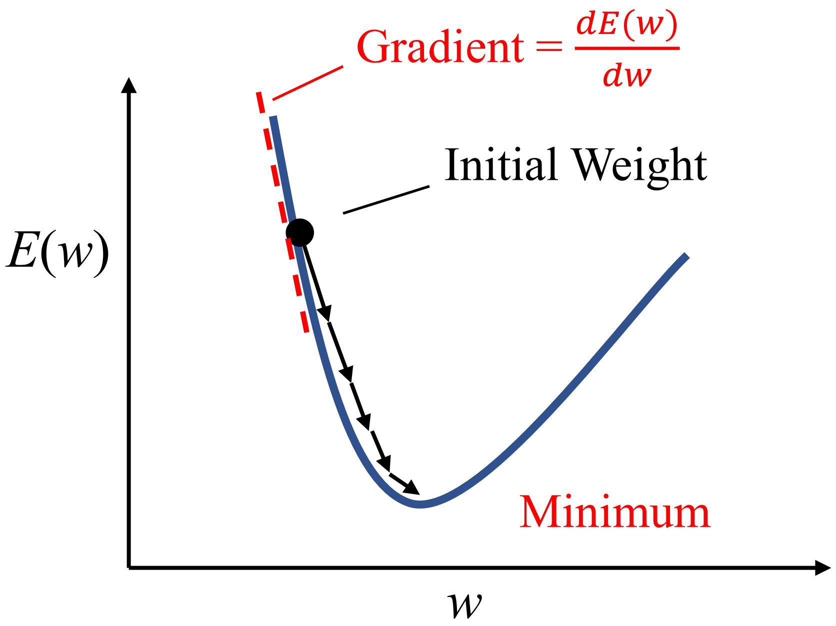


Data points and different hypotheses



Error Function: $E(w)$

Error Function – one weight (5/5)



Note: When w is a scalar, the gradient(梯度) is the tangent line.

Gradient Descent – one weight (1/5)

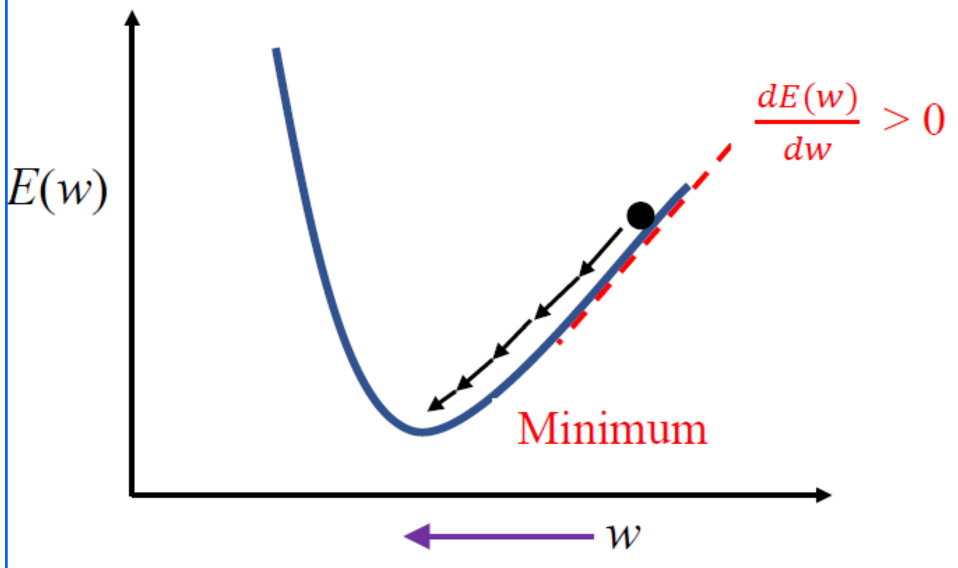
Update the current weight w using the gradient descent $(-\frac{dE(w)}{dw})$ multiplied by some factor called the learning rate, η

Case 1: slope ≥ 0 $(\frac{dE(w)}{dw} \geq 0)$

$$w \leftarrow w - \boxed{\eta \frac{dE(w)}{dw}}$$

Positive

// Decrease w to minimize the error $E(w)$



Decrease w to minimize the error $E(w)$

Gradient Descent – one weight (2/5)

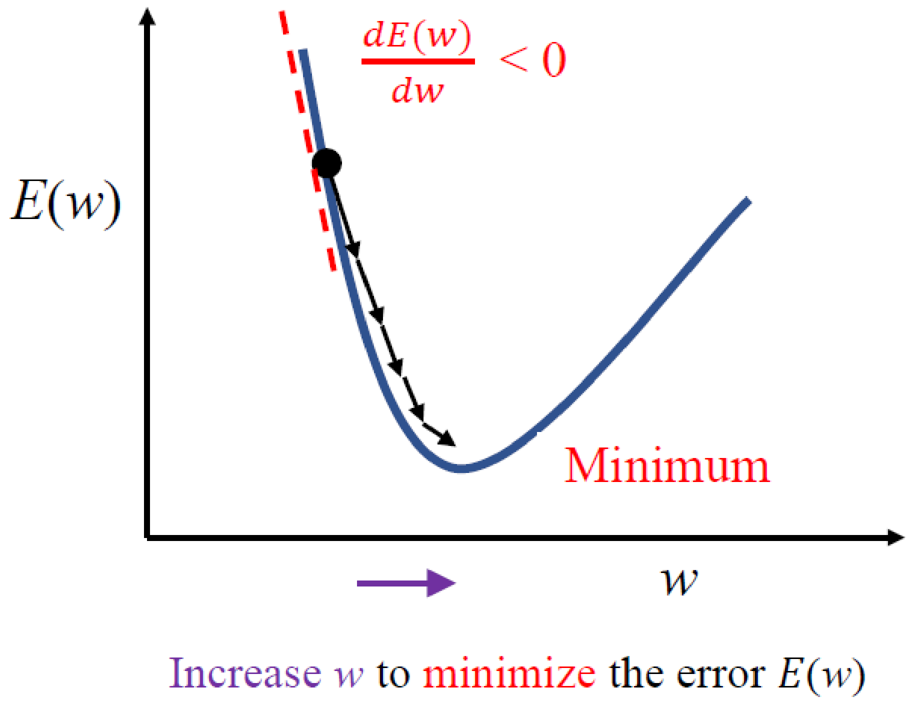
這個error function要往Minimum的方向走的話，
一定都是跟它的微分的負方向有關

Case 2: slope ≤ 0 $\left(\frac{dE(w)}{dw} \leq 0\right)$

$$w \leftarrow w - \boxed{\eta \frac{dE(w)}{dw}}$$

Negative

// Increase w to minimize the error $E(w)$



Gradient Descent – one weight (3/5)

- Training rule for gradient descent:

$$w \leftarrow w + \eta\left(-\frac{dE(w)}{dw}\right)$$

- We make steps down the error function in the direction with the steepest descent(下降最陡的方向).
- The size of each step is determined by the parameter η, which is called the learning rate.
  - If η is too small, gradient descent can be slow.
  - If η is too large(一次走太大步), gradient descent can overshoot the minimum of E. It may fail to converge, or even diverge(發散).

Gradient Descent – one weight (4/5)

- What is $\frac{dE(w)}{dw}$?

  If we assume one single training example only (x, y)
  then $E(h) = E(w) = \frac{1}{2}(h(x) - y)^2 = \frac{1}{2}(wx - y)^2$
  $\Rightarrow$ The gradient of *E* (= the slope of *E*)

$$\frac{dE(w)}{dw} = \frac{d(\frac{1}{2}(wx - y)^2)}{dw} = \frac{1}{2}(2)(wx - y)\frac{d(wx - y)}{dw} = (wx - y)x$$

乘上1/2是為了讓式子變好看

Gradient Descent – one weight (5/5)

- Training rule for gradient descent:

$$w \leftarrow w + \eta\left(-\frac{dE(w)}{dw}\right)$$

$$\Rightarrow \quad w \leftarrow w + \eta\left(-(wx - y)x\right)$$

$$\Rightarrow \quad w \leftarrow w + \eta\,(y - wx)\,x$$

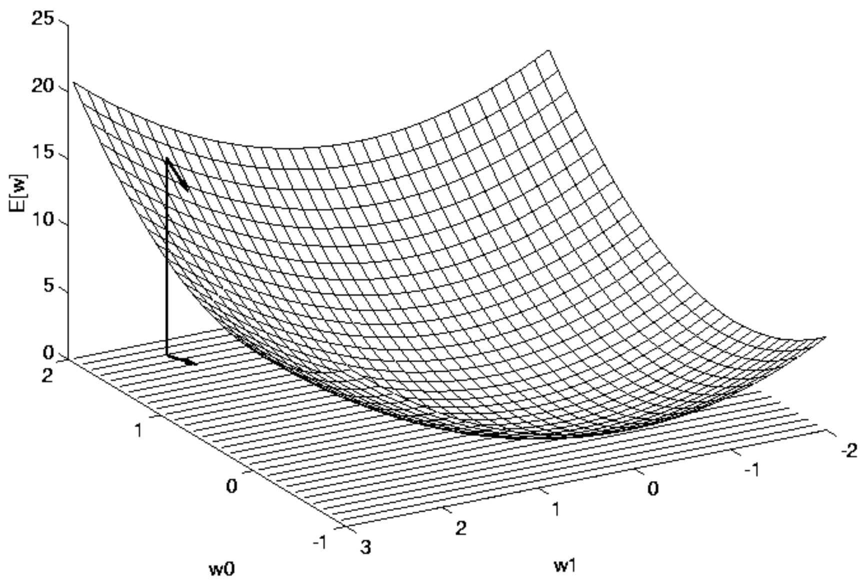
Learning Rate

Input

(Output Label – Estimated Output)

Error Function - two weights

- Example: two weights: $\mathbf{w} = (w_0, w_1)$
  - Arrow: negated gradient at one point
  - Steepest descent along the surface: $-\nabla E(\mathbf{w})$



$$-\nabla E(\mathbf{w}) = -\nabla E(w_0, w_1) = \begin{bmatrix} -\dfrac{\partial E}{\partial w_0} \\ -\dfrac{\partial E}{\partial w_1} \end{bmatrix}$$

Figure adapted from Fig 4.4 in Ref. [3]

Gradient Descent – two weights (1/2)

- If $w_0 \neq 0$ and one single training example only: (x, y), then

$$E(h) = E(w_0, w_1) = \frac{1}{2}(h(x) - y)^2$$

$$= \frac{1}{2}(w_0 x_0 + w_1 x_1 - y)^2 \quad // \; x_0 = 1, \; x_1 = x$$

- The gradient of *E*
- $\nabla E(\mathbf{w}) = \nabla E(w_0, w_1) = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \end{bmatrix} = \begin{bmatrix} (w_0 x_0 + w_1 x_1 - y) x_0 \\ (w_0 x_0 + w_1 x_1 - y) x_1 \end{bmatrix}$

Gradient Descent – two weights (2/2)

- Training rule for gradient descent:

$$w_0 \leftarrow w_0 + \eta\,(y - (w_0 x_0 + w_1 x_1)\,x_0)$$

$$w_1 \leftarrow w_1 + \eta\,(y - (w_0 x_0 + w_1 x_1)\,x_1)$$

- Vector form

Let $\mathbf{w} = \begin{bmatrix} w_0 \\ w_1 \end{bmatrix}$, and $\mathbf{x} = \begin{bmatrix} x_0 \\ x_1 \end{bmatrix}$, then

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\,(y - \mathbf{w}^{\mathrm{T}}\mathbf{x})\mathbf{x}$$

Learning Rate

Input

(Output Label – Estimated Output)

Error Function – (M+1) weights (1/2)

- Now, consider a single example with M attributes:
- $E(h) = \frac{1}{2}(h(\mathbf{x}) - y)^2$

  where $\mathbf{x} = (x_0, x_1, x_2, \ldots, x_M) = \begin{bmatrix} x_0 \equiv 1 \\ x_1 \\ x_2 \\ \vdots \\ x_M \end{bmatrix}$, and

  $h(\mathbf{x}) = w_0\, x_0 + w_1\, x_1 + w_2\, x_2 + \ldots + w_M\, x_M$

  $= \mathbf{w} \cdot \mathbf{x}$ // inner product of two column vectors

  $= \mathbf{w}^T \mathbf{x}$ // row vector $\mathbf{w}^T$ times column vector $\mathbf{x}$

Error Function– (M+1) weights (2/2)

- $\mathbf{w} = (w_0, w_1, w_2, w_3, \ldots, w_M)$ is to be determined and to be fit in the learning problem.
- The space H of candidate hypothesis is the set of all possible real-valued weight vectors:

  $H = \{\mathbf{w} | \mathbf{w} \in R^{M+1}\}$

Gradient Descent – (M+1) weights (1/2)

- Gradient of E with respect to $\mathbf{w} = (w_0, w_1, \ldots, w_M)$:

$$\nabla E(\mathbf{w}) = \nabla E(w_0, w_1, \ldots, w_M) = \begin{bmatrix} \frac{\partial E}{\partial w_0} \\ \frac{\partial E}{\partial w_1} \\ \vdots \\ \frac{\partial E}{\partial w_M} \end{bmatrix} = \begin{bmatrix} (w_0 x_0 + w_1 x_1 + \cdots + w_M x_M - y) x_0 \\ (w_0 x_0 + w_1 x_1 + \cdots + w_M x_M - y) x_1 \\ \vdots \\ (w_0 x_0 + w_1 x_1 + \cdots + w_M x_M - y) x_M \end{bmatrix}$$

$$= \begin{bmatrix} (h(\mathbf{x}) - y) x_0 \\ (h(\mathbf{x}) - y) x_1 \\ \vdots \\ (h(\mathbf{x}) - y) x_M \end{bmatrix} = (h(\mathbf{x}) - y) \begin{bmatrix} x_0 \\ x_1 \\ \vdots \\ x_M \end{bmatrix} = (\mathbf{w}^T \mathbf{x} - y)\, \mathbf{x}$$

Gradient Descent – (M+1) weights (2/2)

- Training rule for gradient descent:

$$\mathbf{w} \leftarrow \mathbf{w} + \eta\,(-\nabla E(\mathbf{w}))$$

$$\Rightarrow \quad \mathbf{w} \leftarrow \mathbf{w} + \eta\,(-(\mathbf{w}^{\mathsf{T}}\mathbf{x} - y)\mathbf{x})$$

$$\Rightarrow \quad \mathbf{w} \leftarrow \mathbf{w} + \eta\,(y - \mathbf{w}^{\mathsf{T}}\mathbf{x})\mathbf{x}$$ // Vector form

- We may write the rule in its component form

$$w_i \leftarrow w_i + \eta\,(y - \mathbf{w}^{\mathsf{T}}\mathbf{x})x_i$$

// Component form; i = 0, 1, …, M

The Gradient Descent Algorithm

Given training data set:

D = $\{(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_N, y_N)\}$ // N training examples

Initialize all weights w_i to random values

// $\mathbf{w} = (w_0, w_1, \ldots, w_M)$; M+1 attributes

UNTIL the termination condition is met, DO

FOR each training example $\mathbf{x}_k \in D$

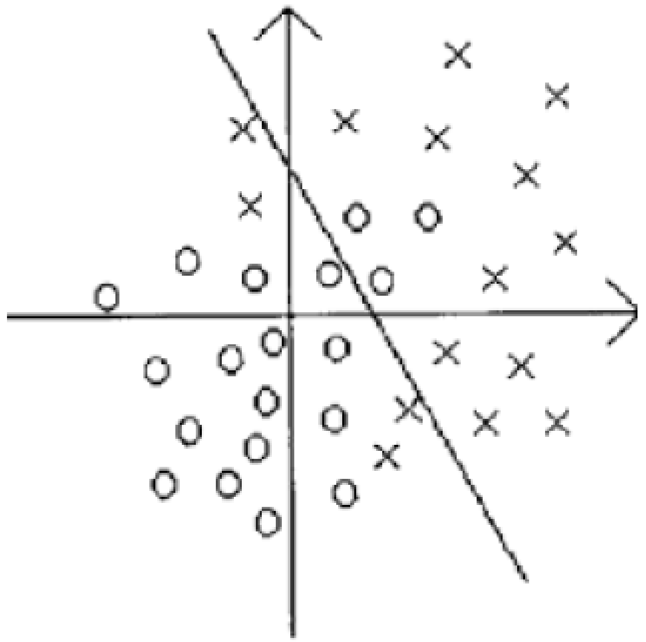
$$\mathbf{w} \leftarrow \mathbf{w} + \eta\,(y_k - \mathbf{w}^T\mathbf{x}_k)\mathbf{x}_k$$

Remarks on the Gradient Descent Algorithm (1/2)

- The gradient descent algorithm is also known as 又稱為
  - Delta rule
  - ADALINE rule
  - Widrow-Hoff rule
- The gradient descent algorithm is used in regression(迴歸) problems whose training data have an infinite(無限) number of outputs.
  - Regression: with numeric output label(s) $\Rightarrow$ an infinite number of values
- However, it can also be used in classification(分類) problems whose training data have a finite(有限) number of outputs.
  - Classification: with nominal output label(s) $\Rightarrow$ a finite number of values

Remarks on the Gradient Descent Algorithm (2/2)

- When we use the gradient descent algorithm to solve a classification problem, it converges toward a best-fit(最適合的) approximation to the target concept If the training examples are not linearly separable(不可線性分離).



Summary

- Linear Classification(分類)
  - Perceptron Learning Algorithm
  - If sign ($\mathbf{w} \bullet \mathbf{x}$) $\neq y$ then $\mathbf{w} \leftarrow \mathbf{w} + y\,\mathbf{x}$
    - **w**: Normal vector of the decision boundary
    - (**x**, y): Training example with nominal output y (= $\pm$1)
- Linear Regression(迴歸)
  - The Gradient Descent Algorithm
  - $\mathbf{w} \leftarrow \mathbf{w} + \eta\,(y - \mathbf{w}^T\mathbf{x})\mathbf{x}$
    - **w**: Normal vector of the linear function
    - (**x**, y): Training example with numeric output y
    - η : Learning rate

Kahoot

(T)1. A one-neuron perceptron can classify two classes of objects.

(F) 2. The average squared difference between classifier predicted output and actual output is the Mean Absolute Error.

(T) 3. The goal of the gradient descent is to minimize a given loss function of the neural network.

(F) 4. Gradient descent always converges to some global minimum.

(T) 5. The “learning rate” is the step size that is the amount the weights are updated during training.

(F) 6. In batch gradient descent, we consider one example at a time to update the current weight and bias values.

(F) 7. Logistic regression is mainly used for regression. (It is popularly used for classification.)

(T) 8. The cost function for logistic regression is the Cross-Entropy.

(T) 9. Logistic regression predicts the probability of the occurrence of a binary outcome.

(F) 10. Logistic regression transforms its output using the tanh function to return a probability value.(tanh function(+1 ~ -1)只能視為是Logistic regression的變形

References

[1] Artificial Intelligence: A Modern Approach, Third edition, Stuart Russell and Peter Norvig, Prentice Hall Series in Artificial Intelligence, 2010.

[2] Learning from Data, Yaser S. Abu-Mostafa, Malik Magdon-Ismail, and Hsuan-Tien Lin, AMLBook, 2012.

[3] Machine Learning, Tom M. Mitchell, McGraw-Hill, 1997.