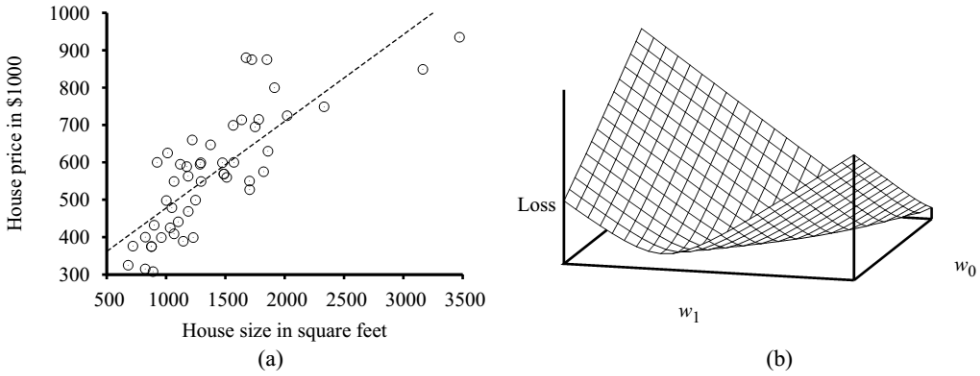

Inteligencia Artificial

**Regresión y clasificación con modelos lineales.**

El espacio de hipótesis está formado por **funciones lineales** de entradas con **valores continuos**.

# Regresión lineal univariada.

- Tiene la forma de una recta: $y = w1\ x + w0$
- usamos *w* porque vamos a pensar los coeficientes como pesos (weights)
- **w** es el vector definido por [w0, w1]
- hw(x) = w1 x + w0




- Se conoce como regresión a la tarea de encontrar un h que minimice la pérdida empírica, tradicionalmente se usa L2.

Método válido para **formas cerradas**

$$Loss(h_{\mathbf{w}}) = \sum_{j=1}^{N} L_2(y_j, h_{\mathbf{w}}(x_j)) = \sum_{j=1}^{N} (y_j - h_{\mathbf{w}}(x_j))^2 = \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 .$$

$$\frac{\partial}{\partial w_0} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 \text{ and } \frac{\partial}{\partial w_1} \sum_{j=1}^{N} (y_j - (w_1 x_j + w_0))^2 = 0 .$$

$$w_1 = \frac{N(\sum x_j y_j) - (\sum x_j)(\sum y_j)}{N(\sum x_j^2) - (\sum x_j)^2}; \quad w_0 = (\sum y_j - w_1(\sum x_j))/N .$$

Método hill-climbing con descenso por el gradiente

$\mathbf{w} \leftarrow$ any point in the parameter space

**loop** until convergence **do**

  **for each** $w_i$ **in** w **do**

$$w_i \leftarrow w_i - \alpha \frac{\partial}{\partial w_i} Loss(\mathbf{w})$$

- alfa: learning rate. Puede ser constante o variable reduciendose a medida que transcurre el aprendizaje.

**Para 1 ejemplo**

$$\frac{\partial}{\partial w_i} Loss(\mathbf{w}) = \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))^2$$
$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - h_{\mathbf{w}}(x))$$
$$= 2(y - h_{\mathbf{w}}(x)) \times \frac{\partial}{\partial w_i}(y - (w_1 x + w_0)) \,,$$

$$\frac{\partial}{\partial w_0} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \,; \qquad \frac{\partial}{\partial w_1} Loss(\mathbf{w}) = -2(y - h_{\mathbf{w}}(x)) \times x$$

$$w_0 \leftarrow w_0 + \alpha\,(y - h_{\mathbf{w}}(x)) \,; \quad w_1 \leftarrow w_1 + \alpha\,(y - h_{\mathbf{w}}(x)) \times x$$

**Para N ejemplos**

$$w_0 \leftarrow w_0 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \,; \quad w_1 \leftarrow w_1 + \alpha \sum_j (y_j - h_{\mathbf{w}}(x_j)) \times x_j \,.$$

- Descenso por el gradiente en lote (batch): utiliza las fórmulas de actualizacion que contemplan todos los ejemplos
- Descenso por el gradiente estocástico: se aplica la regla de actualización individual eligiendo un ejemplo al azar.

# Regresión lineal multivariada.

Es la misma idea pero **x** es un vector

$$h_{sw}(\mathbf{x}_j) = w_0 + w_1 x_{j,1} + \cdots + w_n x_{j,n} = w_0 + \sum_i w_i x_{j,i} \,.$$

Si creamos una entrada *xj, 0* que siempre valga 1 podemos expresar **h** como el producto punto o producto de matrices.

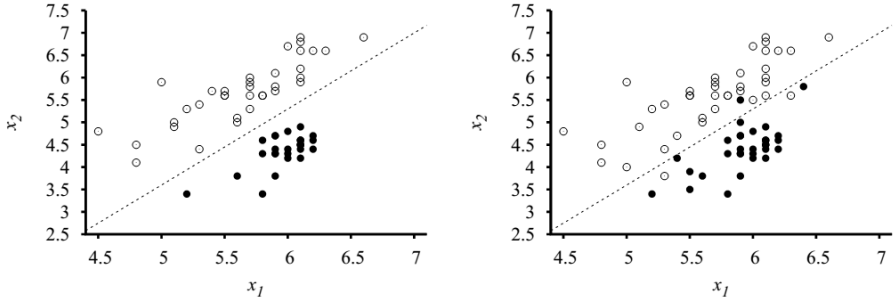
$$h_{sw}(\mathbf{x}_j) = \mathbf{w} \cdot \mathbf{x}_j = \mathbf{w}^\top \mathbf{x}_j = \sum_i w_i x_{j,i} \,.$$

La regla de actualización es:

$$w_i \leftarrow w_i + \alpha \sum_j x_{j,i}(y_j - h_{\mathbf{w}}(\mathbf{x}_j)) \,.$$

Con regresión multivariada es común usar regularización para evitar **overfitting**

# Clasificadores lineales con umbral




- Una **frontera de decisión** es una línea que separa 2 clases.
- Una frontera lineal se llama **separador lineal** y los datos que los admiten **linealmente separables**

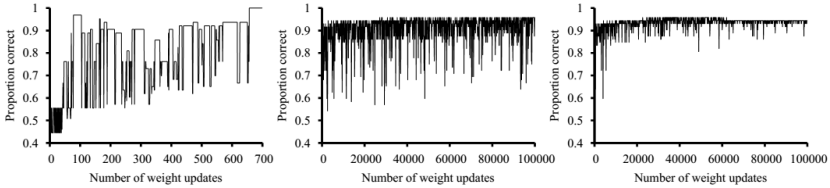
$$x_2 = 1.7x_1 - 4.9 \quad \text{or} \quad -4.9 + 1.7x_1 - x_2 = 0 \,.$$

$$h_{\mathbf{w}}(\mathbf{x}) = 1 \text{ if } \mathbf{w} \cdot \mathbf{x} \geq 0 \text{ and } 0 \text{ otherwise.}$$

$$h_{\mathbf{w}}(\mathbf{x}) = \mathit{Threshold}(\mathbf{w} \cdot \mathbf{x}) \text{ where } \mathit{Threshold}(z) = 1 \text{ if } z \geq 0 \text{ and } 0 \text{ otherwise.}$$

- La función **threshold** no es derivable, por lo tanto no se puede hacer descenso por el gradiente
- En su lugar podemos usar la **regla de aprendizaje del perceptrón**

$$w_i \leftarrow w_i + \alpha \left(y - h_{\mathbf{w}}(\mathbf{x})\right) \times x_i$$



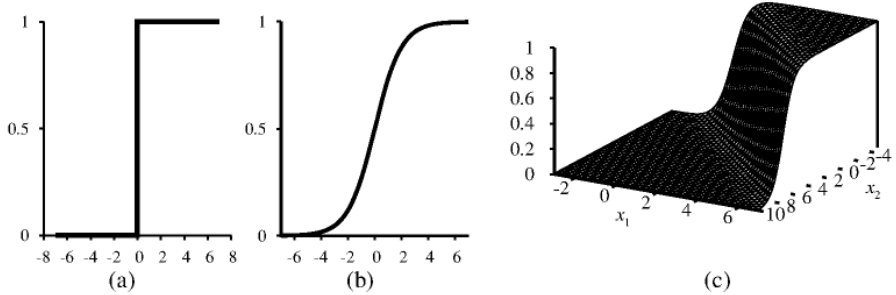



**Figure 18.17** (a) The hard threshold function $Threshold(z)$ with 0/1 output. Note that the function is nondifferentiable at $z = 0$. (b) The logistic function, $Logistic(z) = \frac{1}{1+e^{-z}}$, also known as the sigmoid function. (c) Plot of a logistic regression hypothesis $h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x})$ for the data shown in Figure 18.15(b).

# Clasificadores lineales con regresión logística.

La función **threshold** tiene algunos inconvenientes:

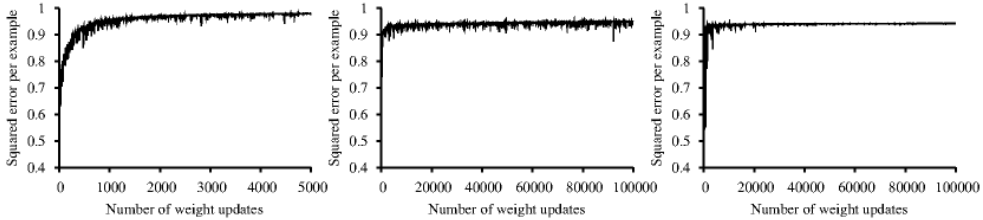
- No es derivable, lo cual hace que el aprendizaje sea poco predecible
- Valores muy cercanos a la frontera son clasificados con 0 o 1

La función **logística** solventa estos problemas.

$$h_{\mathbf{w}}(\mathbf{x}) = Logistic(\mathbf{w} \cdot \mathbf{x}) = \frac{1}{1 + e^{-\mathbf{w} \cdot \mathbf{x}}} .$$

La regla de actualización es:

$$w_i \leftarrow w_i + \alpha \left(y - h_{\mathbf{w}}(\mathbf{x})\right) \times h_{\mathbf{w}}(\mathbf{x})(1 - h_{\mathbf{w}}(\mathbf{x})) \times x_i$$


Squared error per example
1
0.9
0.8
0.7
0.6
0.5
0.4
0
1000
2000
3000
4000
5000
Number of weight updates
(a)
Squared error per example
0
20000
40000
60000
80000
100000
Number of weight updates
(b)
Squared error per example
0
20000
40000
60000
80000
100000
Number of weight updates
(c)

**Bibliografía y enlaces útiles.**


- Russell S., Norvig P.: Artificial Intelligence: A modern Approach. Third Edition. Chapter 18.