Redes Neuronales (1)

Carlos Hurtado L.

Depto. de Ciencias de la Computación, Universidad de Chile

Referencias

- Machine Learning, Tom Mitchell, Capítulo 4
- Apuntes del capítulo 4 de este libro:

  http://www.cs.cmu.edu/~tom/mlbook-chapter-slides.html

Redes Neuronales (Artificiales)

- Método general y práctico para aprendizaje supervisado.
- Modelan relaciones complejas entre datos de entrada y salida.
- Modelo (función) de regresión y clasificación.

Redes Neuronales

- Inspiración en sistemas neuronales biológicos: modelos conexionistas
- Red de numerosas unidades de procesamiento simples: “neuronas”.
- Cada neurona es una función que recibe valores de otras neuronas o desde las entradas del sistema.

Modelo Conexionista

Cerebro Humano $\sim 10^{10}$

- Número de neuronas $\sim 10^{4-5}$
- Conexiones por neurona
- Tiempo de reacción por neuro $\sim .001\ \text{second}$
- Tiempo que toma reconocer

una cara $\sim .1\ \text{second}$

- Si esta tarea se realizara en forma secuencial, se haría en 100 pasos, lo cual es imposible
- Por el contrario, se realiza con un alto nivel de procesamiento paralelo

Modelo Conexionista (cont.)

Redes Neuronales:

- Muchas unidades simple de procesamiento: neuronas
- Muchas conexiones entre neuronas
- Parámetros son pesos de conexiones
- Potencialidad para ejecutarse con un alto grado de paralelismo.
- Aprendizaje: ajuste continuo de pesos en base a datos de entrenamiento

Historia

- 1943 McCullocs y Pitts. “A Logical Calculus of the Ideas immanent in Nervous Activity”
  - Tesis de construcción de redes neuronales sólo usando matemática y algoritmos.
- 1962 Rosenblatt. Primer algoritmo de entrenamiento
- 1969 Minsky y Papert. Serie de papers seminales. Mustran limitaciones como tiempo exponencial de entrenamiento y necsidad de tener más de un nivel.

Historia

- 1974 Werbos. Algoritmo de Retropropagación: método eficiente de entrenamiento donde el efecto de las entradas se propaga por la red ajustando pesos.

Aplicaciones Comunes

- Reconocimiento de fonemas en señales de voz
- Reconocimiento de caracteres desde escritura manual
- Clasificación de imágenes
- Predicción financiera

Ejemplo: Sistema ALVINN (Pomerleau's 1993)

- Red que conduce un automóvil en carretera pública con tráfico a 70 millas/hora en trayectos de 90 millas
- Opera procesando imágenes de una cámara decidiendo movimientos del volante
- La red es entrenada usando la conducción de un piloto humano durante 5 minutos

Sistema ALVINN (Pomerleau's 1993)

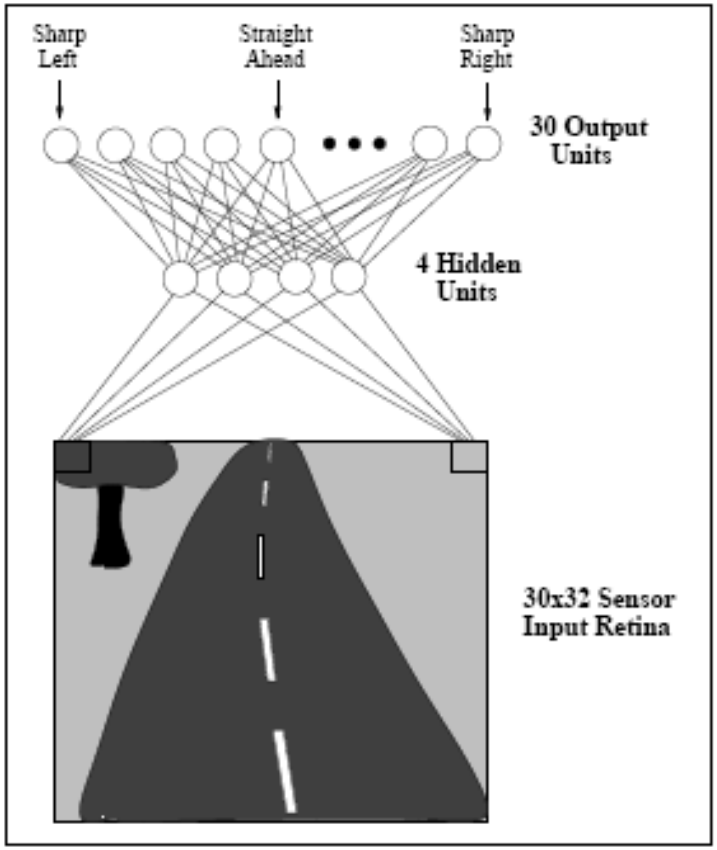
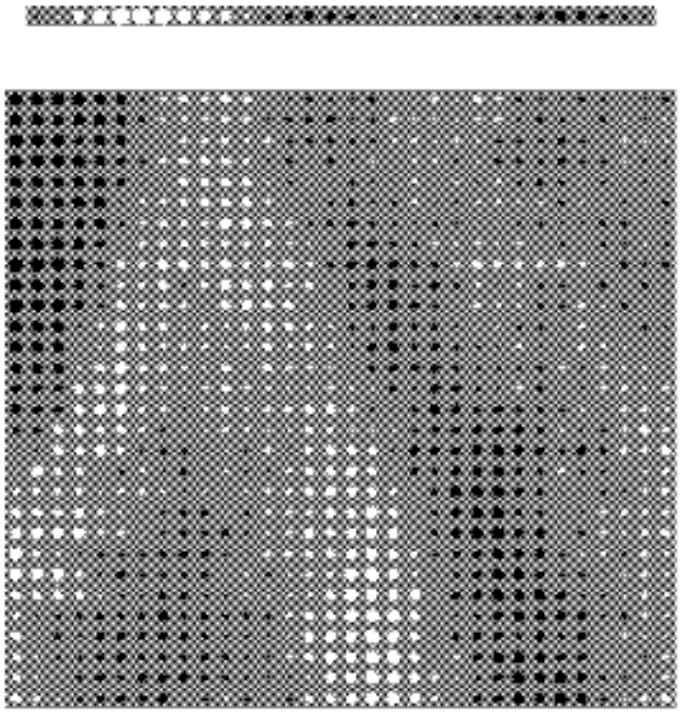


Imagen de 30 x 32 pixeles

Peso de unidad externa

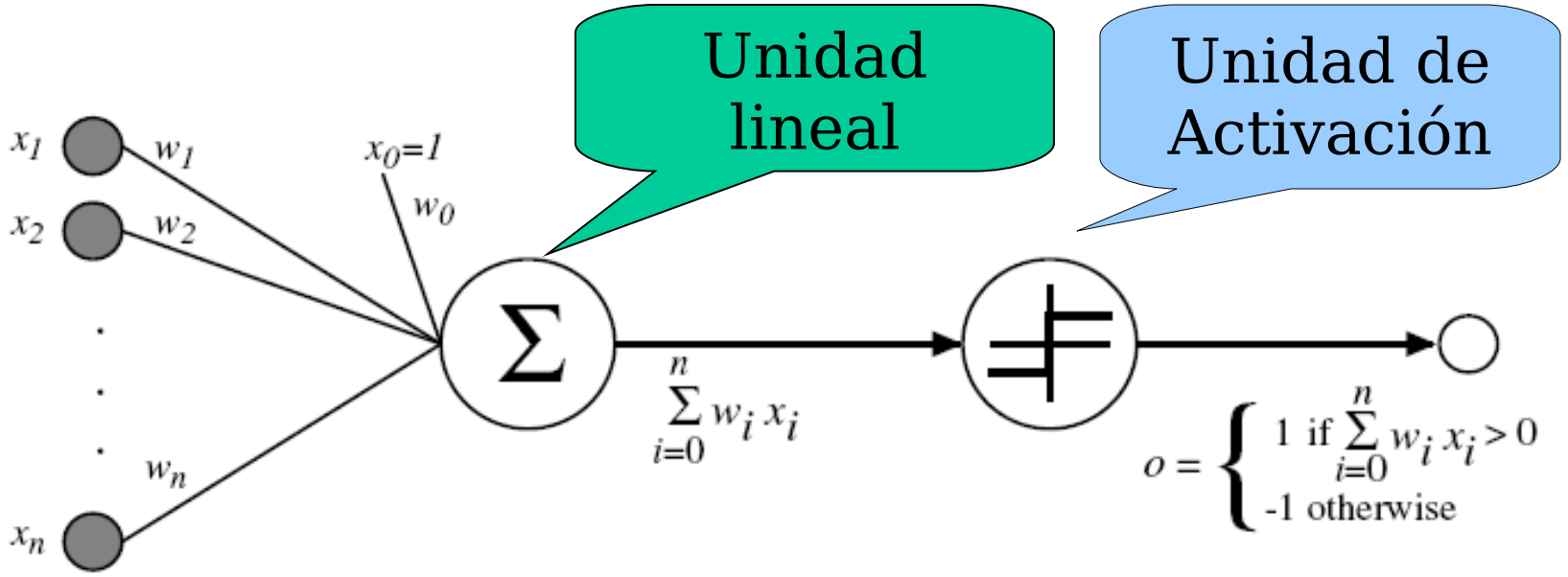


Pesos unida interna: blanco = positivo, negro = negativo.

Casos donde Conviene usar Redes Neuronales

- Aprendizaje de una función f:X→Y
  - X: vector de muchas variables numéricas
  - Y: variable numérica, discreta o vector
  - Forma f no se conoce (por ejemplo, no se puede aplicar regresión lineal u otra forma)
- Datos de entrada son ruidosos y complejos, típicamente provienen de sensores como cámaras y micrófonos.
- No es importante la “interpretabilidad” del modelo.
- Se tiene tiempo y capacidad computacional suficiente para entrenar
- Se requiere aplicar el modelo en forma eficiente

Perceptrón



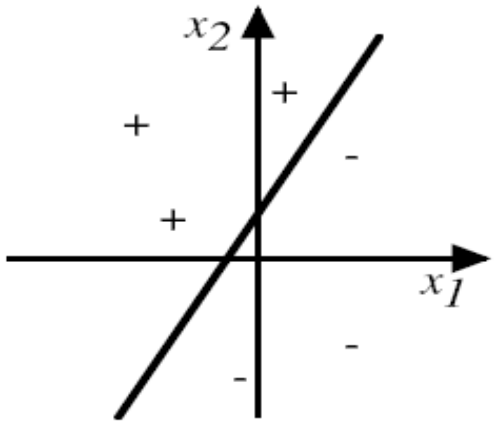
$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Perceptrón: Notación Vectorial

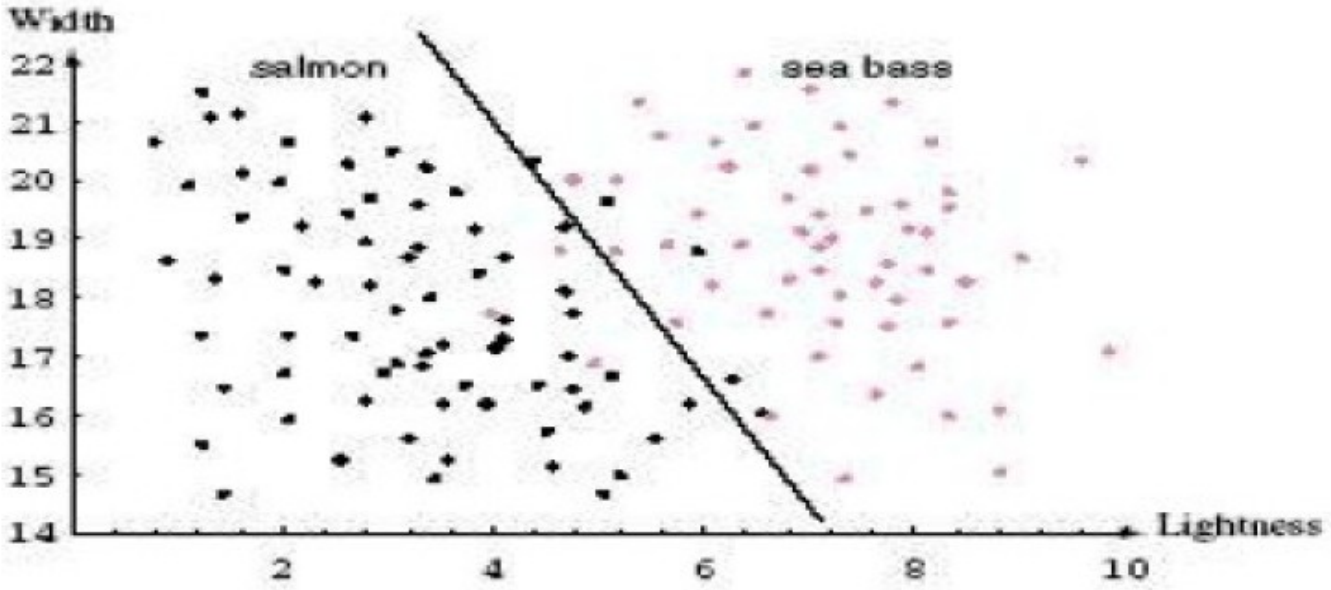
$$o(\vec{x}) = \begin{cases} 1 & \text{if } \vec{w} \cdot \vec{x} > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Perceptrón como Modelo de Clasificación

- Un Perceptrón es un modelo que define un límite de decisión para las clases 1 y -1.
- Para un Perceptrón este límite siempre es lineal (hiperplano)



Perceptrón para Salmón vs. Corvina



Perceptrón y Funciones Booleanas

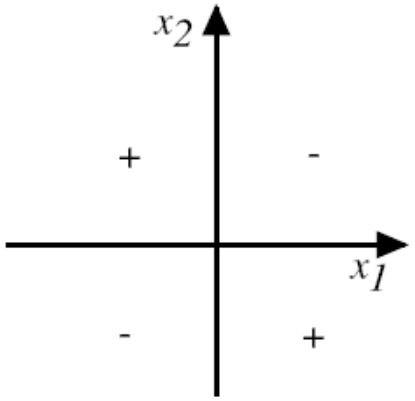
- Puede representar todas las funciones Booleanas primitivas: AND,NAND, OR, NOR
- Por ejemplo, para AND basta w_0=-0,8 y w_1=w_2=0,5

Poder expresivo de un Perceptrón

- Puede representar cualquier límite de decisión lineal
- La combinación de varios Perceptrones en un red neuronal permite modelar límites de decisión con forma de poliedros.

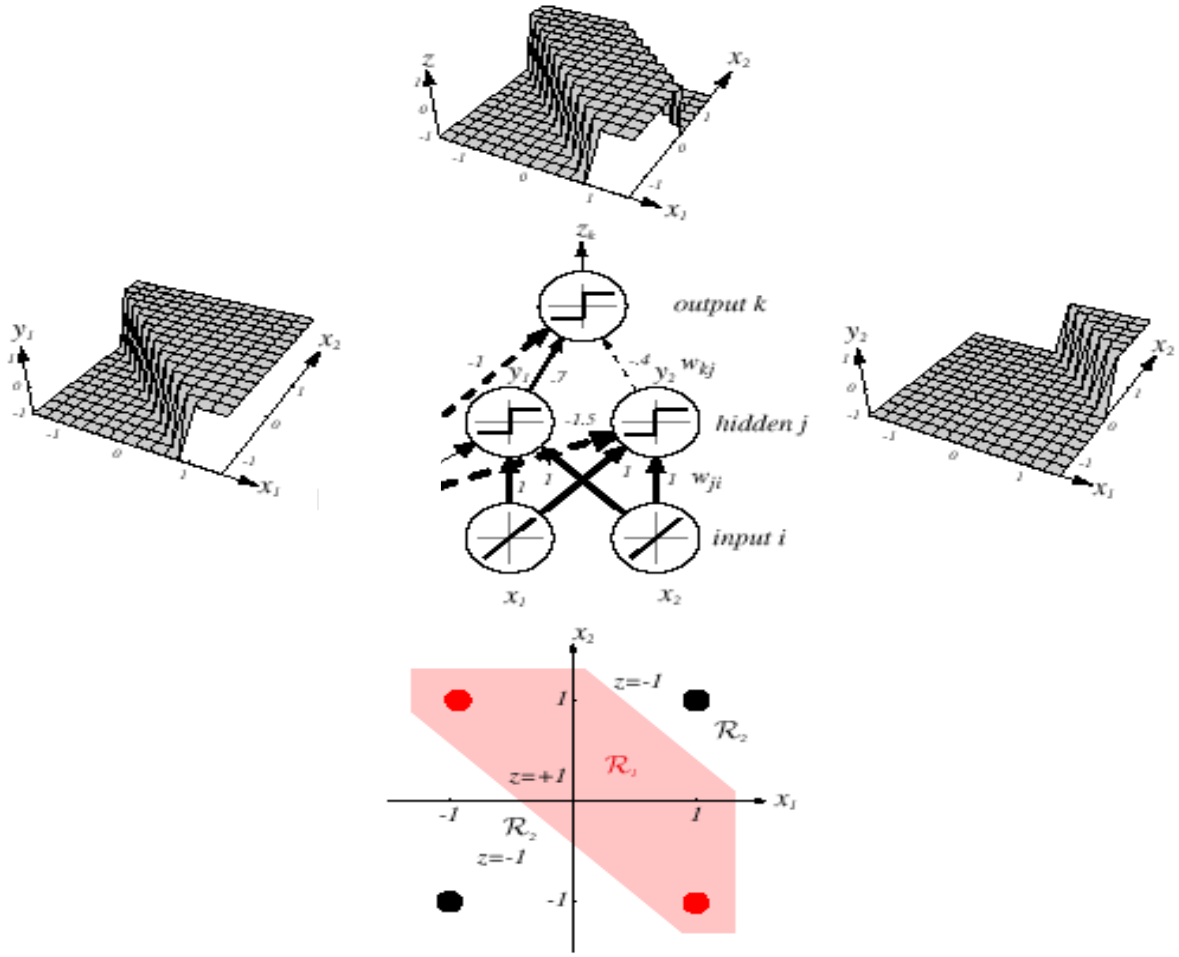
Poder expresivo de un Perceptrón

- Por ejemplo, un perceptrón no puede representar XOR:



- Una combinación de Perceptrones permite modelar cualquier función Booleana.

Modelación de XOR usando perceptrones



Ejercicio

- Construir una red de Perceptrones que modele la función Booleana XOR

Entrenamiento de un Perceptrón

- Cada dato de entrenamiento es un par de la forma $\langle \vec{x}, t \rangle$ donde $\vec{x}$ es un vector de valores de input y t es el valor de la clase en {1,-1}.
- El problema de entrenamiento consiste en estimar los pesos.

Regla de Entrenamiento de un Perceptrón

$$w_i \leftarrow w_i + \Delta w_i$$

donde $\Delta w_i = \eta(t - o)x_i$

$t = c(\vec{x})$ es el output de un dato de entrenamiento

o es el output del Perceptrón

η es una constante (e.g., 0.1) llamada tasa de aprendizaje

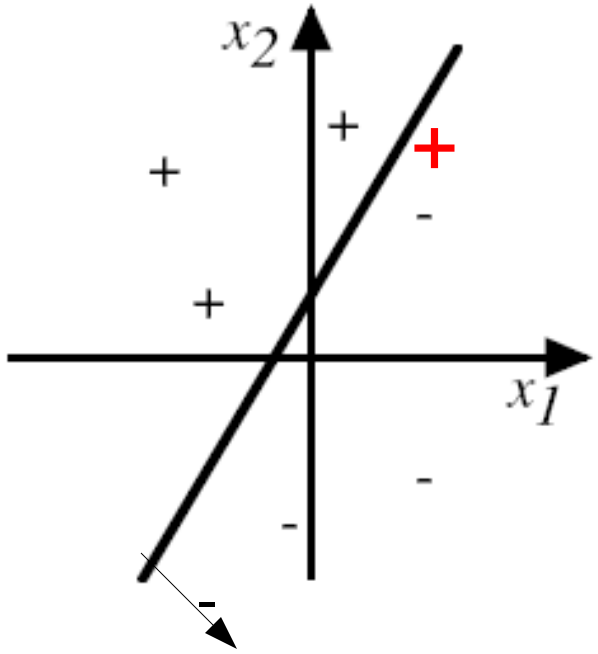
Entrenamiento de un Perceptron

- Aplicar regla de entrenamiento iterativamente hasta que ningún dato este mal clasificado
- Este procedimiento converge si:
  - Las clases son linealmente separables en los datos de entrenamiento
  - La tasa de aprendizaje es suficientemente pequeña

¿Por qué funciona?

- Hay tres casos:
  - *(A) t-o = 0 (no hay error)*
  - *(B) t-o = 2*
  - *(C) t-o =-2*
- *Ejemplo, caso (B):*

$$\Delta w_i = \eta(t - o)x_i$$



¿Por qué funciona?

- Si *t-o =2, tenemos que actualizar los pesos de modo que ahora*

$$\vec{w} \cdot \vec{x} > 0$$

- *Luego debemos aumentar wi si xi es positivo y disminuirlo si xi es negativo*
- *Esto sucede ya que* $\Delta w_i = \eta(t-o)x_i$ *tiene el mismo signo que xi.*
- *Si* η *es pequeño, no hay oscilación*

Problema de Regla de Entrenamiento

- No converge si límite de decisión no es lineal.
- En este caso tenemos error por sesgo.
- Necesitamos encontrar el Perceptrón que minimice el error.
- Esto requiere definir noción de error para un Perceptrón.

Método de Descenso por Gradiente

- Consideramos sólo la unidad lineal del Perceptrón (equivalente un modelo de regresión lineal):

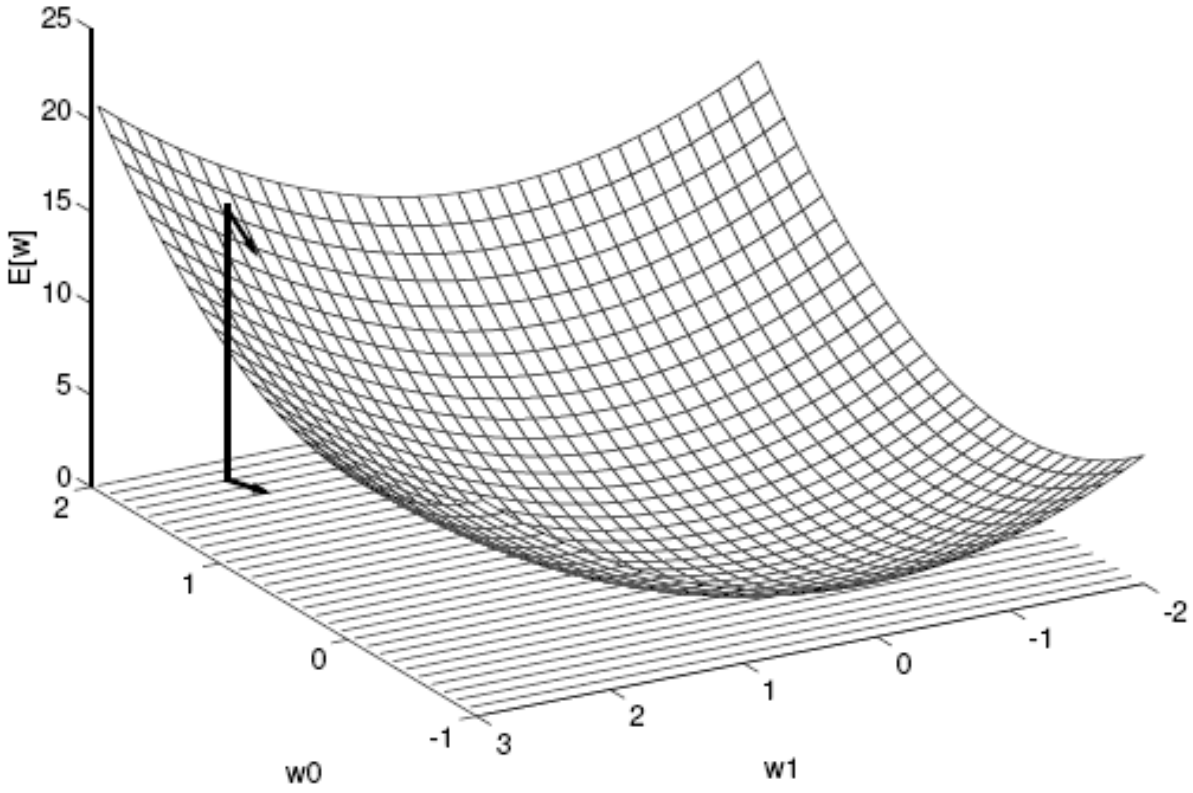
$$o = w_0 + w_1 x_1 + \cdots + w_n x_n$$

- Un buen modelo debe minimizar el error cuadrado

$$E[\vec{w}] \equiv \frac{1}{2} \sum_{d \in D} (t_d - o_d)^2$$

- Donde D es el conjunto de entrenamiento, t_d es la clase del dato y o_d es el valor de la función lineal

Espacio de Pesos Posibles



Descenso por el Gradiente

Gradiente:

$$\nabla E[\vec{w}] \equiv \left[\frac{\partial E}{\partial w_0}, \frac{\partial E}{\partial w_1}, \cdots \frac{\partial E}{\partial w_n}\right]$$

Regla de Entrenamiento:

$$\Delta \vec{w} = -\eta \nabla E[\vec{w}]$$

$$\Delta w_i = -\eta \frac{\partial E}{\partial w_i}$$

Gradiente del Error Cuadrado

$$
\begin{aligned}
\frac{\partial E}{\partial w_i} &= \frac{\partial}{\partial w_i}\frac{1}{2}\sum_d (t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d \frac{\partial}{\partial w_i}(t_d - o_d)^2 \\
&= \frac{1}{2}\sum_d 2(t_d - o_d)\frac{\partial}{\partial w_i}(t_d - o_d) \\
&= \sum_d (t_d - o_d)\frac{\partial}{\partial w_i}(t_d - \vec{w}\cdot\vec{x_d}) \\
\frac{\partial E}{\partial w_i} &= \sum_d (t_d - o_d)(-x_{i,d})
\end{aligned}
$$

Algoritmo de Descenso por Gradiente

- Initialize each w_i to some small random value
- Until the termination condition is met, Do
  - Initialize each Δw_i to zero.
  - For each $\langle \vec{x}, t \rangle$ in $training\_examples$, Do
    - Input the instance $\vec{x}$ to the unit and compute the output o
    - For each linear unit weight w_i, Do

$$\Delta w_i \leftarrow \Delta w_i + \eta(t - o)x_i$$

  - For each linear unit weight w_i, Do

$$w_i \leftarrow w_i + \Delta w_i$$

Recapitulando

- Regla de entrenamiento de Perceptrón
  - Converge a limite de decisión exacto dado:
    - Datos de entrenamiento son separables linealmente y
    - Tasa de aprendizaje pequeña
- Entrenamiento de Perceptrón usando descenso por gradiente:
  - Converge a error cuadrado mínimo dado:
    - Tasa de aprendizaje pequeña
    - No requiere que datos de entrenamiento sean separables linealmente