Clase 4:
Métodos Supervisados

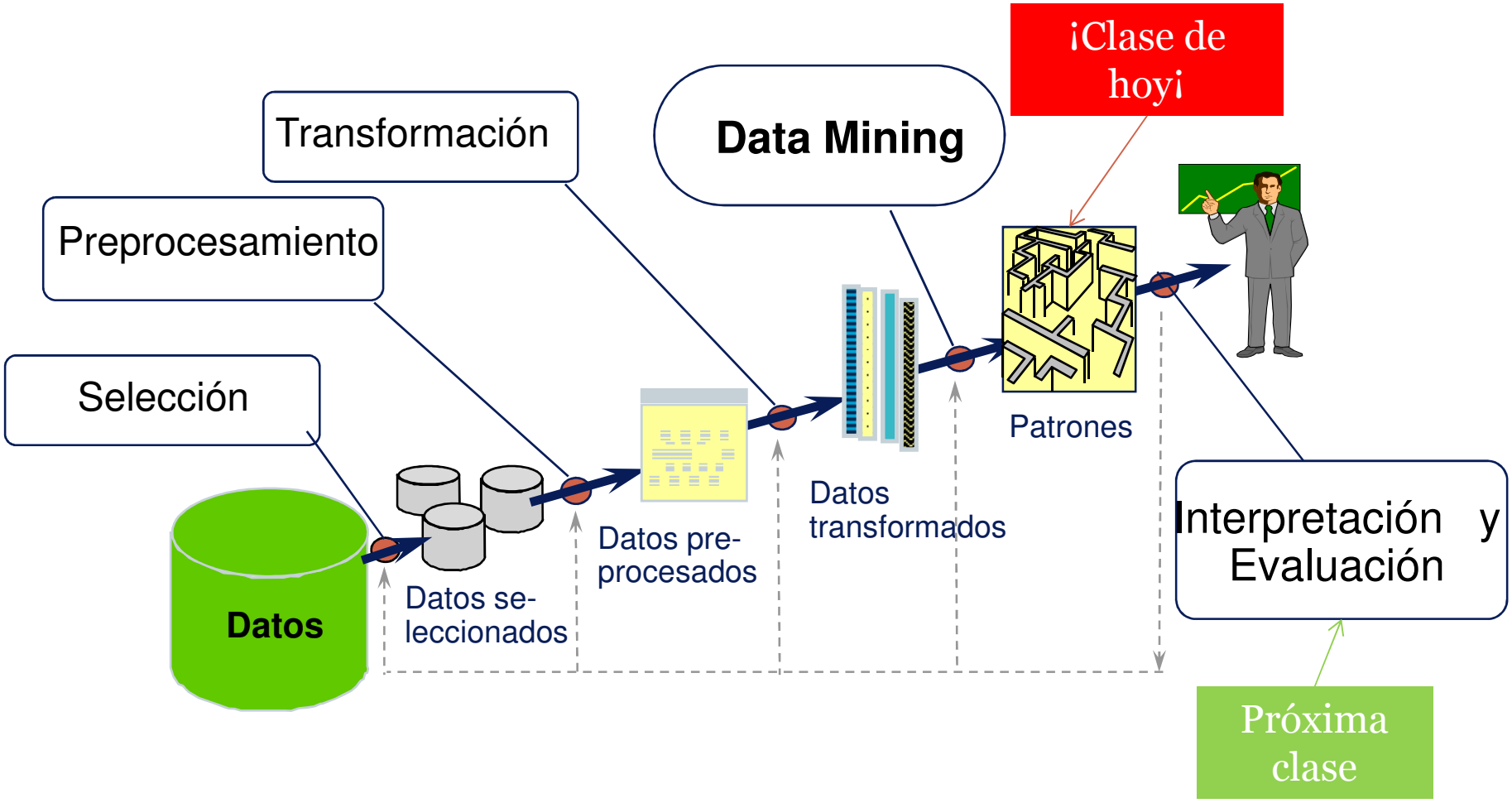
SEBASTIÁN MALDONADO

DIPLOMADO *BUSINESS INTELLIGENCE*

27 DE DICIEMBRE, 2011

DIAPOSITIVAS: CRISTIÁN BRAVO,
SEBASTIÁN MALDONADO

Proceso KDD


¡Clase de hoy¡
Transformación
Data Mining
Preprocesamiento
Selección
Patrones
Datos
Datos se-leccionados
Datos pre-procesados
Datos transformados
Interpretación y Evaluación
Próxima clase

Tipos de Aprendizaje

- Aprendizaje supervisado
  - **Se utiliza el conocimiento *a-priori* del comportamiento de un conjunto de observaciones: Conjunto de Entrenamiento**
  - Ejemplos: Regresión Logística, **Arboles de Decisión**, **Redes Neuronales**, **Redes Bayesianas**, **Naïve Bayes**, **Support Vector Machines**, etc…
- Aprendizaje no-supervisado
  - No se utiliza conocimiento *a-priori* del comportamiento de un conjunto de observaciones.
  - Ejemplos: Fuzzy C-Means, Algoritmos de Clustering, Kohonen Self Organizing Maps.

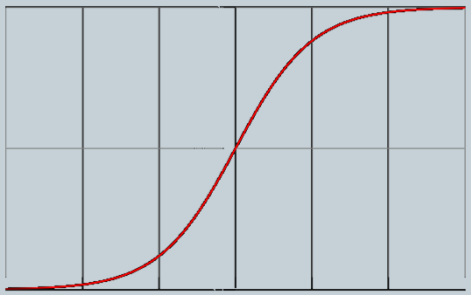
Regresión Logística

SEBASTIÁN MALDONADO.

Historia

- Alphonse Quetelet (estudio de poblaciones, S. XIX):
  - Método clásico: Regresiones lineales. $P(t) = \alpha + \beta t$
  - Poblaciones no crecen indefinidamente.
  - Problema: ¿Cómo ajustar a valores en rangos acotados?
  - Solución: ¡Decirle a su pupilo que lo solucione!
- Pierre-Françoise Verhulst:
  - Solución: Función Logística.

$$L(t) = \frac{\exp(\alpha + \beta t)}{1 + \exp(\alpha + \beta t)}$$



  - Comportamiento correcto en poblaciones en Rusia, Bélgica, etc.

Construcción

- Problema con variable dependiente (p) binaria.
- X: Regresores explicativos (dependientes).
- Regresión lineal entrega valores entre [0 - ∞] .
- Solución: Utilizar función logística para modelar el fenómeno, regresión lineal para modelar el "odd ratio" (efecto en la probabilidad).

$$p(z) = \frac{1}{1+e^{-z}} = \frac{e^{z}}{1+e^{z}}$$

$$z = b_0 + b_1 X_1 + b_2 X_2 + \ldots + b_k X_k$$

Construcción (2)

- Ahora sí se tiene una variable que puede tomar cualquier valor, por lo que se plantea el buscar para ella una ecuación de regresión tradicional:

$$y = \ln\frac{p}{1-p} = x_1\beta_1 + x_2\beta_2 + \ldots + x_K\beta_K + \varepsilon$$

- $\ln\frac{p}{1-p}$ se conoce como la función de enlace logit.

- Linealiza la relación entre probabilidad modelada y componente sistemático.

Interpretación de Coeficientes

- De la ecuación:
  - Aumento en X, con coeficiente positivo -> aumento en posibilidad de evento.
  - Aumento en X, con coeficiente negativo -> disminución en posibilidad de evento.
- Coeficiente sin variable (constante): “riesgo intrínseco” a la población. Riesgo del modelo.
- Logit: “Odds ratio” para el elemento.

$$\frac{p}{1-p} = \exp\left(\frac{z}{1-z}\right)$$

Obtención de Parámetros

- Método: Estimación según “Máxima Verosimilitud”.
  - Método para estimar parámetros.
- Alternativa: Mínimos cuadrados (regresión lineal).
  - Ventaja: Sencillez de implementación.
  - Desventaja 1: Requiere normalidad en las variables (rarísimo!!).
  - Desventaja 2: Entrega resultados sesgados (coeficientes sobredimensionados).
- Ventaja MV: No existe restricción sobre variables. (!)
  - Desventaja: Método complejo, imposible previo a los ‘90.

Método de Máxima Verosimilitud

- Muestra con estimadores p(x) (modelo logístico).

$$p(x_i) = p(y = 1|x_i) = \frac{\exp\left(\beta_0 + \sum_{j=1}^{V} \beta_j x_{i,j}\right)}{1 + \exp\left(\beta_0 + \sum_{j=1}^{V} \beta_j x_{i,j}\right)}$$

- Para estimar se maximiza función de verosimilitud:

$$\max_{\beta} \mathcal{L}(X) = \left(\prod_{i \backslash y_i = 1} p(x_i)\right)^{n_1} \left(\prod_{i \backslash y_i = 0} (1 - p(x_i))\right)^{N - n_1}$$

  - n_1: Cantidad de casos positivos.
- Estimadores corresponden a solución de este problema.
- Supuesto: Muestra y variables independientes.

Propiedades Máxima Verosimilitud

- Invarianza Funcional: Si θ es un parámetro a estimar, $\hat{\theta}$ su estimador y $f(\cdot)$ una función cualquiera.

$$\gamma = f(\theta) \Rightarrow \hat{\gamma} = f(\hat{\theta})$$

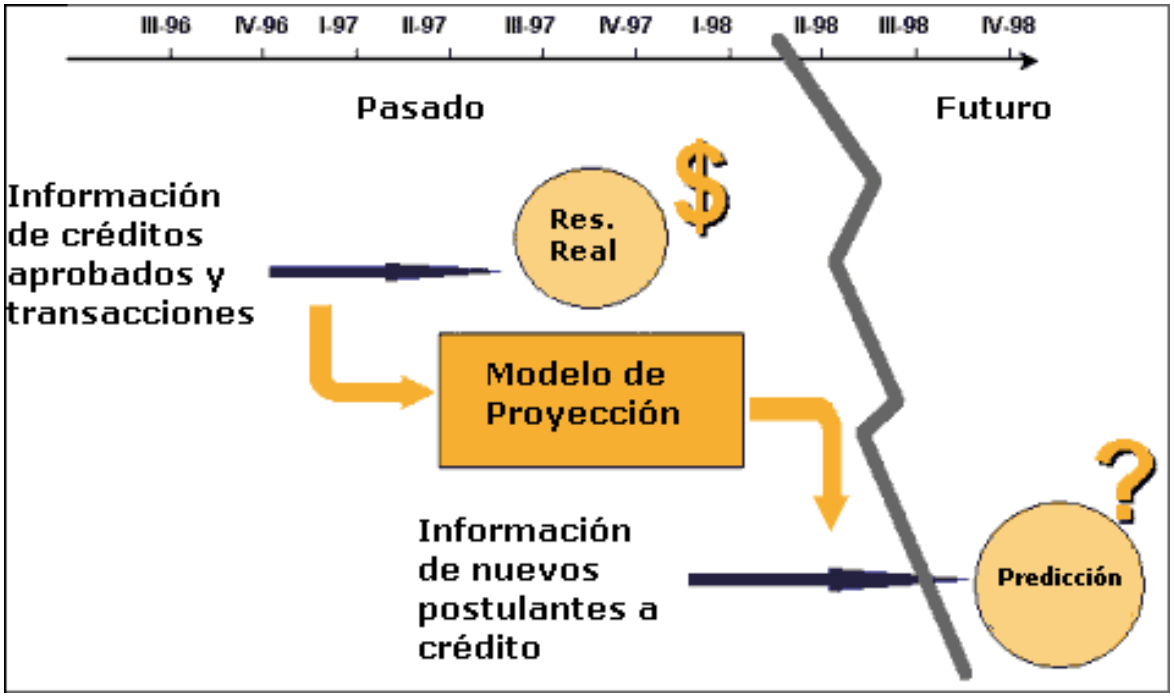
  - Estimador del parámetro asociado a la función es la función del estimador.
- Propiedades asintóticas:
  - Estimador es asintóticamente insesgado.
  - Estimador es asintóticamente eficiente.
    - Eficiente: Alcanza límite inferior de Cramer – Rao cuando tamaño del dataset tiende a infinito (mínimo error cuadrático medio).
  - Estimador es asintóticamente normal.

Aplicación: Credit Scoring

- **¿Qué es Credit Scoring?**
  - Método cuantitativo usado para predecir la probabilidad de que un aspirante a un crédito no sea buen pagador en caso de recibirlo.
  - Basado en información histórica del postulante: historial de pago de boletas, créditos anteriores, deuda, etc.

- **¿Por qué es importante?**
  - Morosidad trae altos costos para la empresa crediticia.
  - Es esencial para la entidad decidir de manera rápida y transparente quién es un buen candidato a un crédito y quién no lo es.

Funcionamiento



Beneficios

- Ayuda a reducir la discriminación.
  - Provee un análisis objetivo del mérito del postulante para recibir un crédito.
  - Permite a los proveedores enfocarse sólo en la información relacionada con la asignación del crédito y así evitar subjetividad.
- Permite acelerar y a hacer más consistente el proceso de asignación de créditos.
  - Proceso más automatizado.
  - Reduce significativamente la necesidad de intervención humana y por ende los costos asociados a este proceso.

Beneficios (2)

- Permite utilizar la información generada para formular mejores estrategias de cobranza y utilizar sus recursos más eficientemente.
- En particular: realizar una mejor predicción de las reclamaciones, controlar el riesgo de manera efectiva y determinar el precio de los seguros de manera adecuada.
- Permite ofrecer mayor cobertura a más clientes a un precio equitativo, reaccionar rápido ante los cambios del mercado y obtener ventajas competitivas.

Árboles de Decisión y Clasificación

SEBASTIÁN MALDONADO.

Árboles de Clasificación

- Corresponde a la técnica más famosa de clasificación. Nace en los años ‘70.
- Idea natural: una decisión nace de una serie de condiciones en cadena.
- Ej: Diagnóstico médico.

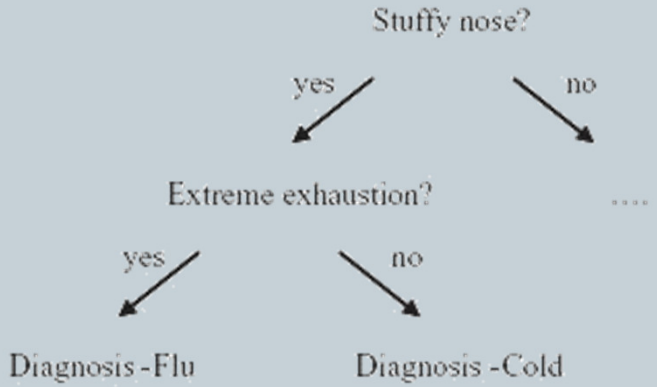


Figure 6.31. Decision tree for the diagnosis of colds and flu

Tipos de Algoritmos

- Algoritmo ID3 (*Iterative Dichotomiser 3*)
  - Desarrollado por Ross Quilan el año 1983
  - Solamente permite clasificar base de datos con atributos categóricos.

- Algoritmo C4.5
  - Desarrollado por Ross Quilan el año 1993
  - Básicamente el mismo algoritmo ID3, pero considera la posibilidad de clasificar con atributos con valores continuos.
  - Se generan rangos que permiten manejar los valores continuos como valores categóricos.
    - Test Chi – Cuadrado es de mayor importancia.

Construcción

1. Origen: Base de datos con N atributos y dos clases: positivo, negativo.
2. Partir de un nodo con todos los elementos. Tomar ese nodo.
3. Criterio de detención:
   - Si en el nodo tengo sólo atributos de la misma clase.
     - Asignar a nodo la clase, volver a nodo anterior. Si estoy en nodo original (raíz), terminar.
   - Si no me quedan atributos.
     - Asignar a nodo la clase con mayor cantidad de elementos (votación por mayoría). Volver a nodo anterior.
4. Si tengo atributos...
   - Elegir aquel atributo ¿que mejor clasifique? a los datos.
   - Ramificar utilizando ese nodo.
   - Avanzar al nodo siguiente y volver a 3.

Método de Selección de un Atributo

- Se necesita una medida de la cantidad de información que aporta un atributo.
- Ej: Entropía (de la teoría de la información).
  - $E_2(K) = - p^+ * \log_2 p^+ - p^- * \log_2 p^-$ (Entropía de un nodo)
  - K: Nodo considerado
  - p^+ / p^- : frecuencia relativa de ejemplos positivos/negativos en nodo K. ($p^+ + p^- = 1$)
  - Definimos $0*\log_2 0 := 0$.
  - $E_2(K) \geq 0$, $E_2(K) = 0 \Leftrightarrow p^+ = 0$ o $p^- = 0$.
    - Entropía de K es máximo $\Leftrightarrow p^+ = p^-$
- Algoritmo: Seleccionar nodo con menor entropía (Mejorar aporte de información).
  - ID3 utiliza entropía.

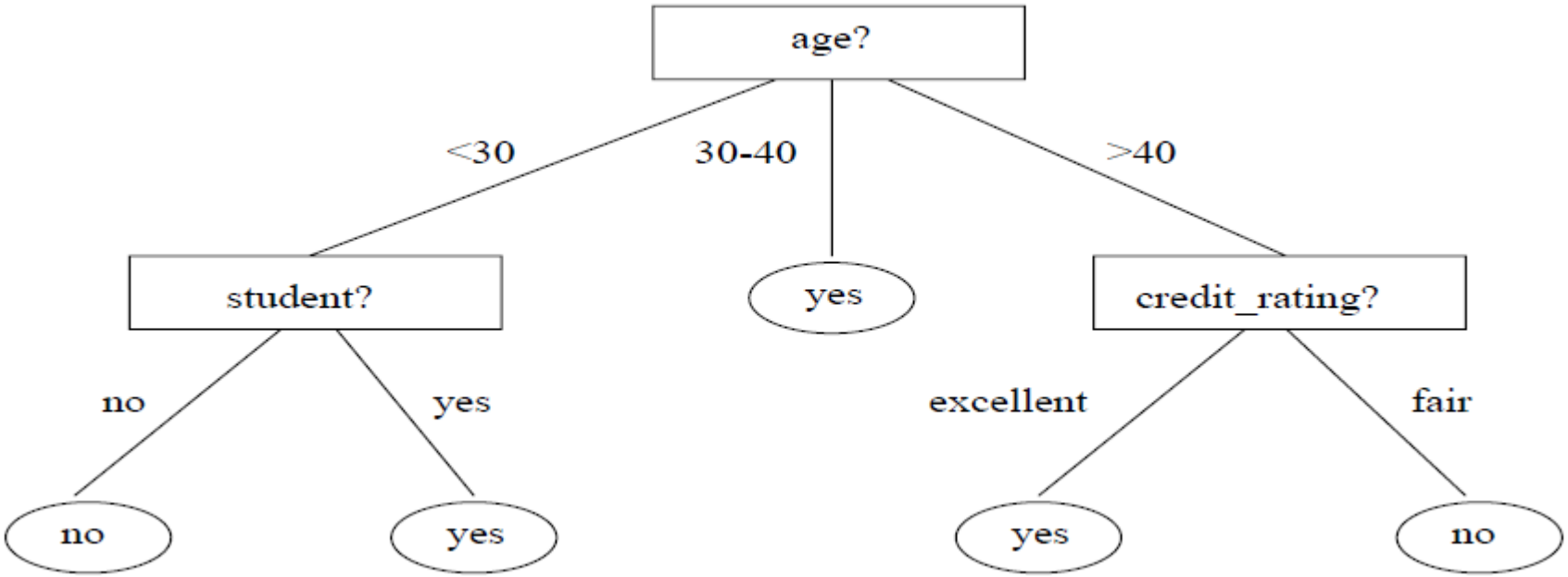
Construcción (ID3)

- Para cada atributo calcular:
  - MI(K) = $\sum_{i=1}^{m} p_i * E_2(K_i)$ (Medida de Información en el nodo aportada por el atributo)
  - m: Número de valores distintos del atributo considerado.
  - p_i: Probabilidad *a priori* que un ejemplo tome el valor i del atributo considerado
  - K_i: nodo i sucediendo al nodo K (i=1, ..., m) .
  - $E_2(K_i)$: Entropía del nodo K_i (i=1, ..., m) .
  - Gain(K): Ganancia en información o reducción de entropía si se utiliza el atributo K: Gain(K)=E2(Nodo Sup.)-MI(K)

Ejemplo

rid	age	income	student	credit_rating	Class: buys_computer
1	<30	high	no	fair	no
2	<30	high	no	excellent	no
3	30-40	high	no	fair	yes
4	>40	medium	no	fair	yes
5	>40	low	yes	fair	yes
6	>40	low	yes	excellent	no
7	30-40	low	yes	excellent	yes
8	<30	medium	no	fair	no
9	<30	low	yes	fair	yes
10	>40	medium	yes	fair	yes
11	<30	medium	yes	excellent	yes
12	30-40	medium	no	excellent	yes
13	30-40	high	yes	fair	yes
14	>40	medium	no	excellent	no

Ejemplo


age?
<30
30-40
>40
student?
yes
credit_rating?
no
yes
excellent
fair
no
yes
yes
no

Otros Métodos de Selección

- Índice de Gini:
  - Pretende medir el grado de impureza de un nodo.
  - Tiende a crear ramificaciones desbalanceadas, agrupando una clase mayoritaria en un nodo, y el resto en otros nodos.
- Índice de Twoing:
  - No es una medida de impureza.
    - NO alcanza máximo cuando la impureza es máxima.
  - Es recomendada cuando existen más de dos clases en el atributo objetivo.

Sobreajuste

- Se dice que un modelo **M** está **sobre-ajustado**, si existe otro modelo **m'** tal que.
  - Error de M en conjunto de **entrenamiento** es **menor** a Error de m' en conjunto de **entrenamiento** y…
  - Error de M en conjunto de **testeo** es **mayor** a Error de m' en conjunto de **testeo**.
  - Bajo poder de predicción para muestras que no están representadas por aquellas utilizadas para entrenar el modelo. Se **pierde generalidad.**

- En árboles de decisión:
  - Puede ocurrir cuando hay muchas ramas.
  - Peor caso cuando el árbol crece de manera que separa cada dato presente en la muestra de todos los demás.
    - Regla resulta asignar cada muestra nueva a su objeto más parecido…

Evitando el Sobreajuste en la Clasificación

- Pre-poda: **Parar la construcción del árbol antes que se termine de construir.**
  - Entropía.
  - Índice de Gini, Twoing.
  - Test Chi-cuadrado.
  - No expandir según algún criterio:
    - Detener el crecimiento del árbol dado un número mínimo de muestras presentes en un nodo o después de cierto número de niveles.
- Post-Poda**: Remover las ramas de un árbol de decisión completo.**
  - Se debe utilizar un conjunto de datos distinto que el de entrenamiento.
  - Al plantear el problema como un modelo de optimización, por el momento solo se deben considerar Heurísticas
    - Problema NP-hard.

Ventajas y Desventajas

- Ventajas:
  - Fáciles de entender y transparentes (uso legal en EE. UU.).
  - Acepta todo tipo de variables y valores (¡nulos incluidos!)
  - Descubre relaciones complejas (en las variables).
- Desventajas:
  - Cuando hay muchas variables se vuelve complejo de entender.
  - No estudia relaciones no lineales.
  - Cuando hay muchas variables continuas, pueden ser mejor otros métodos.

Redes Neuronales

SEBASTIÁN MALDONADO

Definición

- "… a neural network is a massively parallel distributed processor that has a natural propensity for storing experiential knowledge and making it available for use. It resembles the brain in two respects:
  - Knowledge is acquired by the network through a learning process.
  - Interneuron connection strengths known as synaptic weights are used to store the knowledge".

Haykin (1994), p. 2

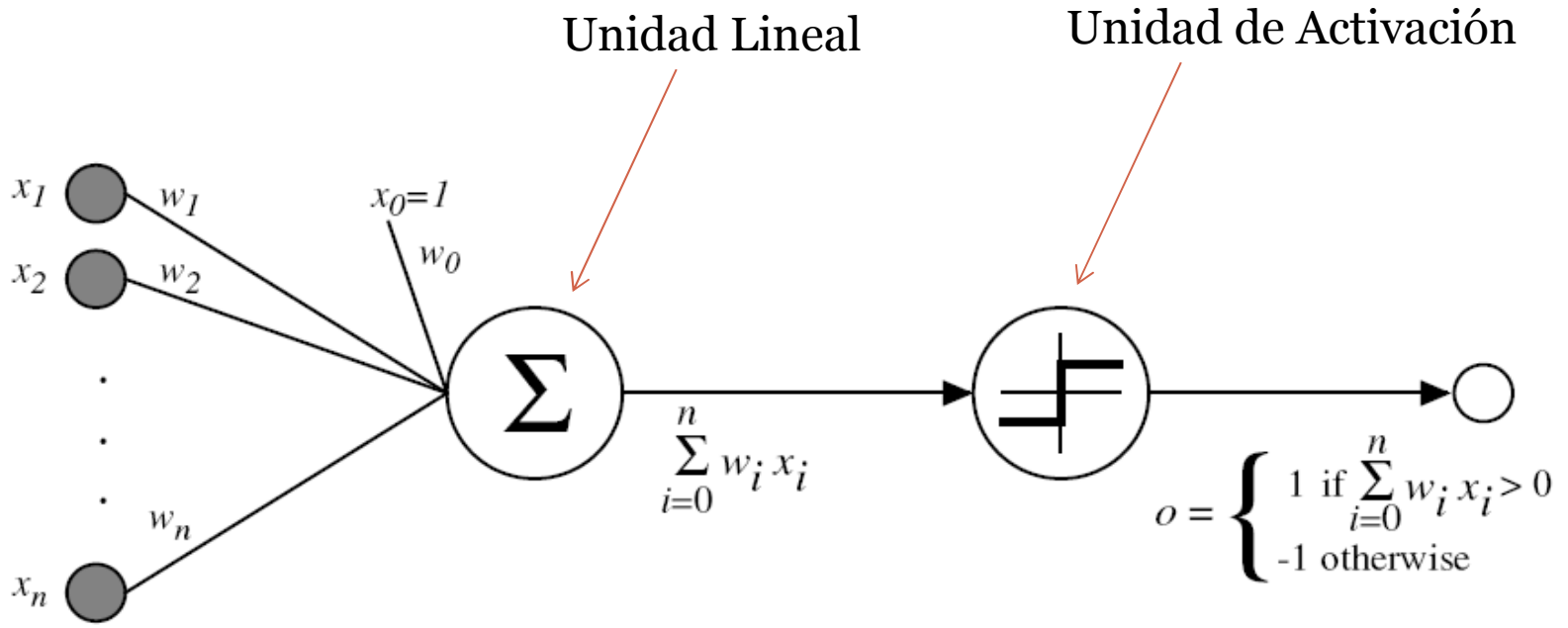
Redes Neuronales

- Método de regresión y clasificación de aprendizaje supervisado.
  - Las redes neuronales pertenecen al conjunto de herramientas de clasificación y regresión no lineal.
  - Se ha demostrado que es un "aproximador" universal.
    - Cualquier función continua se puede aproximar por una red neuronal (en particular utilizando un perceptrón multicapa).
  - Modelo adecuado para abordar un gran número de problemas. Puede :
    - Aproximar funciones no lineales.
    - Filtrar ruido en los datos.

Algoritmo Perceptrón

- Perceptrón (1958, Rosemblatt)
  - Modelo de clasificación que define un límite de decisión para clases +1 y -1.
  - Puede representar cualquier límite de decisión lineal.
  - Sólo permite entradas binarias.
  - Sólo puede clasificar datos que sean linealmente separables, ya que el algoritmo mediante el cuál se entrena el modelo sólo converge si se tiene tal propiedad (teorema de convergencia del perceptrón).

Algoritmo Perceptrón (2)

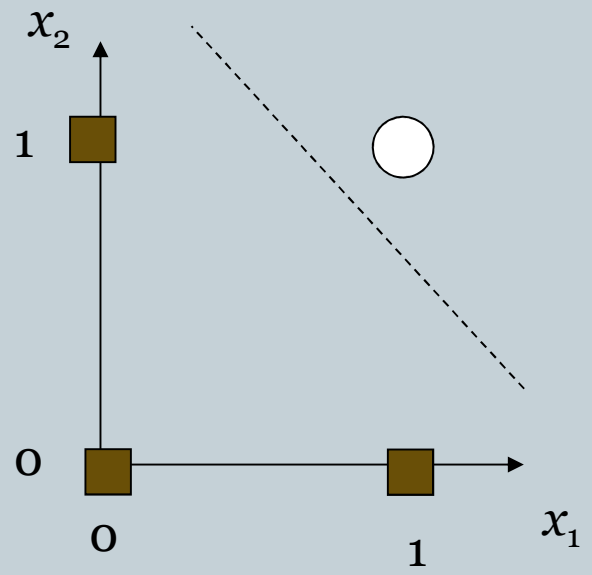


$$o(x_1, \ldots, x_n) = \begin{cases} 1 & \text{if } w_0 + w_1 x_1 + \cdots + w_n x_n > 0 \\ -1 & \text{otherwise.} \end{cases}$$

Representación Funciones Lógicas

Función AND

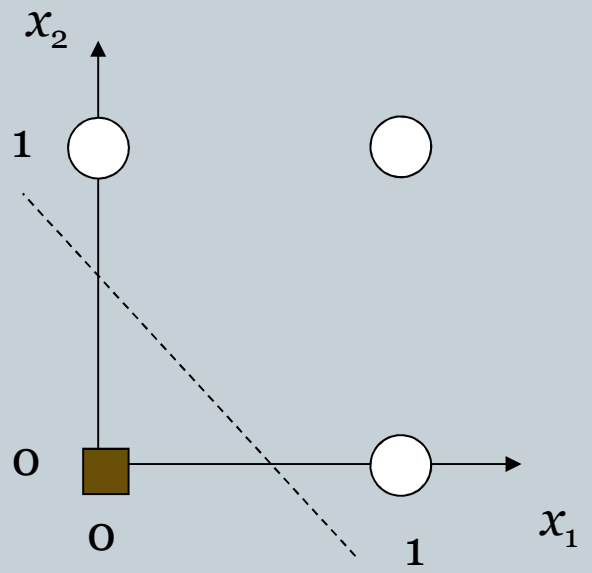
x_1	x_2	y
0	0	0
0	1	0
1	0	0
1	1	1



Representación Funciones Lógicas (2)

Función OR

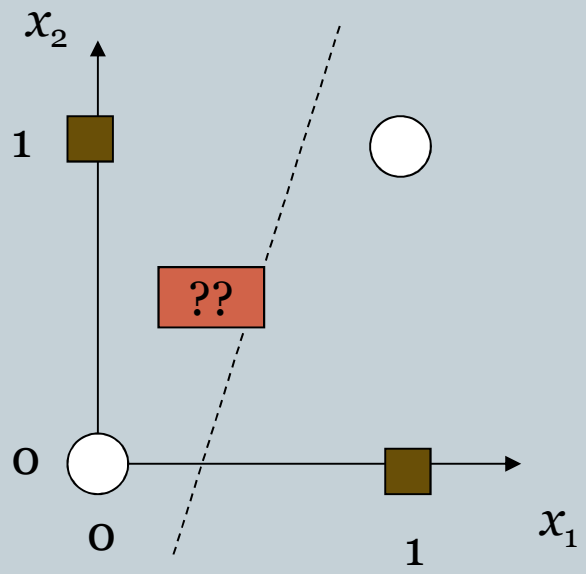
x_1	x_2	y
0	0	0
0	1	1
1	0	1
1	1	1



Representación Funciones Lógicas (3)

Función XOR

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



¡Perceptrón no puede por si sólo modelar la función booleana XOR!

Limitaciones del Algoritmo

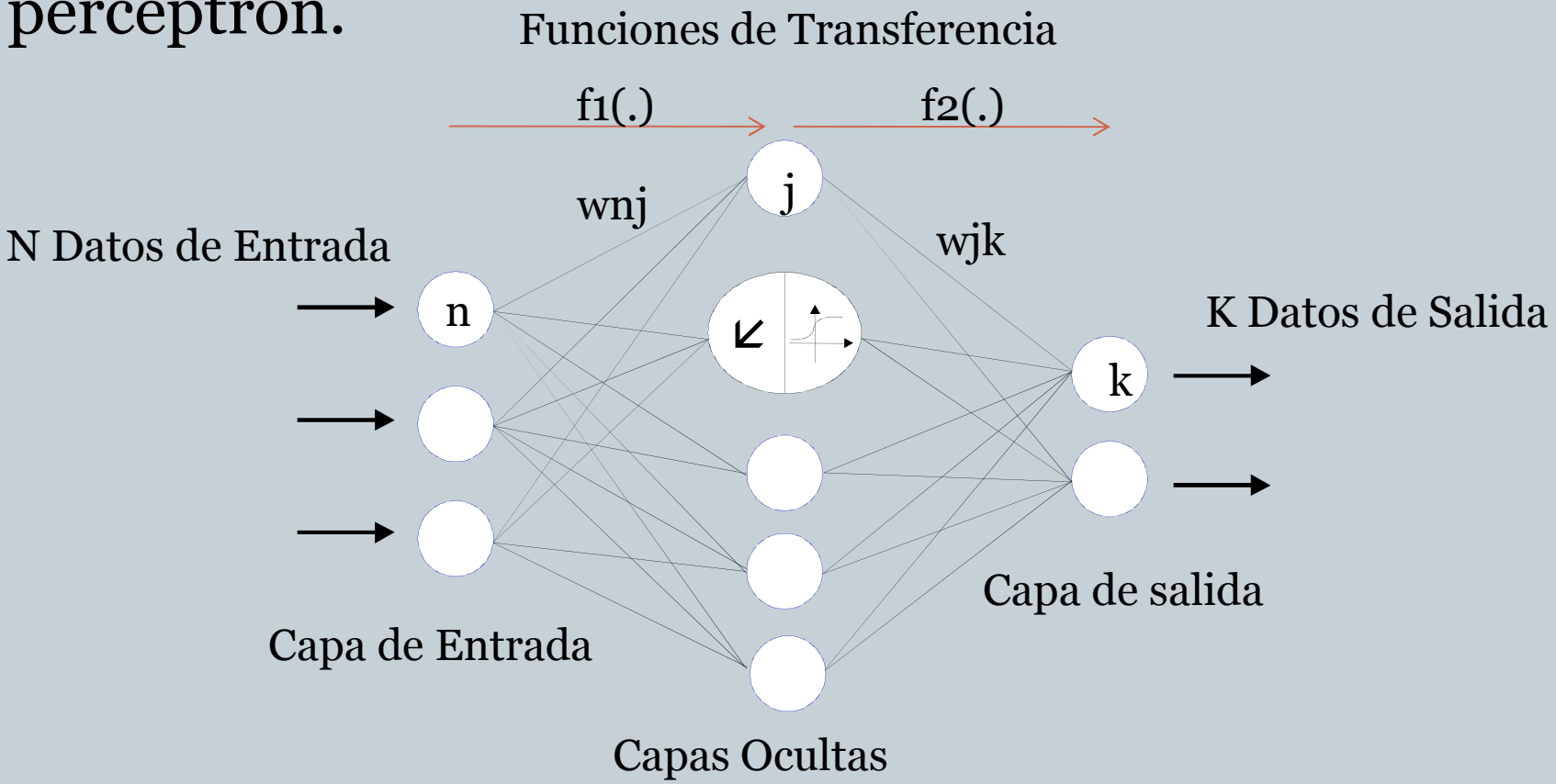
- Consecuencia:
  - *"The perceptron has shown itself worthy of study despite (and even because of!) its **severe limitations**. It has many features to attract attention: its linearity; its intriguing learning theorem; its clear paradigmatic simplicity as a kind of parallel computation. There is no reason to suppose that any of these virtues carry over to the many-layered version. Nevertheless, we consider it to be an important research problem to elucidate (or reject) our intuitive judgement that the extension to multilayer systems is sterile."*

M. Minsky and S. Papert, 1969, "Perceptrons"

- Publicación detuvo la investigación de redes neuronales por casi 15 años.

Perceptrón Multicapa

- Nace como la respuesta a los problemas del perceptrón.



Estructura Perceptrón Multicapa

- Característica Principal: ¡Nodos asignados en capas!
  - **Capa de entrada**: se encargan de recibir las señales o patrones que proceden del exterior y propagar las señales a todas las otras neuronas de la siguiente capa.
  - **Capas ocultas**: son las que tienen la misión de realizar el procesamiento no lineal de los patrones recibidos.
  - **Capa de salida**: actúa como salida de la red, proporcionando al exterior la respuesta de la red, para cada uno de los patrones de entrada.
  - **Funciones de Transferencia:** Función que lleva las salidas de una neurona a otra. Corresponde a la "sinapsis" y es el centro de la técnica.

Activación de Capas

Las activaciones, en la primera iteración, se calculan de la siguiente manera:

- Para las neuronas de la capa de entrada (a_i^1), se tiene que las señales recibidas del exterior activan a cada una de las *i* neuronas. Es decir,

$$a_i = x_i \; \forall i \in \{1, \dots, N\}$$

  donde $X = (x_1, x_2, \dots, x_N)$ representa el vector asociado a una observación que entrada a la red.

- Luego de la activación de las neuronas de la capa oculta *c* se lleva a cabo aplicando una función de activación f_1 a la suma de los productos de las activaciones que recibe por sus correspondientes pesos, es decir,

$$a_i^c = f_1\left(\sum_{k=1}^{K} w_{j,i}^{c-1} \cdot a_j^{c-1} + u_i^c\right)$$

  donde a_j^{c-1} son las activaciones de las neuronas de la capa *c*-1.

Activación de Capas (2)

- La activación de las neuronas de la capa de salida viene dada por una función de activación f_2 sobre una expresión similar a la de las capas ocultas.
- Existe una gran variedad de funciones de transferencia que se pueden ajustar a cada caso de estudio. Por ejemplo si se tienen datos normalizados entre 0 y 1, solamente se pueden utilizar funciones de transferencia que puedan operar en ese intervalo.
- Para estandarizar en un rango 0 y 1 se debe considerar la siguiente función:

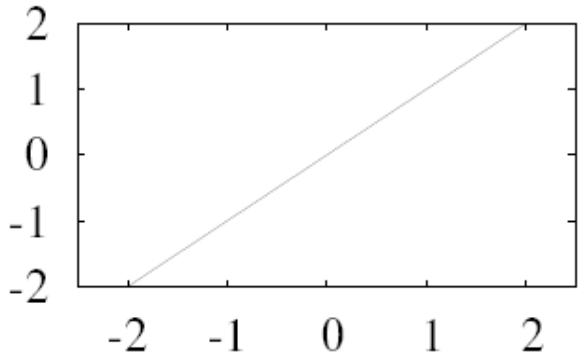
$$\Phi(x) = \frac{x - min_x}{max_x - min_x}$$

- La función sigmoidal $f_1(x)$, representa una de las principales funciones utilizadas, junto con la función identidad (o lineal).

$$f_1(x) = \frac{1}{1 + e^{-x}}$$

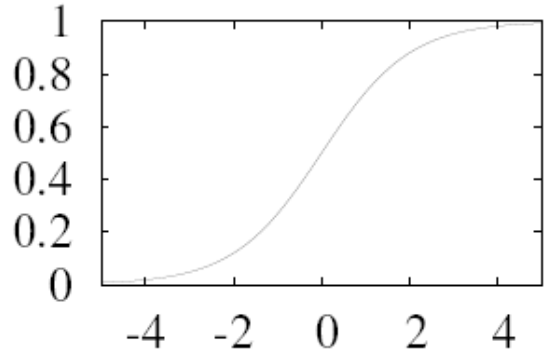
Funciones de Transferencia

Identity (Linear)



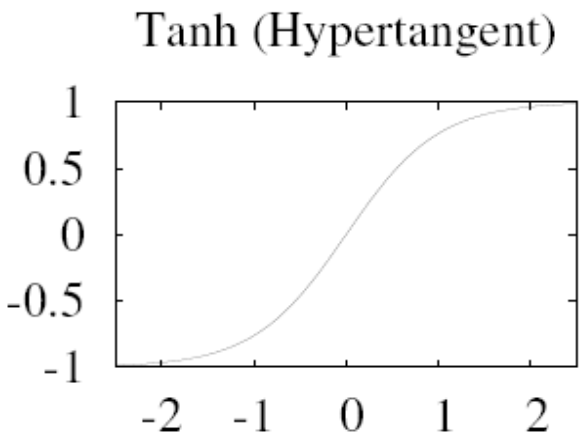
$$identity(x) = x$$

Sigmoid

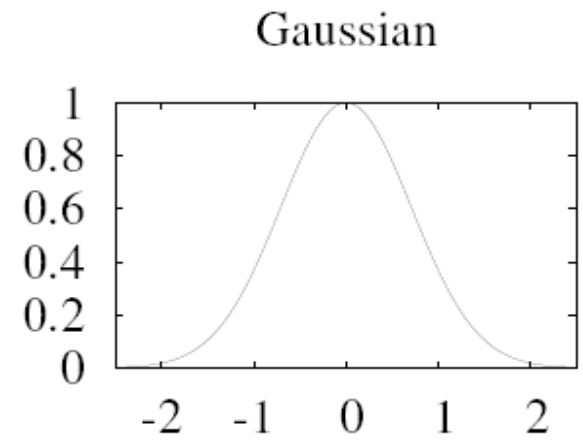


$$sigmoid(x) = \frac{1}{1 + e^{-x}}$$

Funciones de Transferencia (2)



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



$$gaussian(x) = e^{-x^2/\sigma^2}$$

Salidas de la Red

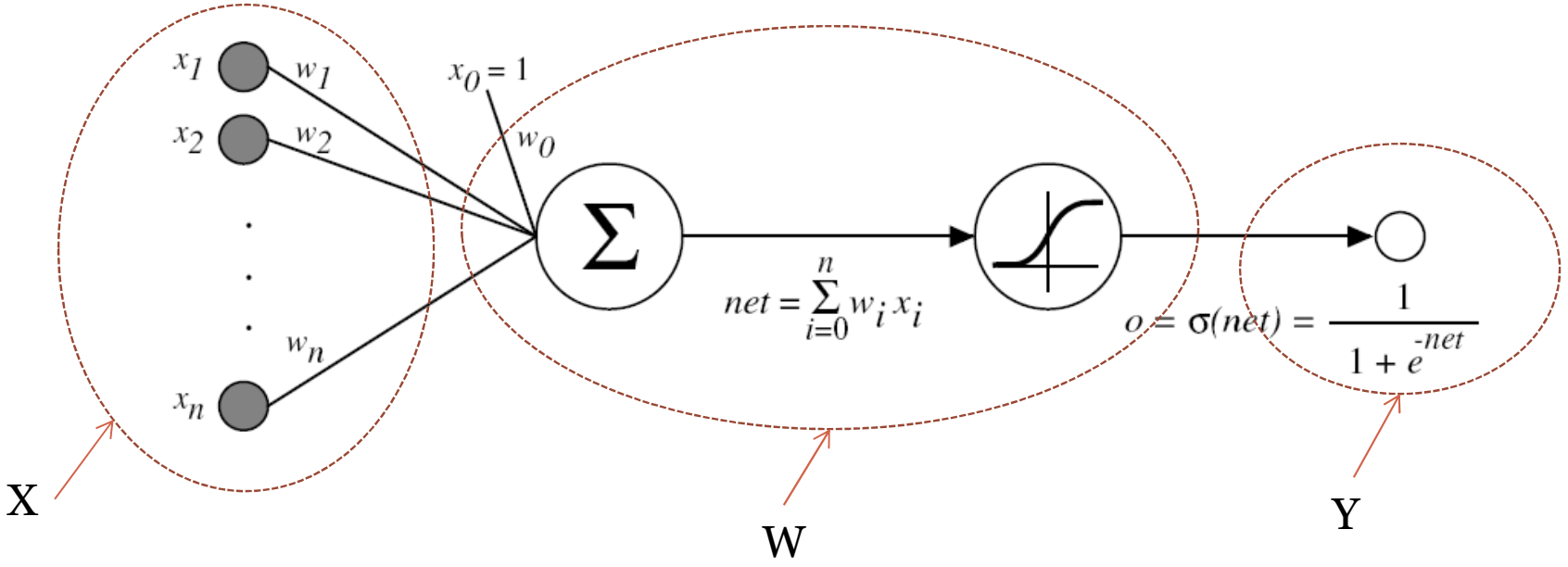
- Se define $Y = (y_1, y_2, \ldots, y_K)$ como el vector de salida de la red.
- Finalmente se puede ver que la Red Neuronal es una función $F: \mathbb{R}^N \rightarrow \mathbb{R}^K$ continua no lineal que:
  - Permite obtener el resultado asociando los valores de las variables de entrada (Atributos de las observaciones) a un espacio continuo de salida.
    - Dimensión de entrada igual al número de atributos de entrada.
    - Dimensión de salida igual al número de clases.
  - La función F se puede representar por una expresión analítica, que por lo general es sumamente compleja (varias composiciones de funciones).

Red Neuronal - Perceptrón multicapa [10]

- Finalmente tenemos una red neuronal es una función

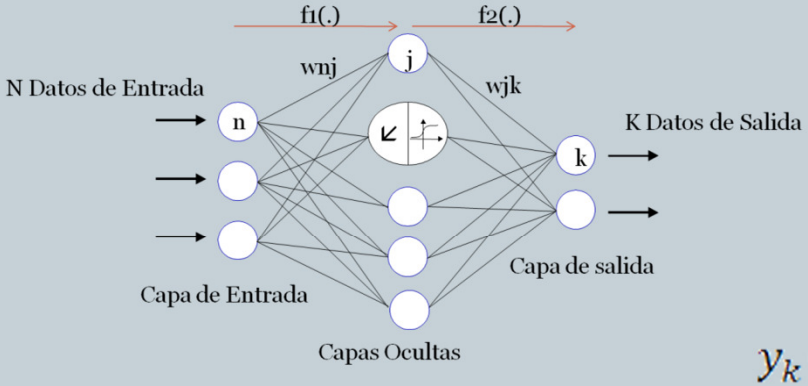
$$Y = F(X, W)$$

donde Y es el vector formado por las salidas de la red, X es el vector de entrada a la red, y W es el conjunto de todos los parámetros de la red (pesos y umbrales), y F una función continua no lineal.



Problema General

- Se desea resolver el siguiente problema de optimización:



$$\min_{w} \underbrace{\frac{1}{M}\sum_{m=1}^{M}\left(\frac{1}{2}\sum_{k=1}^{K}(s_k(m)-y_k(x_m))^2\right)}_{Error\ de\ Clasificación}$$

$$y_k(x_m) = f_2\left(\sum_{j=1}^{J} w_{k,j}\cdot f_1\left(\sum_{n=1}^{N} w_{n,j}\cdot x_n^m + u_j\right) + u_k^c\right)$$

- M = {1,..,M} cantidad de observaciones.
- K = cantidad de neuronas de salida (clases del problema).
- x_n^m = valor del atributo n para la observación m.
- $S_k(m)$ = 1 si el atributo m es de la clase k, 0 si no.
- $y_k(x_m)$ = valor calculado para la observación m.

Problema general: Funciones de Error

- Existen otras funciones de error posibles.
- Penalización sobre los pesos:

$$\frac{1}{2}\sum_{m=1}^{M}\sum_{k=1}^{K}(s_k(m)-y_k(x_m))^2+\gamma\sum_{(i,j)\in W}w_{i,j}^2$$

- Sensibilidad sobre las variables:

$$\frac{1}{2}\sum_{m=1}^{M}\sum_{k=1}^{K}(s_k(m)-y_k(x_m))^2+\mu\sum_{j\in Variables}\left(\frac{\partial S_k(m)}{\partial x_m^j}-\frac{\partial y_k(x_m)}{\partial x_m^j}\right)^2$$

- Función de costo “Cross Entropy”:

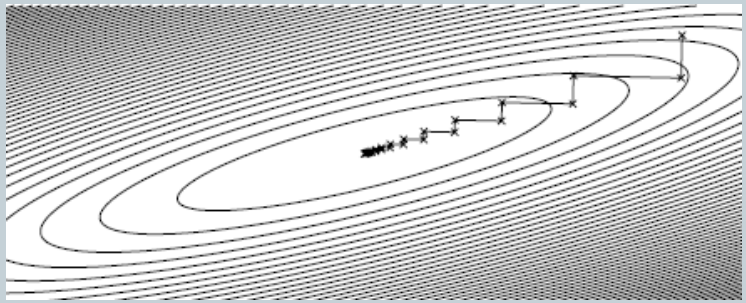
$$\sum_{m=1}^{M}\sum_{k=1}^{K}(s_k\log(y_k(x_m)))$$

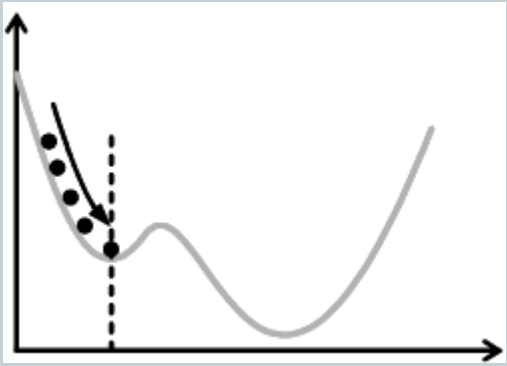
Algoritmo Backpropagation (1)

- Algoritmo que permite encontrar de manera heurística la solución al problema de minimización del error de la red neuronal.
- Descrito originalmente el año 1974 (Paul Werbos), pero no fue reconocido hasta el año 1986 (Rumerhalt et.al.).
- Compuesto de manera general por los siguientes pasos:
  1. Se presentan las observaciones a la red y utilizando los pesos actuales se calculan los valores de salida.
  2. Se calculan los errores tomando las diferencias entre los resultados obtenidos y los resultados esperados.
  3. El error se retro-alimenta a través de la red y los pesos son ajustados para minimizar el error.

Algoritmo Backpropagation (2)

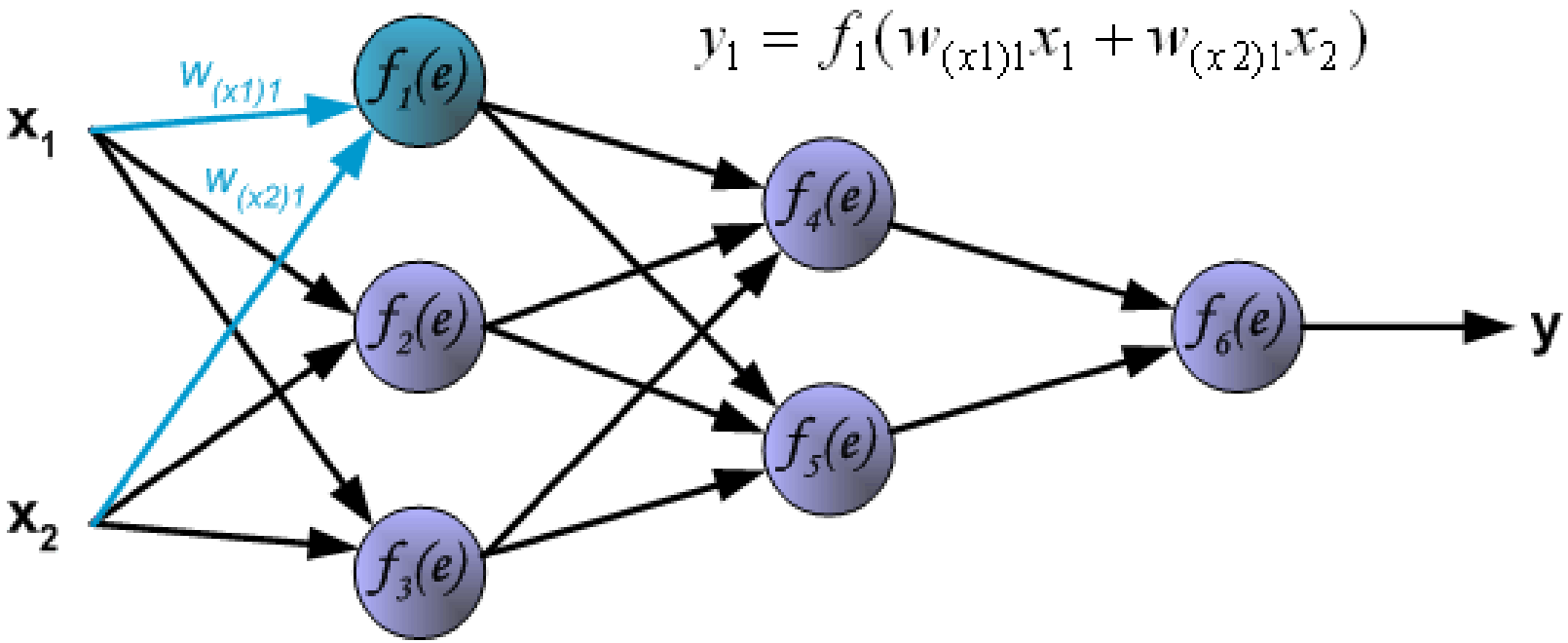
- Una de las principales problemáticas del algoritmo backpropagation es que se presenta la situación de encontrar como solución mínimos locales.





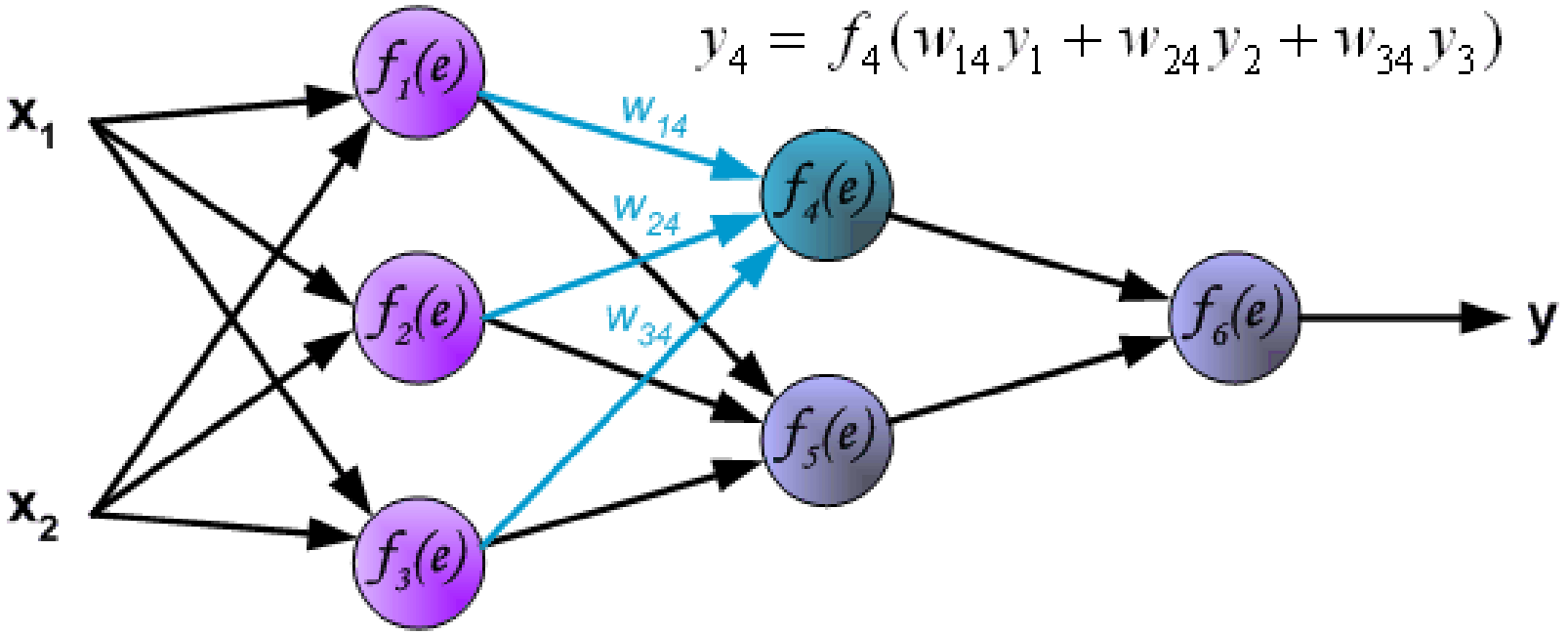
- Se puede evitar este problema modificando los valores de la tasa de aprendizaje.

Algoritmo Backpropagation: Funcionamiento



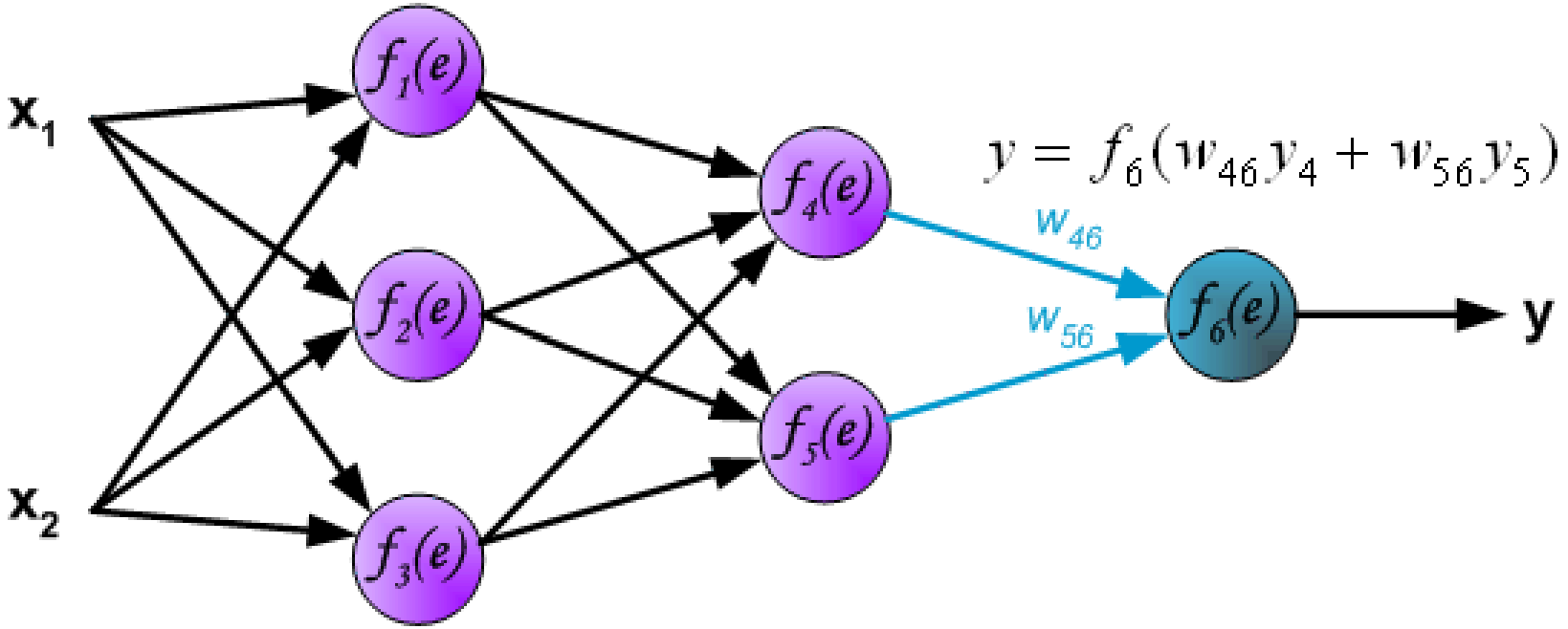
[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento



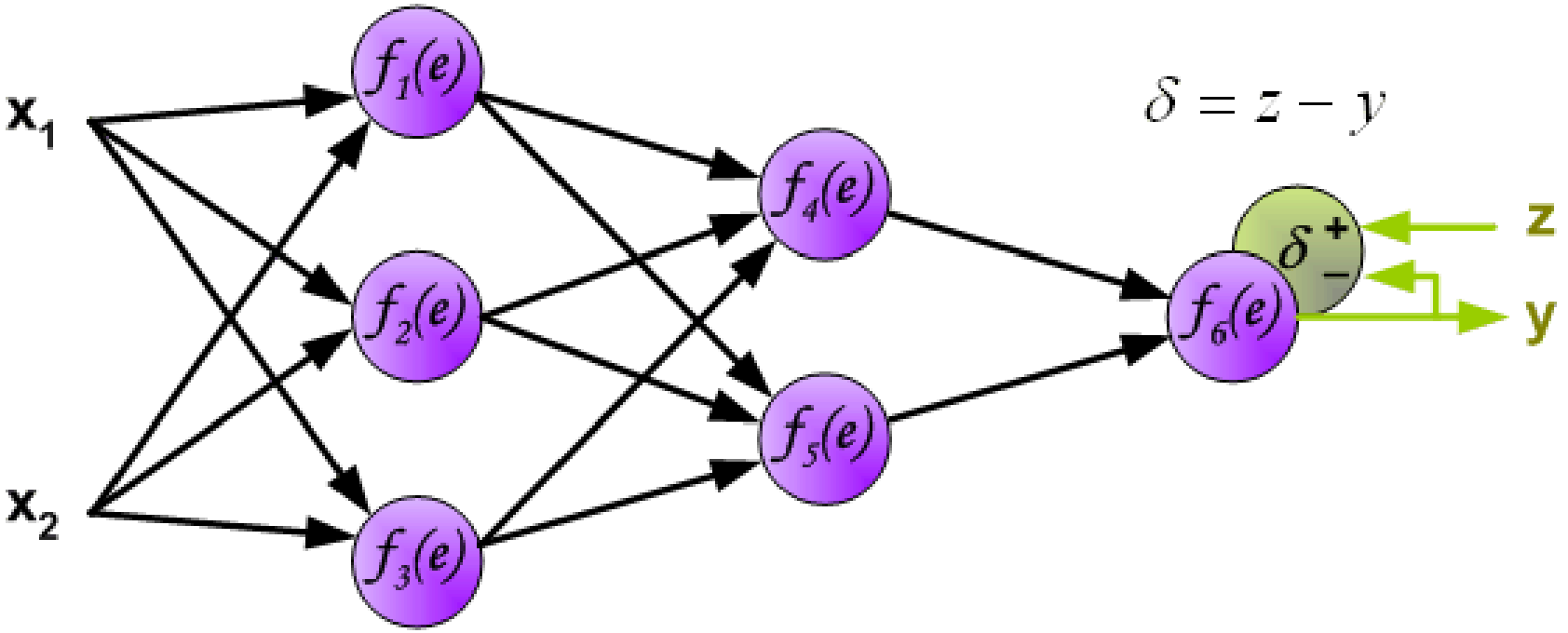
[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento



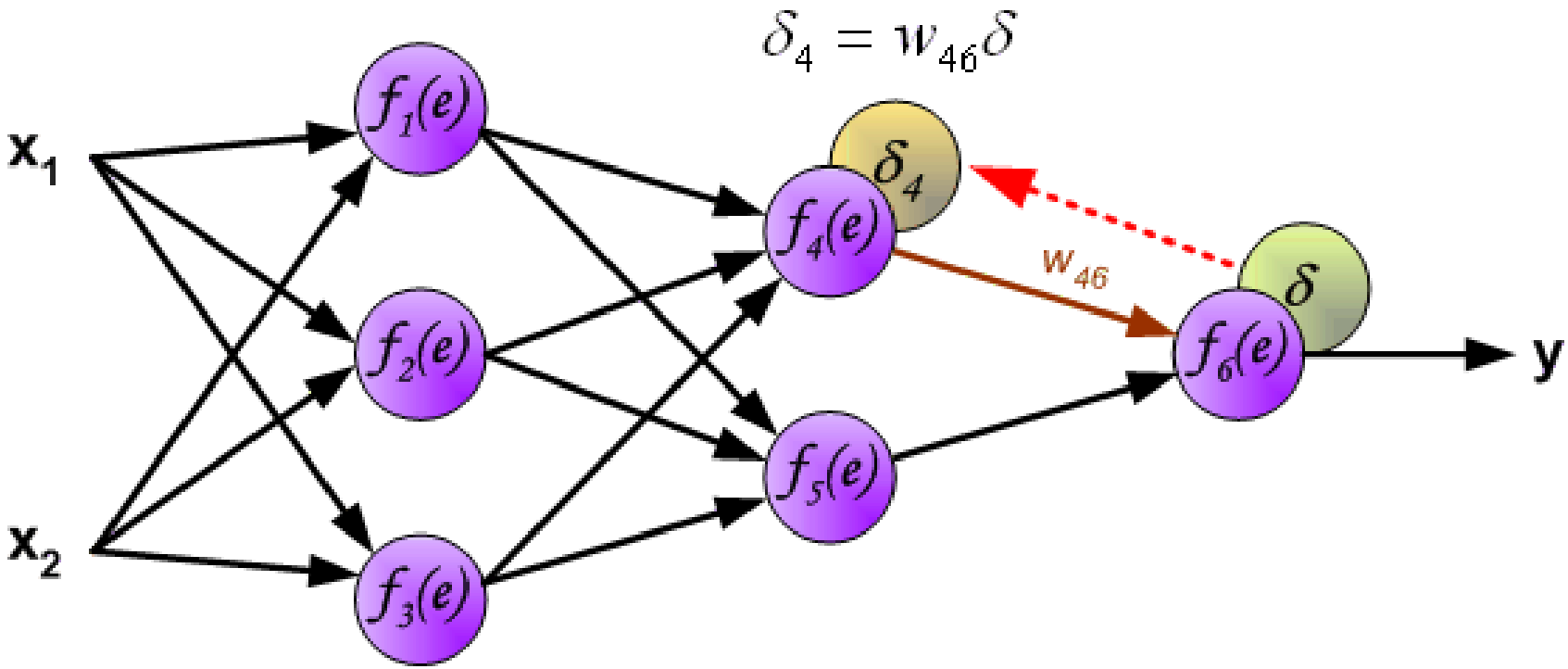
[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento (4)



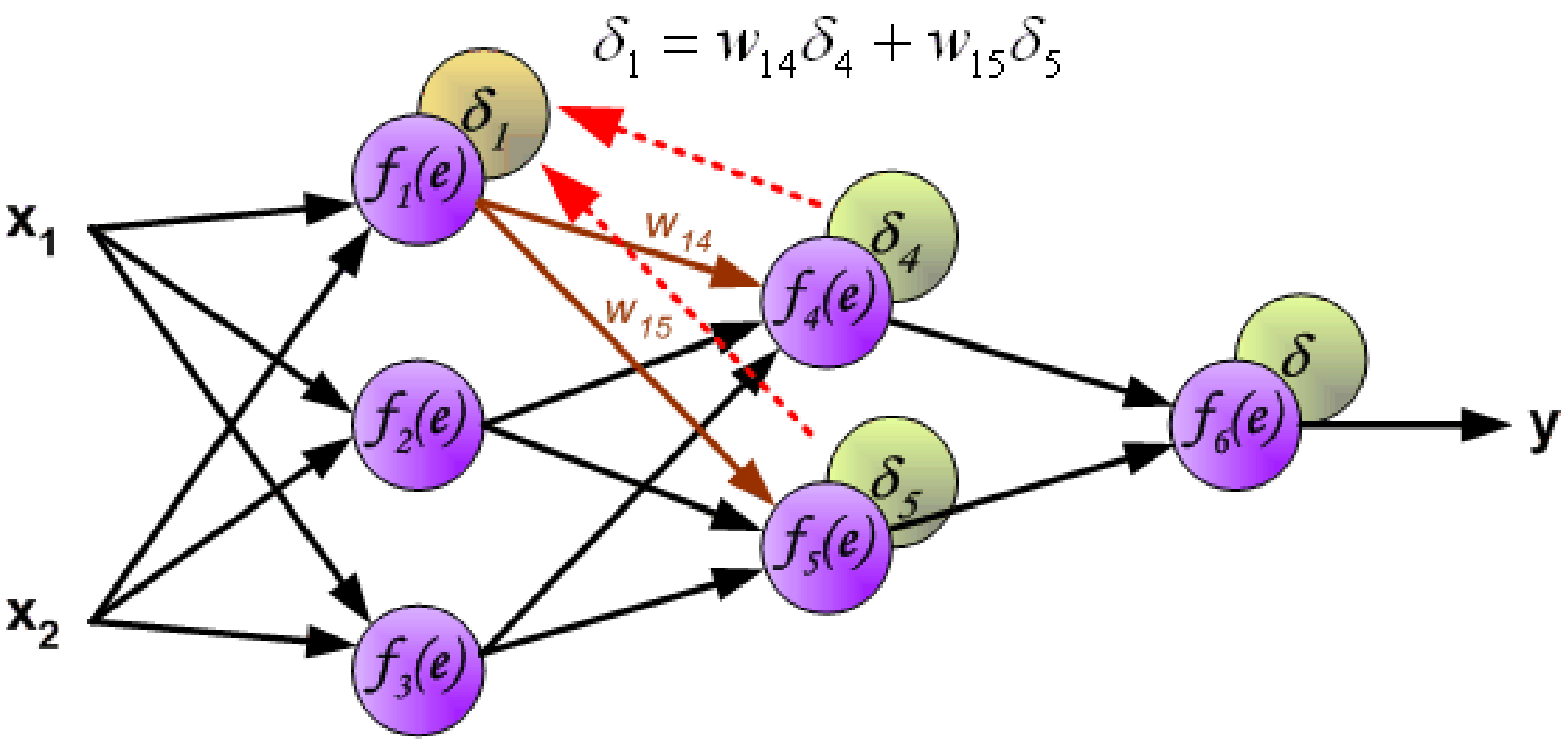
[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento



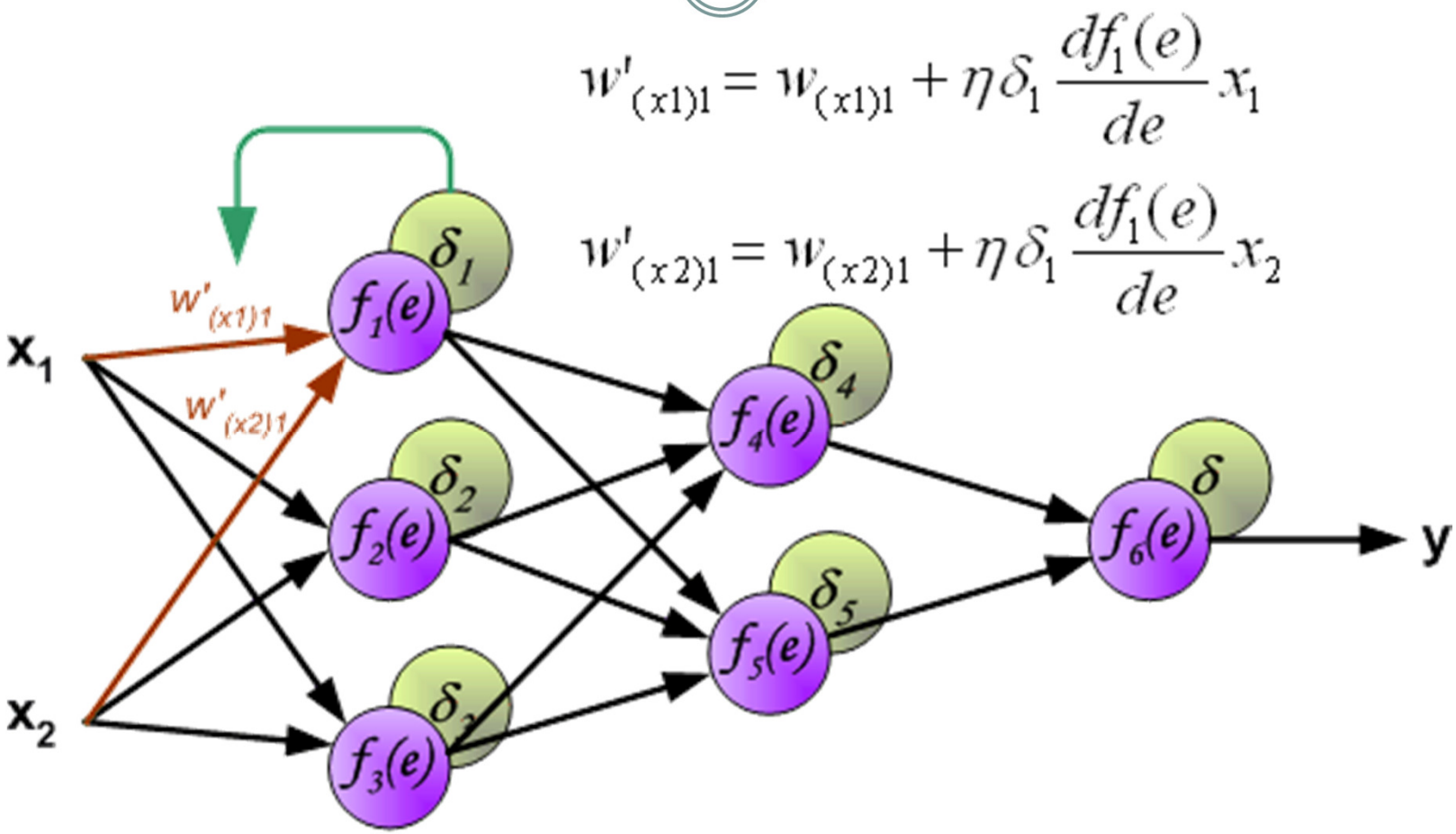
[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation [3]



[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento



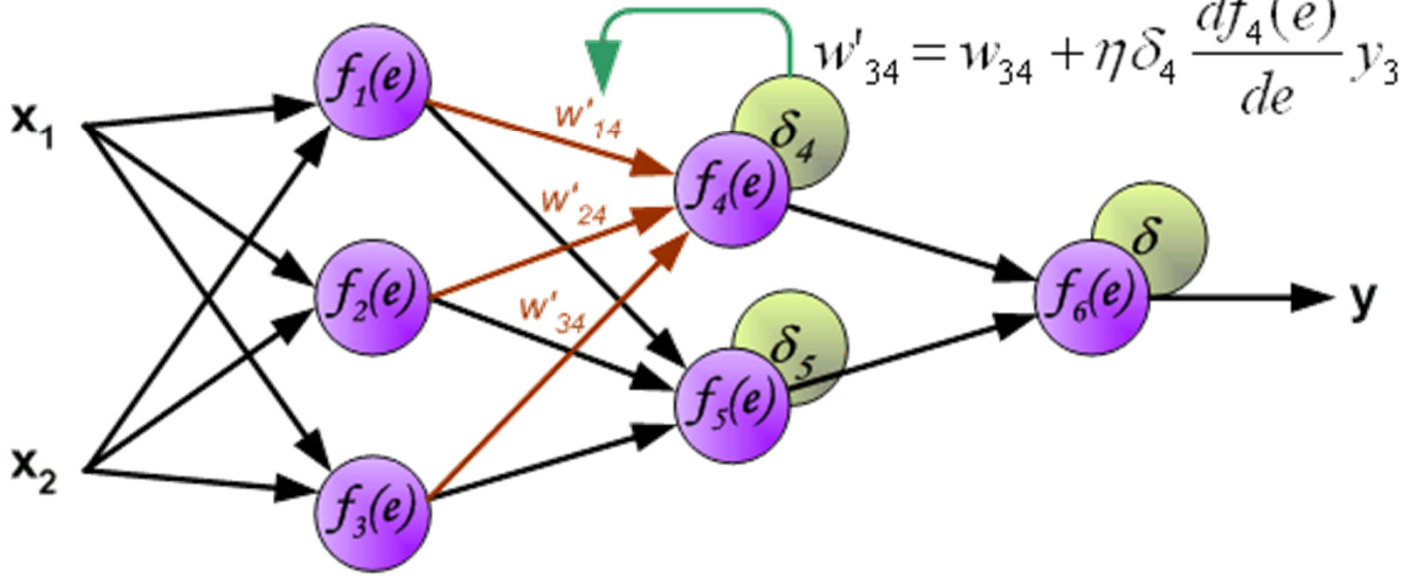
[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento

$$w'_{14} = w_{14} + \eta \delta_4 \frac{df_4(e)}{de} y_1$$

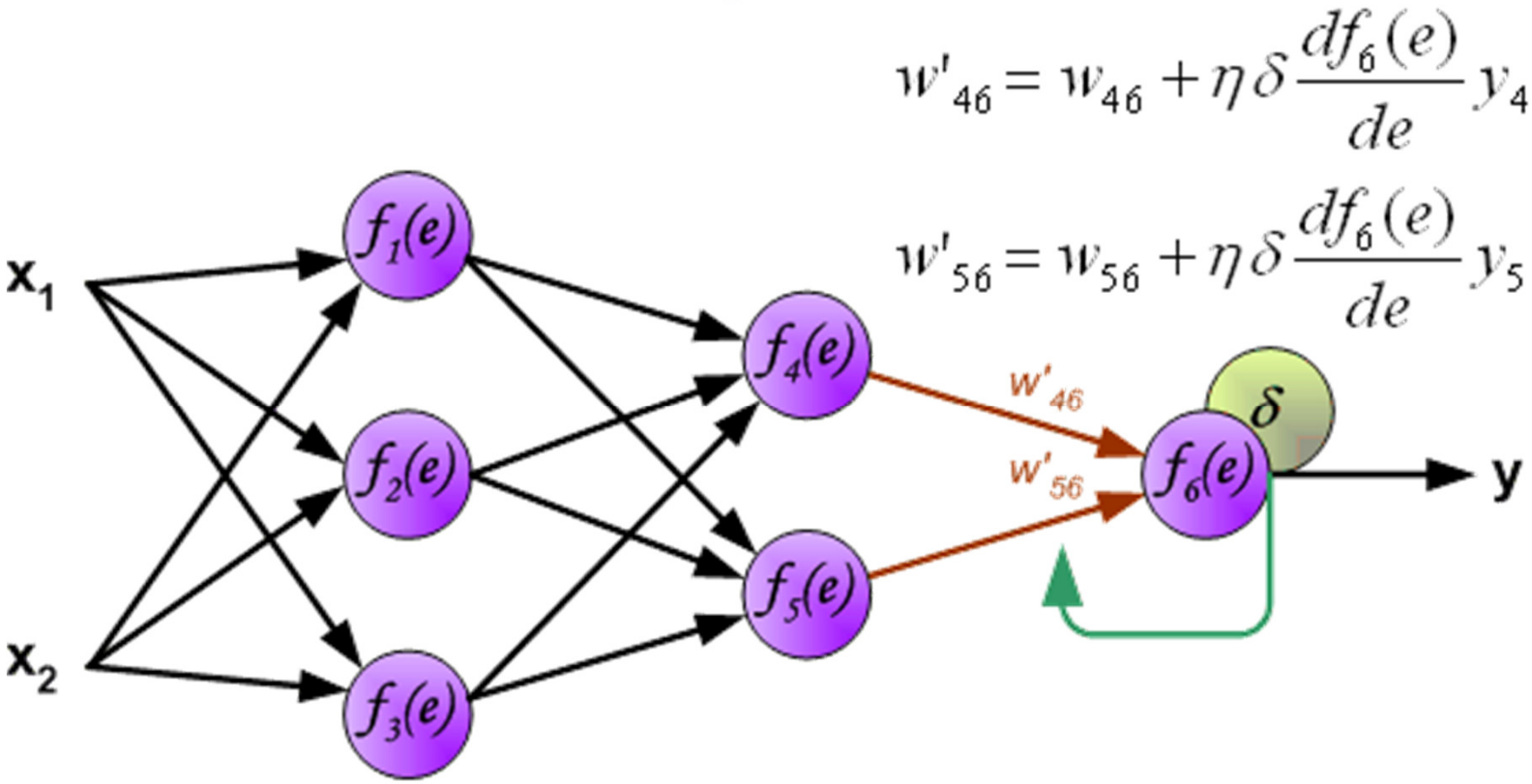
$$w'_{24} = w_{24} + \eta \delta_4 \frac{df_4(e)}{de} y_2$$

$$w'_{34} = w_{34} + \eta \delta_4 \frac{df_4(e)}{de} y_3$$



[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Algoritmo Backpropagation: Funcionamiento



[referencia: Mariusz Bernacki, **Principles of training multi-layer neural network using *backpropagation* algorithm**]

Ajuste de parámetros: Cantidad de Capas Ocultas

- Cualquier función booleana puede ser representada por una red neuronal con solo una capa intermedia. Lamentablemente puede necesitar un numero exponencial (en numero de entradas) de nodos en la capa media.
- Cualquier función continua acotada puede ser aproximada con, bajo error, por una red neuronal con una sola capa intermedia. [Cybenko 1989; Hornik et al. 1989]
- Cualquier función puede ser aproximada, con cierto nivel de precisión, con una red neuronal con dos capas ocultas. [Cybenko 1988]
- ¡¡Teoremas fundamentales de aprendizaje!!

Ajuste de parámetros: Cantidad de Neuronas

- Cantidad de Neuronas
  - La cantidad de neuronas de entrada y salida están definidas por el problema.
  - La cantidad de neuronas en las capas ocultas determina los grados de libertad del modelo:
    - Numero muy pequeño de neuronas pueden que no sean suficientes para problemas muy complejos
    - Numero muy grande de neuronas pueden sobre entrenar el modelo y tener una perdida de generalidad ante nuevas observaciones.
- Finalmente, la decisión la toma el data miner.

Ajuste de parámetros: Cantidad de Neuronas (2)

- La cantidad de neuronas en las capas ocultas depende de una serie de factores.
  - Cantidad de observaciones en el conjunto de entrenamiento.
  - Cantidad de 'ruido' en los datos.
  - Complejidad del problema de clasificación.
  - Cantidad de atributos (neuronas de entrada) y clases (neuronas de salida).
  - Funciones de activación entre las capas.
  - Algoritmo de entrenamiento. (!backpropagation no es el único!)
- Una opción es ir evaluando varias redes neuronales para ir determinando el numero apropiado de neuronas (ensayo y error).
- Otra opción es comenzar con un numero grande de neuronas y conexiones, y a medida que se va construyendo la red neuronal, se van podando aquellas conexiones que son innecesarias.

Ajuste de parámetros: Decaimiento

- Decaimiento de los pesos (Weight decay)
  - Para prevenir que los pesos vayan creciendo sin control alguno a valores muy grandes (señal de sobre entrenamiento), es conveniente agregar un decaimiento a los pesos de la forma:

$$w(n+1) = (1-\varepsilon)w(n)$$

  - Pesos que no son necesarios y no se van actualizando en cada iteración del algoritmo, van a decaer hasta anularse, mientras que aquellos que si son necesarios y se van actualizando de manera continua con backpropagation y ajustando con el decaimiento.

Ajuste de parámetros: Otros

- Numero de épocas
  - Para evitar el sobre entrenamiento y el tiempo computacional necesario para entrenar la red, se puede fijar un cierto numero de épocas de entrenamiento de acuerdo al comportamiento observado del error de entrenamiento y de prueba.
- Entrenamiento con ruido
  - Se puede dar el caso que sea necesario agregar ruido a las observaciones de entrenamiento de manera de entregar una mayor generalidad al modelo.
- Función de activación
  - Una red neuronal MLP entrenada con el algoritmo backpropagation entrena generalmente más rápido si se utiliza una función de activación anti-simétrica ($f(-x) = -f(x)$)

Ajuste de parámetros: Otros (2)

- Tasa de aprendizaje.
  - Se recomienda utilizar una combinación de tasas de aprendizaje sobre distintas redes.
  - Este parámetro, a grandes rasgos, permite definir la velocidad por sobre la cual se va acercando al óptimo del problema de optimización definido sobre una red neuronal artificial.
- Momentum.
  - Se puede incluir un parámetro llamado "momentum" utilizado para la actualización de los pesos en el algoritmo de backpropagation.
  - Permite considerar la "cantidad de movimiento" que cada peso tiene al irse actualizando.

$$\Delta w_{i,j}(n) = \eta \delta_j x_{i,j} + \alpha \Delta w_{i,j}(n-1)$$

Momentum

- No existe una regla general para los valores de ambos parámetros, pero para el momentum se recomiendan valores cercanos a 0.9.
- Tasas de aprendizaje pequeñas (0,3 – 0,4) aumentan convergencia, pero ralentizan entrenamiento.

Condiciones Iniciales

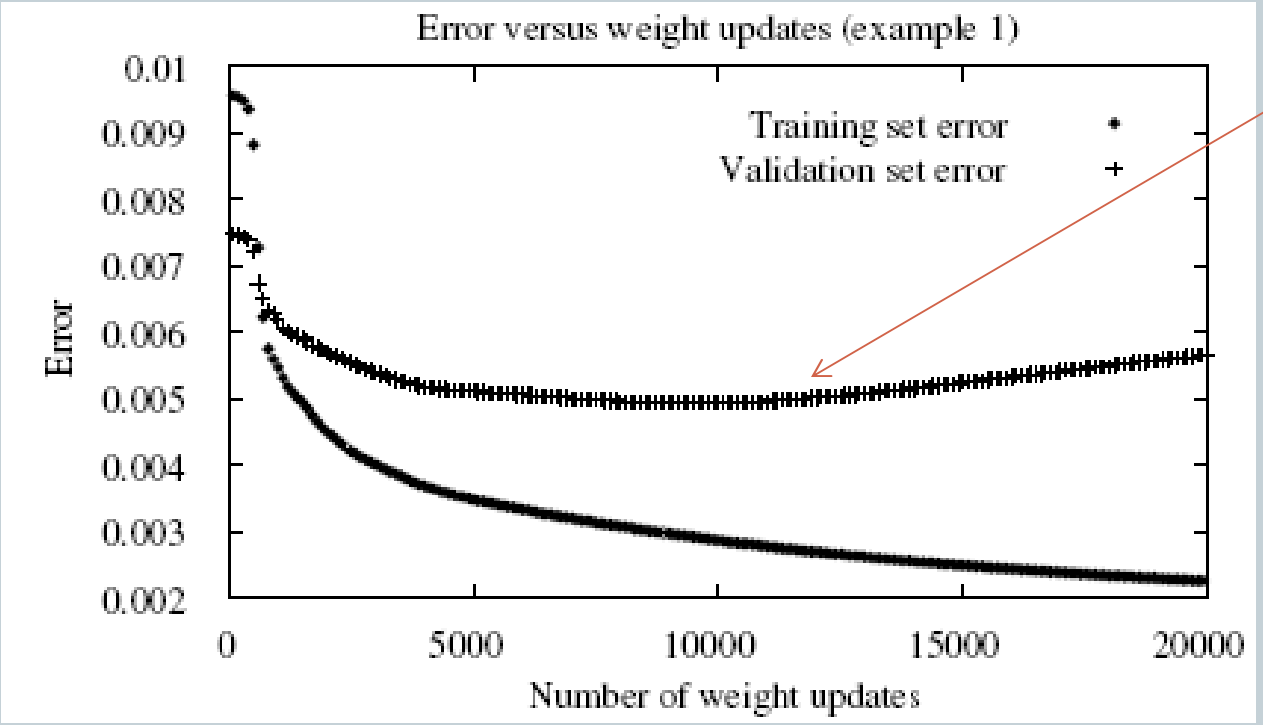
- Preprocesamiento de datos de entrada
  - Los datos de entrada deben estar pre-procesados de manera que su media sea cero, o un valor muy bajo con respecto a la varianza.
  - Los datos de entrada no deben estar correlacionados.
    - Intentar de utilizar variables que no expliquen las mismas características de las observaciones de una base de datos.
    - Una alternativa es utilizar PCA u otros métodos que aseguren variables independientes.
  - Las variables de entrada deben tener una varianza similar.
  - Las variables de entrada deben estar usualmente en la misma escala.
- Pesos inícials
  - Pesos inícials deben ser valores pequeños para evitar la “saturación” de las neuronas.
    - Valores de los pesos de “entrada-capa-media” son mayores que pesos “capa-media-salida” dado que actualizan sus valores con los errores de backpropagation.

Actualización de los Pesos

- Existen dos aproximaciones básicas para actualizar los pesos durante el entrenamiento de la red:
  - Entrenamiento "*On-line*": Pesos son actualizados con backpropagation luego que cada observación es presentada a la red neuronal.
  - Entrenamiento "*Batch*": Pesos son actualizados una vez que todas las observaciones son presentadas a la red neuronal.
- Se recomienda el entrenamiento "*Batch*" dado que se utiliza la "*steepest*" dirección de descenso, la cual es la más adecuada para el problema de optimización no lineal que se desea resolver.
- Entrenamiento "*On-line*" solo entrega una menor complejidad computacional del entrenamiento en un orden.
- Entrenamiento "*On-line*" es sensible al orden que se presentan las observaciones a la red neuronal.

Overfitting en Redes

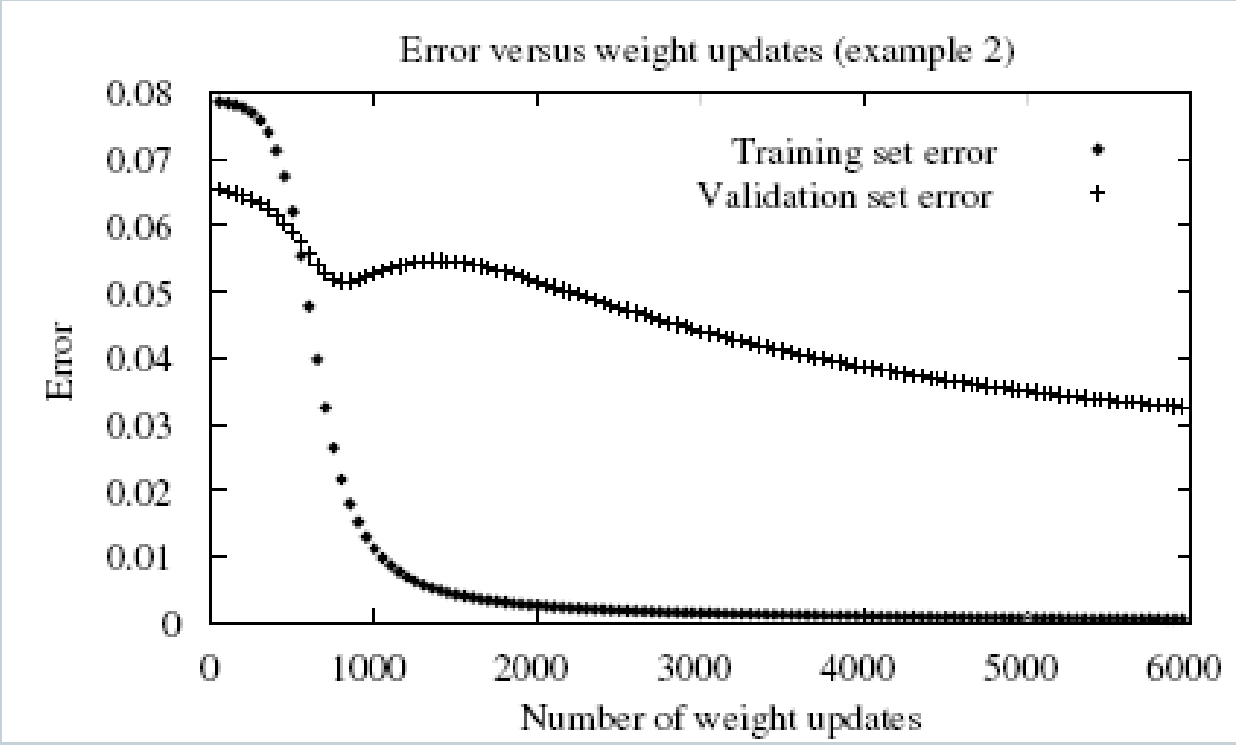
- Sobre entrenamiento en Redes Neuronales



[Fuente: T.Mitchell, Machine Learning]

Overfitting en Redes (2)

- Entrenamiento NO es lineal.



[Fuente: T.Mitchell, Machine Learning]

Ventajas Redes Neuronales

- Algoritmo MUY potente. Permite aproximar cualquier función.
  - Robusto frente a muchas clases y atributos de entrada.
- Flexible:
  - Datos de entrada deben ser preprocesados, pero bajo condiciones poco restrictivas.
  - Funciones de transferencia y de salida entregan múltiples salidas.
  - Parámetros permiten adecuar aún más resultado esperado.
- No linealidad entrega aplicabilidad amplia.

Desventajas Redes Neuronales

- Algoritmo muy complejo.
  - Alcanzar convergencia óptima es muy difícil.
  - Complejidad computacional exponencial (depende del número de atributos).
    - Empíricamente, ha resultado ser cúbica en la gran mayoría de los problemas.
- Gran cantidad de parámetros a explorar.
  - Requiere experiencia y mucho ensayo y error.
- Flexibilidad es una espada de doble filo.
  - Gran cantidad de formas de errar.