# EVALUACIÓN #3a

# SISTEMAS DE DETECCIÓN DE INSTRUSOS

# USANDO

# TÉCNICAS DE MACHINE LEARNING

Prof. Nibaldo Rodríguez A.

## OBJETIVO GENERAL

- **Implementar y evaluar un sistema de detección de intrusos usando técnicas de machine learning**

## OBJETIVO ESPECÍFICOS

1. **Calibrar los pesos de salida de una red neuronal artificial usando el algoritmo de la pseudo-inversa Moore-Penrose.**
2. **Calibrar los pesos ocultos de una red neuronal artificial usando un algoritmo de optimización de enjambres de partículas.**
3. **Evaluar el rendimiento del IDS usando las métricas de F-score de cada categoría (normal/ataque).**

# MACHINE LEARNING (ML)

Prof. Nibaldo Rodríguez A.

# MACHINE LEARNING (ML)

**DEFINICIÓN:**

Capacidad de extraer **Patrones** desde la **Data** para generar su propio conocimiento y tomar decisiones.

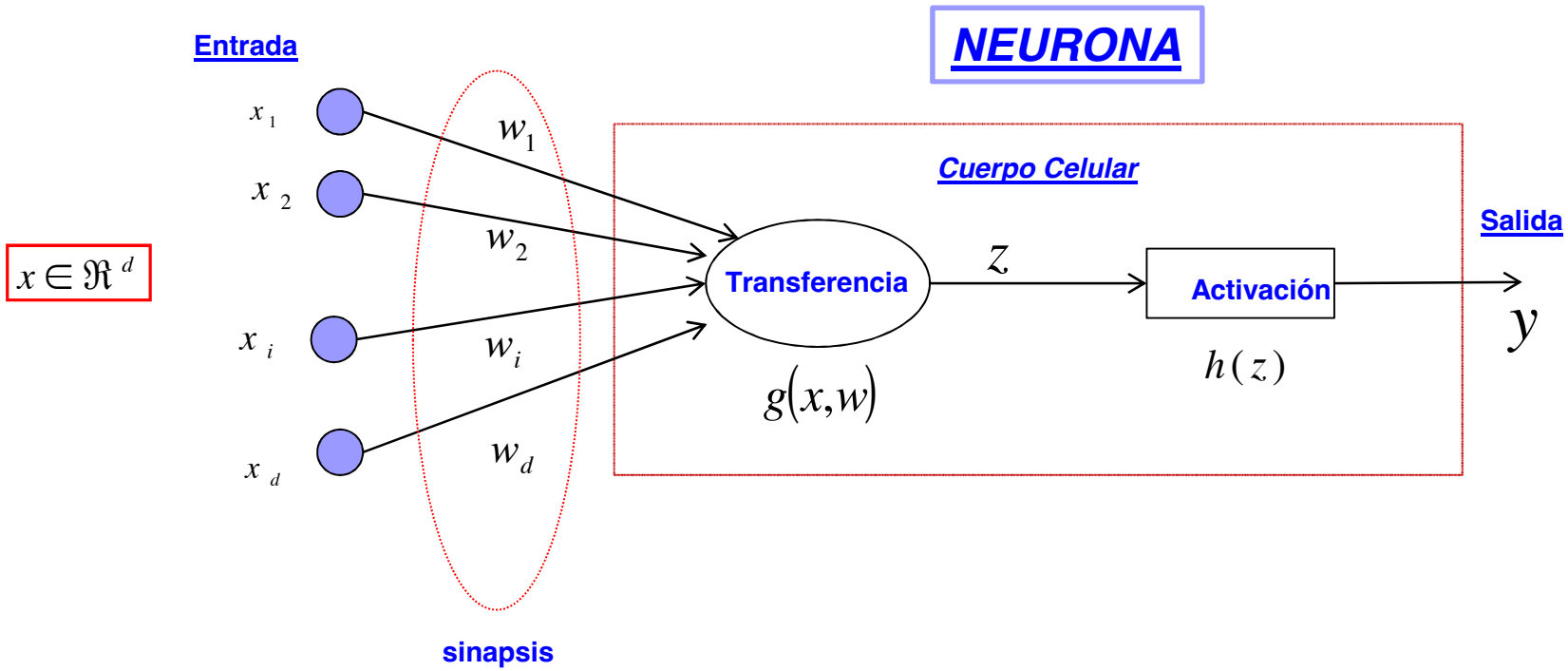
**Patrones:**
Información Relevante dada por un experto o por un algoritmo inteligente

**Algoritmo Inteligente:**
Capacidad para Hallar una Relación Funcional entre los patrones y diversos posibles resultados

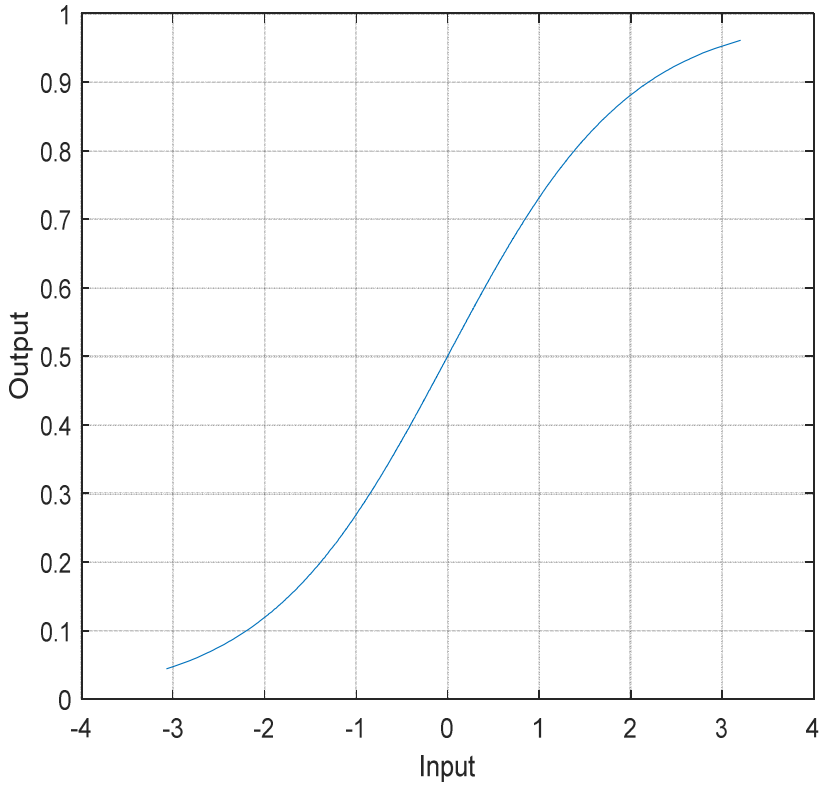
# ARTIFICIAL NEURAL NETWORK (ANN)

Prof. Nibaldo Rodríguez A.

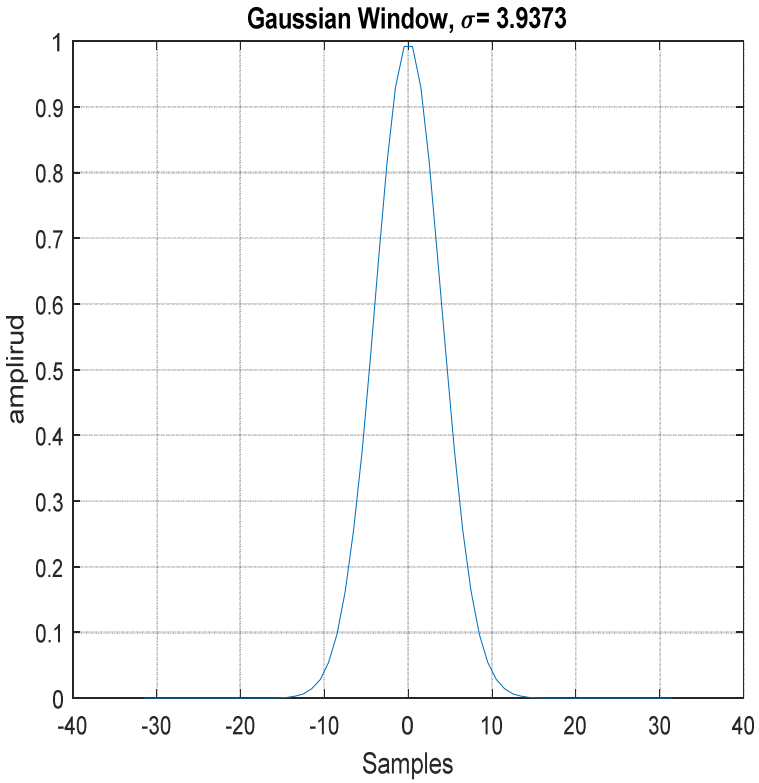
# Unidad Básica de una ANN



# Función Activación

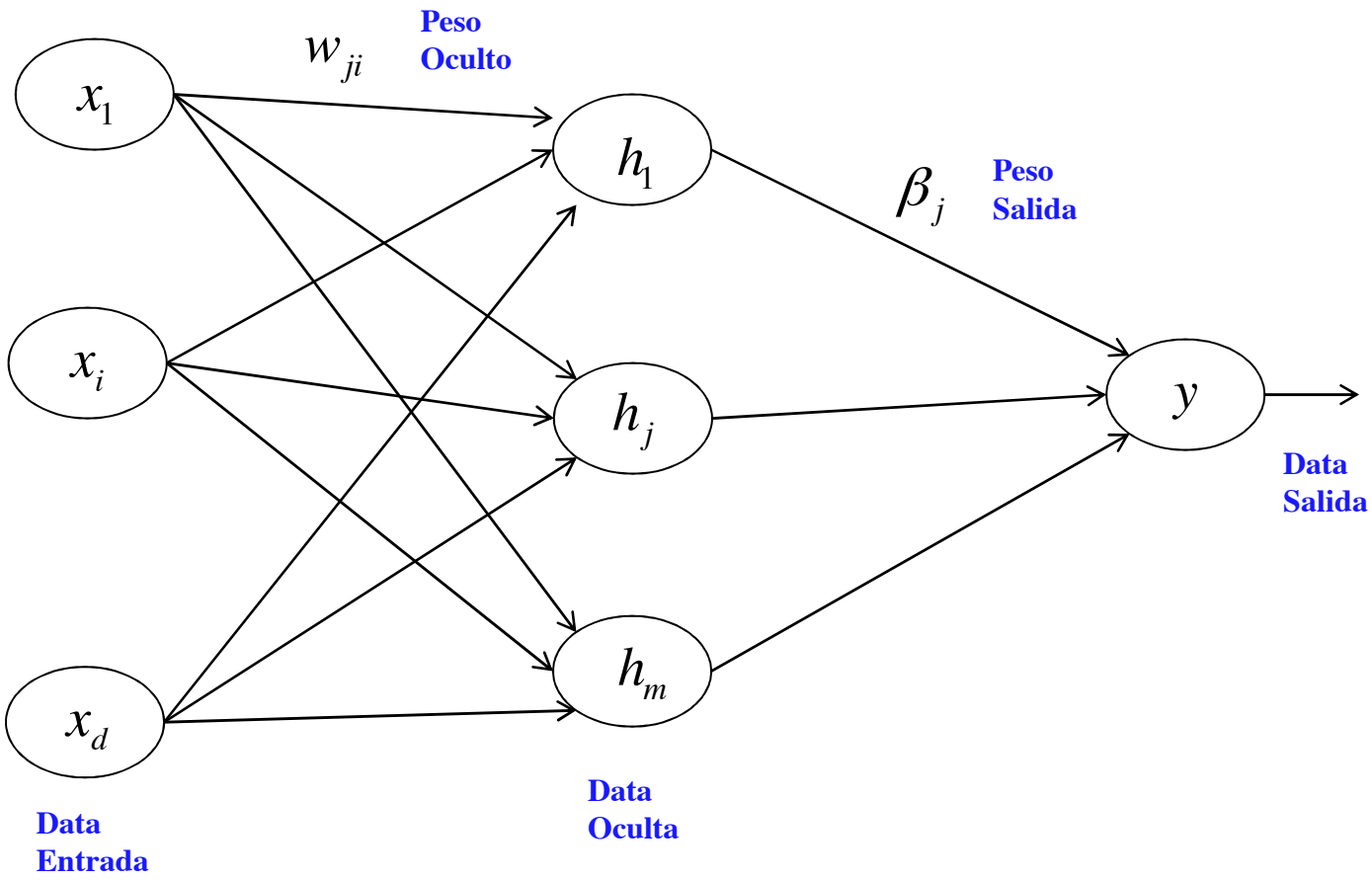



$$F\ (x\ ) = \frac{1}{1\ +\ e^{\ -\ x}}$$




$$F(x) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left( -0.5 \frac{x^2}{\sigma^2} \right)$$

# Topología de una ANN



# FASE #1: Forward de la ANN

- **Step 1:** Activación Capa Oculta

- d: nodos de entradas
- m: nodos ocultos
- N: Número de muestras

$$X \in \Re^{(d \times N)}$$

$$w \in \Re^{(m \times d)}$$

$$H = h(w, X), \; H \in \Re^{(m \times N)}$$

- **Step 2:** Activación Capa Salida

- Un nodo de salida

$$\beta \in \Re^{(1 \times m)}$$

$$y = \beta \times H, \; y \in \Re^{(1 \times N)}$$

**Función de Activación**

$$h(w, X) = \; ?$$

# Red Neuronal Artificial (ANN)

## 1.- Función de Transferencia y Activación

$$h(w, X) = \frac{1}{1 + \exp\left[-(w \times X)\right]}$$

# Red Neuronal Artificial (ANN)

## 2.- Función de Transferencia y Activación

$$h(w, X) = \frac{e^{(w \times X)} - e^{(-w \times X)}}{e^{(w \times X)} + e^{(-w \times X)}}$$

## 3.- Función de Transferencia y Activación

$$h(w, X) = \exp\left(-\frac{1}{2}\|X - w\|^2\right)$$

## 4.- Función de Transferencia y Activación

$$h(w, X) = \left(1 - \|x - w\|\right) \times \exp\left(-\frac{1}{2}\|x - w\|^2\right)$$

## 5.- Función de Transferencia y Activación

$$h(w, X) = \cos\left(5 \times \|X - w\|\right) \times \exp\left(-\frac{1}{2}\|X - w\|^2\right)$$

## 6.- Función de Transferencia y Activación

$$h(w, X) = \left(1 + \|x - w\|\right)^{-\frac{1}{2}}$$

## 7.- Función de Transferencia y Activación

$$z = w \times X$$

$$h(w, X) = \max \{0.1 \times z, z\}$$

# FASE #2: Entrenar la SNN

**APRENDIZAJE SUPERVISADO:**
**(calibración de los pesos)**

# Optimización de Pesos de Salida:
## Pseudo-inversa Moore-Penrose

$$\beta = \left( H \times H^T + \frac{I}{C} \right)^{-1} H \times Y^T$$
$$C \in \left[ 10 , 10^8 \right]$$

- **H =** matriz de data de los nodos ocultos.
- **I =** matriz de identidad de tamaño **m** .
- **Y =** datos deseado (target, categoría).
- **()$^T$** = transpuesta de una matriz.
- **()$^{-1}$**= matriz pseudo-inversa.
- **C** = parámetro de penalidad de pseudo-inversa.

# Optimización Pesos Ocultos.

1. Particle Swarm Optimization (PSO)

2. Quantum Particle Swarm Optimization (QPSO)

# *Particle Swarm Optimization* (PSO)

## ENJAMBRE

Mejor Partícula Individual: $p = [w_1 . w_2 \ldots . , w_D ], \; p \in \Re^{(1 \times D)} , D = (m \times d)$

Mejor Partícula Global: $p_g = [w_1 . w_2 \ldots . , w_D ], \; p_g \in \Re^{(1 \times D)}$

Enjambre:
Np: Número de partículas

$$X = \begin{bmatrix} w_{1,1} & w_{1,2} & \ldots . & \ldots & w_{1,D} \\ w_{2,1} & w_{2,2} & .. & .. & w_{2,D} \\ .. & .. & .. & .. & . \\ .. & .. & .. & .. & .. \\ w_{N_p,1} & w_{N_p,2} & \ldots & \ldots & w_{N_p,D} \end{bmatrix}$$

# *Particle Swarm Optimization* (PSO)

## ENJAMBRE INICIAL

Generar partículas aleatorias: $X(i,:) = w, \quad i = 1,\ldots,N_p$

$$w = rand\ (m,d) \times 2 \times r - r$$

$$r = \sqrt{\frac{6}{m+d}}$$

# *Particle Swarm Optimization* (PSO)

**Actualización del Enjambre**

Iteración: *k*-ésima

$$X(k+1) = X(k) + V(k+1)$$

$$V = velocidadd$$

# *Particle Swarm Optimization* (PSO)

## VELOCIDAD

$$V(k+1) = \alpha \times V(k) + c_1 r_1 \left[P(k) - X(k)\right] + c_2 r_2 \left[p_g(k) - X(k)\right]$$

**Inercia:** $\alpha \in (0,1)$ $\quad c_1 = 1.05 \quad c_2 = 2.95 \quad r_1, r_2 \in rand(0,1)$

## INERCIA

$$\alpha = \alpha_{max} - \left( \frac{\alpha_{max} - \alpha_{min}}{IterMax} \right) \cdot IterActual \qquad \alpha_{min} = 0.1 \qquad \alpha_{max} = 0.95$$

## Partículas Inicial del Enjambre

$$P(0), V(0) = 0 \in \Re^{(N_p \times D)}, \; p_g(0) = 0 \in \Re^{(1 \times D)}$$

## *Quantum* –PSO (QPSO)

### ENJAMBRE INICIAL

Generar partículas aleatorias: $X(i,:) = w, \quad i = 1,\ldots,N_p$

$$w = rand\ (m,d) \times 2 \times r - r$$

$$r = \sqrt{\frac{6}{m+d}}$$

# *Quantum* –PSO (QPSO)

## NUEVO ENJAMBRE

Iteración: *k*-ésima

$$X_{ij} = \begin{cases} P_{i,j}(k) + \alpha \cdot \left| mBest_j - X_{ij} \right| \cdot Ln\left(\frac{1}{\mu}\right) & Si \quad rand > 0.5 \\ P_{i,j}(k) - \alpha \cdot \left| mBest_j - X_{ij} \right| \cdot Ln\left(\frac{1}{\mu}\right) & \text{o.c} \end{cases}$$

## Partículas del Enjambre

$$P_{i,j}(k) = \varphi \times P_{i,j}(k-1) + (1-\varphi) \times p_{g,j} \qquad \varphi, \mu \in rand(0,1)$$

$$P(0) = 0 \in \Re^{(N_p \times D)}, \quad p_g(0) = 0 \in \Re^{(1 \times D)}$$

# *Quantum* –PSO (QPSO)

$$mbest_j = \frac{1}{N_p}\sum_{i=1}^{N_p} p_{i,j}, \qquad j = 1,\ldots,D$$

$$\alpha = (b-a)\left[\frac{MaxIter - iter}{MaxIter}\right] + a, \quad a = 0.2 \quad b = 0.95$$

# Training para PSO-QPSO

**Minimizar Función de Costo: MSE**

$$E = \frac{1}{N}\sum_{n=1}^{N} e^2(n)$$

$$e(n) = d(n) - y(n),$$

$d(n)$ : valor deseado $\qquad y(n)$ = valor de salida de la red

$N$ = Num. muestras de training

# Evaluación de Rendimiento

## Matriz de Confusión

| | | Valor Real | |
|---|---|---|---|
| | | P | N |
| PREDICHO | P | VP | FP |
| | N | FN | TN |

$$P = \frac{VP}{VP + FP}$$

Precision (precisión)

$$R = \frac{VP}{VP + FN}$$

Recall (sensibilidad)

$$F = 2 \times \frac{P \times R}{P + R}$$

F-score

$$A = \frac{VP + VN}{VP + FP + VN + FN}$$

Accuracy (exactitud)

# CONTINUARÁ….