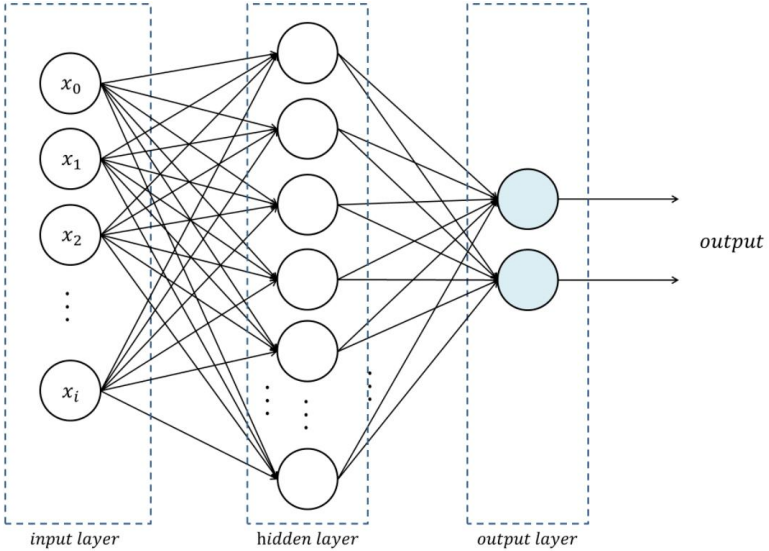
Módulo 3: Aprendizaje Supervisado

3.3. Redes Neuronales (Parte II)

Rafael Zambrano
rafazamb@gmail.com

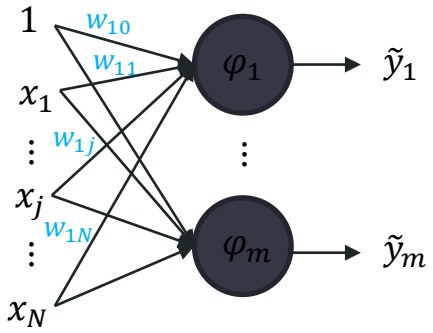
Dimensionando las Redes Neuronales



- Red sub-dimensionada
  - No tiene capacidad de representar el mapeo $x \rightarrow y$
  - No aprende, el error se mantiene elevado
- Red sobredimensionada
  - Lenta
  - Tendencia al overtting

- ¿Cuántas capas de neuronas?
- ¿Cuántas neuronas por capa?
- ¿Qué tipo de neurona?

Redes de una capa

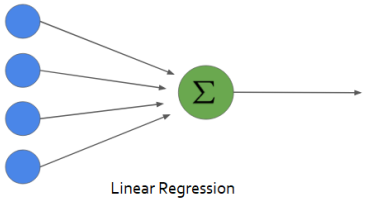


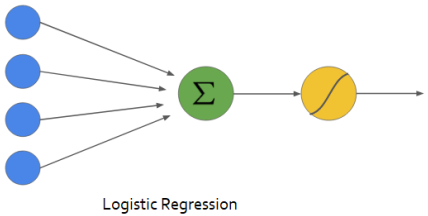
- Si las neuronas son lineales, la red es un regresor lineal

$$\tilde{y}_1 = \sum x_j w_{1j} + w_{1o} = w_{10} + w_{11}x_1 + \cdots + w_{1j}x_j + \cdots + w_{1N}x_N$$

- Si las neuronas son de tipo tgh(), la salida es un regresor lineal con salida limitada (empleada en clasificadores)

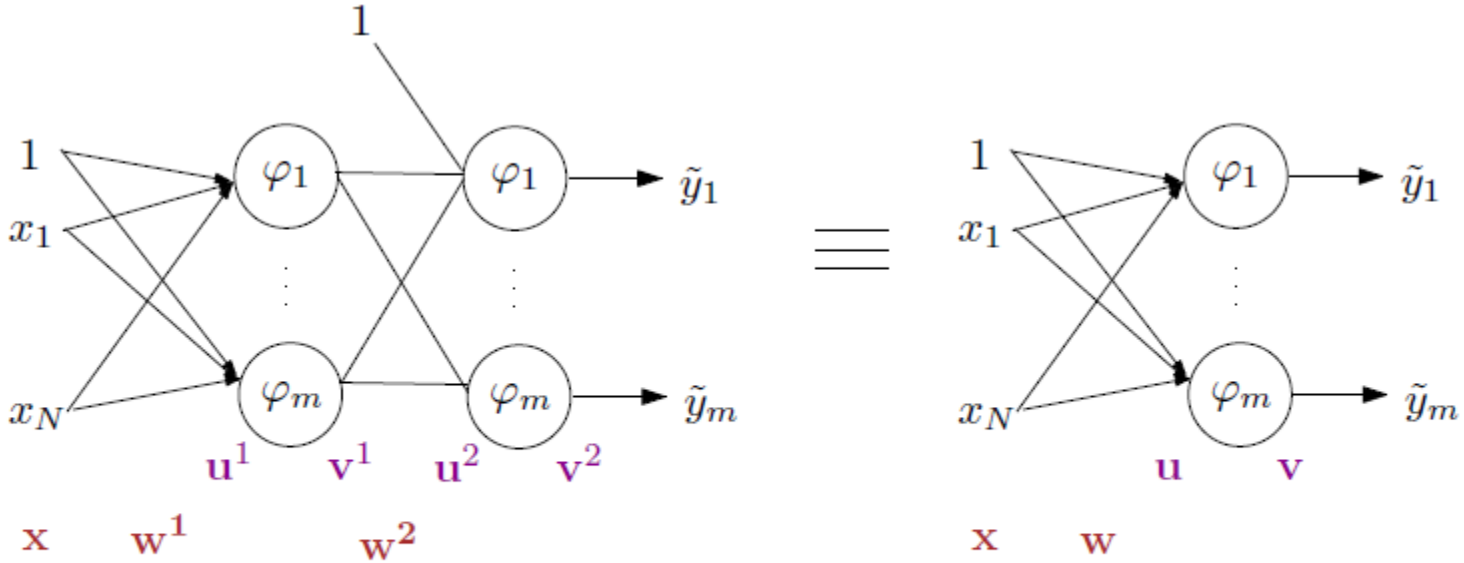
$$\tilde{y}_1 = \text{tgh}(\sum x_j w_{1j} + w_{1o})$$





Redes de dos capas

- Si todas las neuronas son lineales, la red es idéntica a una red con una sola

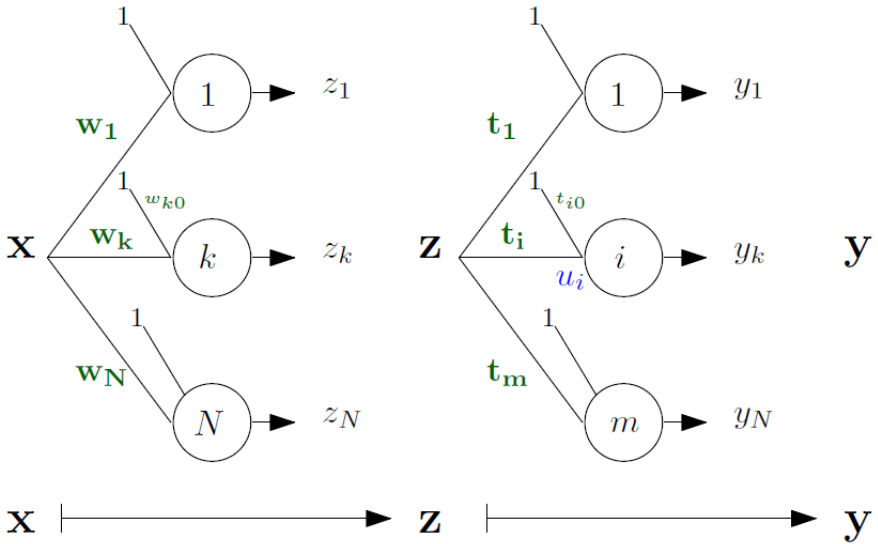


$$\mathbf{u}^2 = \mathbf{w}^2\mathbf{v}^1 = \mathbf{w}^2\mathbf{w}^1\mathbf{x}$$

$$\mathbf{u} = \mathbf{w}\mathbf{x} = \mathbf{w}^2\mathbf{w}^1\mathbf{x}$$

Redes de dos capas

- La capa intermedia tiene neuronas de tipo tgh()
- Teorema: Si una funcion es L^2 ($\int |f|^2 < \infty$) en un dominio, entonces una serie de tangentes hiperbólicas es un aproximador universal para la función en el dominio.
- Una red neuronal con dos capas, la primera con neuronas de tipo tgh() y la segunda con neuronas lineales es un aproximador universal para todas las funciones de interés práctico



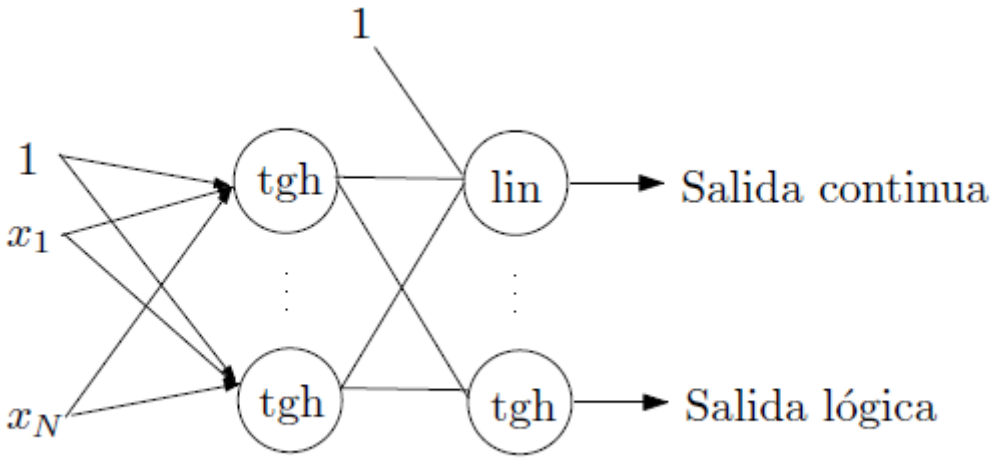
$$z_k = tgh(\mathbf{x}^T \mathbf{w_k} + w_{k0})$$

$$u_i = \mathbf{z}^T \mathbf{t_i} + t_{i0} = t_{i0} + \sum_k t_{ik} tgh(\mathbf{x}^T \mathbf{w_k} + w_{k0})$$

$$\tilde{y}_i = \begin{cases} u_i & \text{Neurona de salida lineal} \\ tgh(u_i) & \text{Neurona de salida tgh (usada en clasificadores)} \end{cases}$$

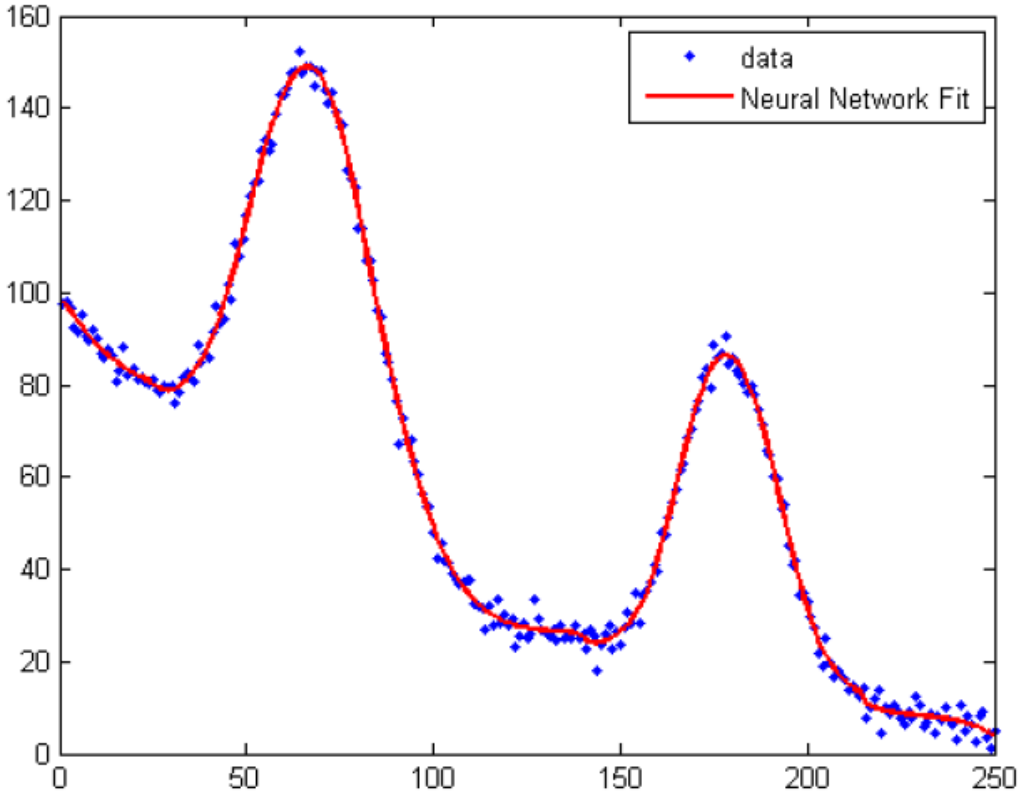
$$y_i \in \{-1, 1\}$$

Redes de dos capas



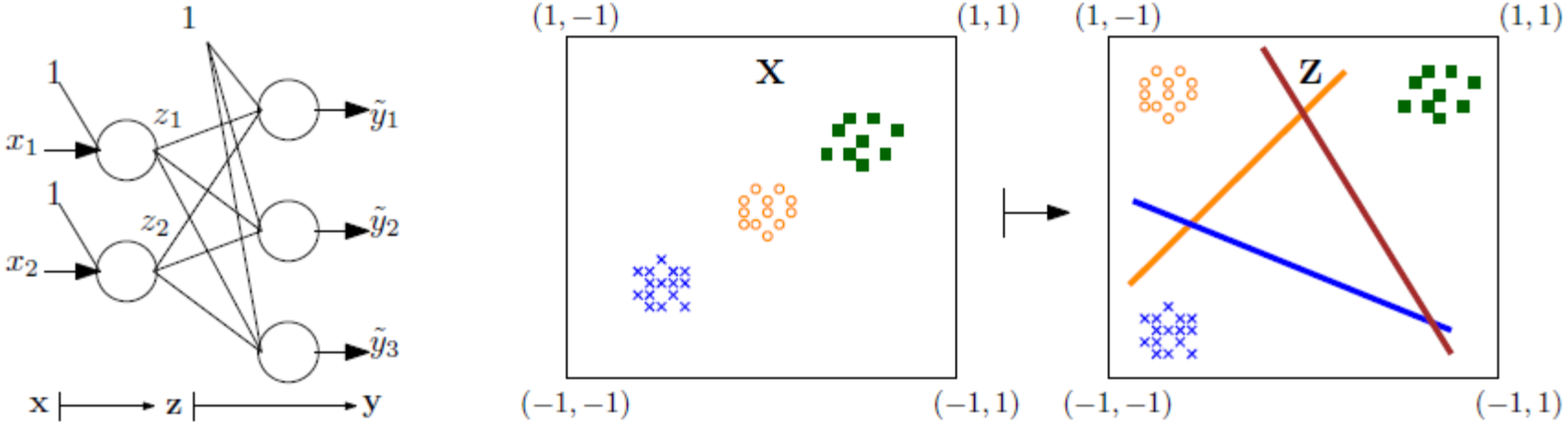
Redes de dos capas (regresión)

$$\tilde{y}_i = t_{i0} + \sum_k t_{ik} tgh(\mathbf{x^T w_k} + w_{k0})$$



Redes de dos capas (clasificación)

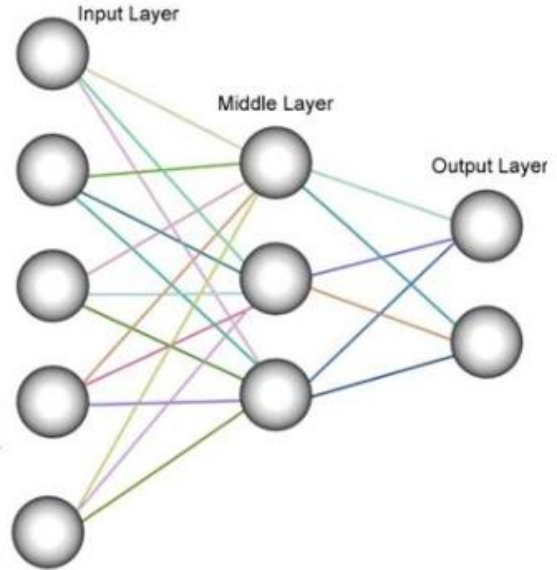
$$\tilde{y}_i = tgh(t_{i0} + \sum_k t_{ik} tgh(\mathbf{x}^{\mathbf{T}}\mathbf{w_k} + w_{k0}))$$



La capa de salida separa clases que sí son linealmente separables en el dominio **z**
La capa intermedia mapea clases no linealmente separables de **x** en clases linealmente separables en **z**

Dimensionando las Redes Neuronales

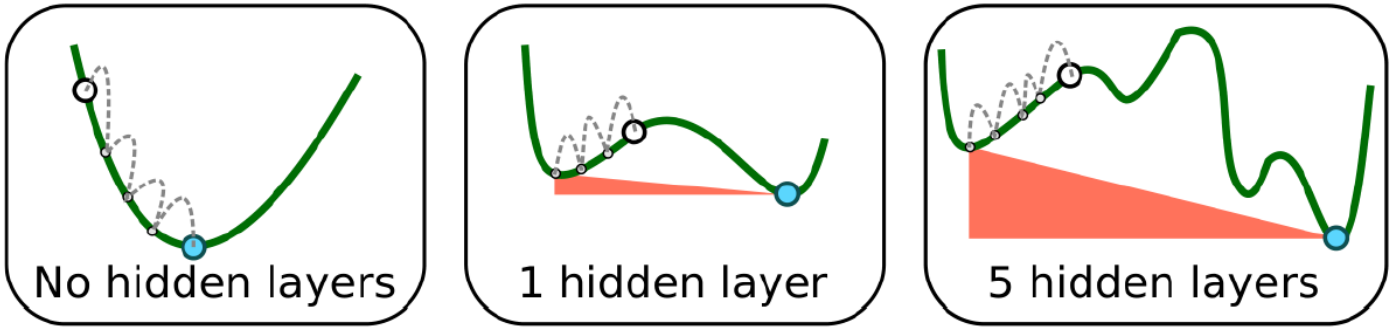
- Usar una o dos capas
- En la capa de salida
  - Regresión (*y* es continuo): neurona lineal (y = u)
  - Clasificación (*y* es categórico): neurona tgh (y = tgh(u))
- En la capa intermedia usar siempre neuronas de tipo tgh
- **Redes de una capa**
  - Si el error es alto, utilizar dos capas
- **Redes con dos capas**
  - Numero de neuronas en la capa intermedia: m_1
  - Regla general: $n > m_1 > m$
  - Si el error es grande, aumentar m_1
- Redes de 3 o mas capas no son necesarias.



Dimensionando las Redes Neuronales

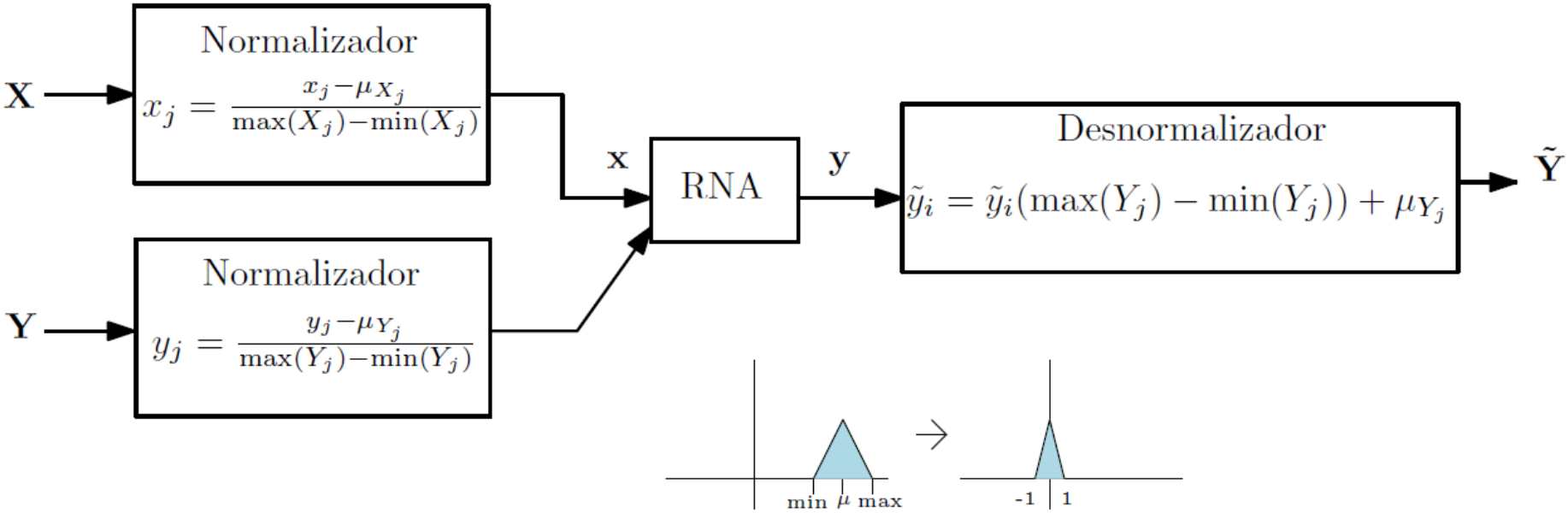
- El algoritmo del gradiente descendente puede finalizar en un mínimo local de la función de error

- Cuando introducimos más no-linearidad en la red (más capas), se multiplican los mínimos locales

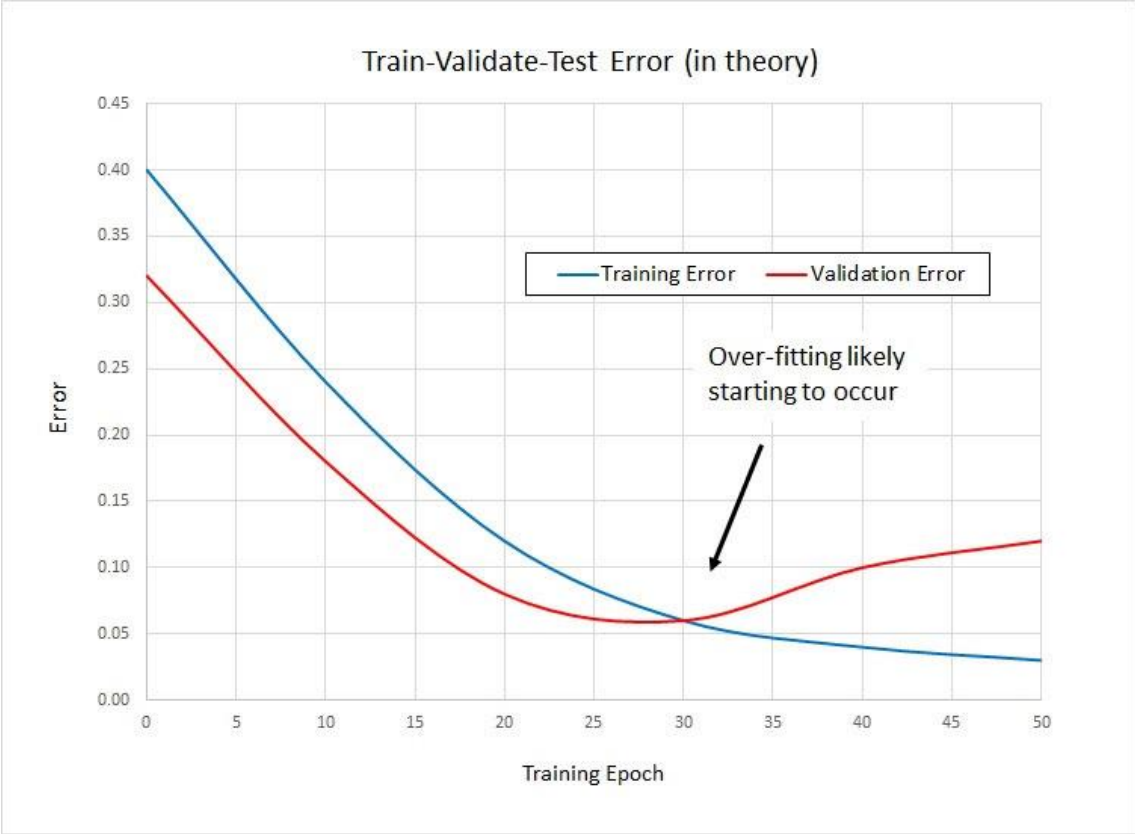


Normalización de variables

- Fundamental para el buen condicionamiento del proceso numérico de optimización
- Queremos valores en el intervalo (-1,+1)



Entrenamiento en Redes Neuronales



Redes Neuronales en R

- En R, existen varias librerías que permiten utilizar redes neuronales. El paquete “neuralnet” es uno de ellos

https://cran.r-project.org/web/packages/neuralnet/neuralnet.pdf

- Entrenamiento (Regresión):

```
modelo <- neuralnet(formula,data=train,hidden=m1,linear.output=T)
```

- Entrenamiento (Clasificación):

```
modelo <- neuralnet(formula,data=train,hidden=m1,linear.output=F,act.fct="tanh")
```

- Predicción

```
compute(modelo,test)
```

¡Gracias!

Contacto: Rafael Zambrano

rafazamb@gmail.com