

**Israel Emmanuel Jiménez Lopez.**

*Alumno*

**Reconocimiento de patrones.**

*Curso*

**Jesús Alejandro Flores Hernández**

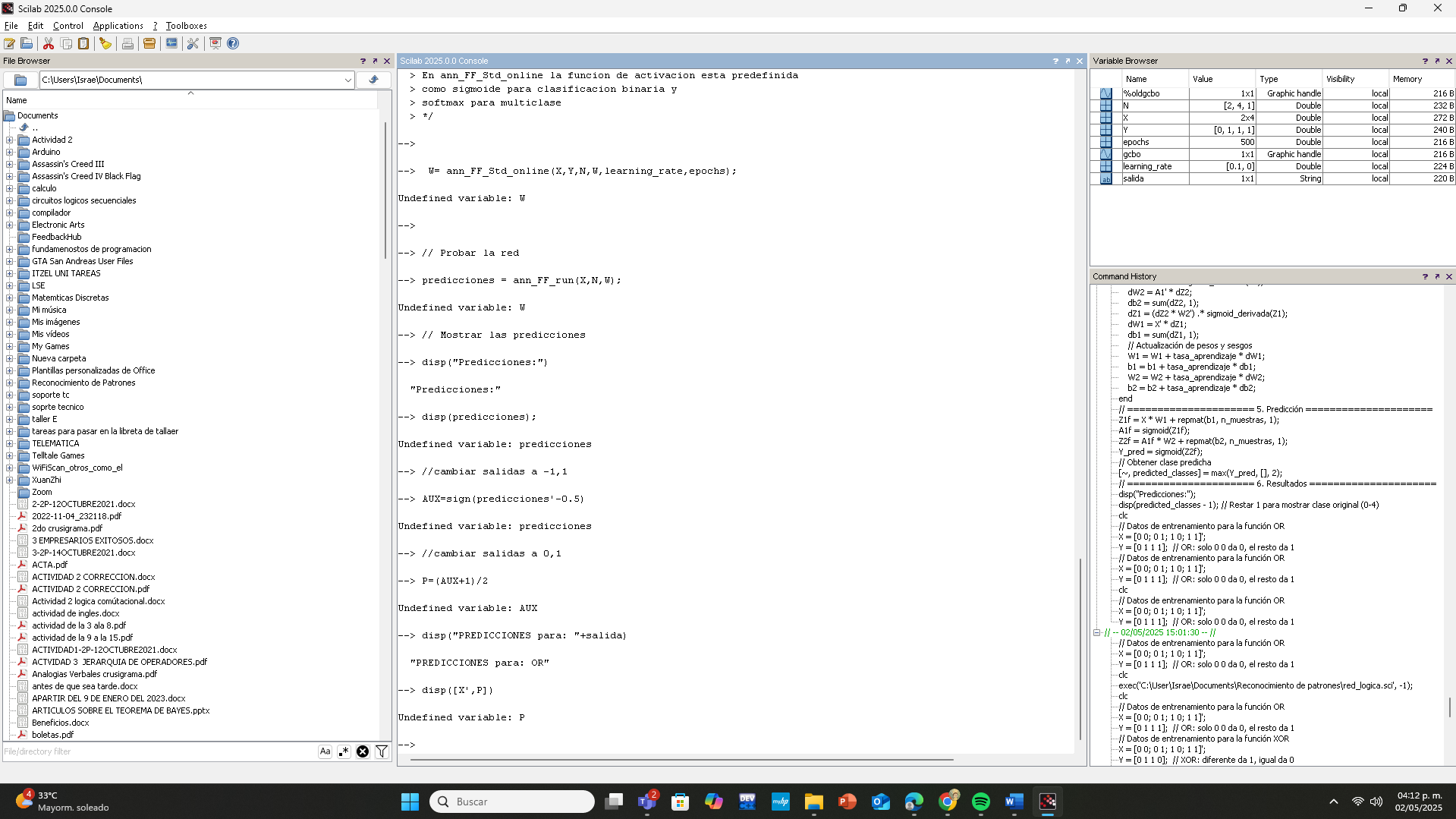
*Profesor*

# actividad Utilizar biblioteca ANN

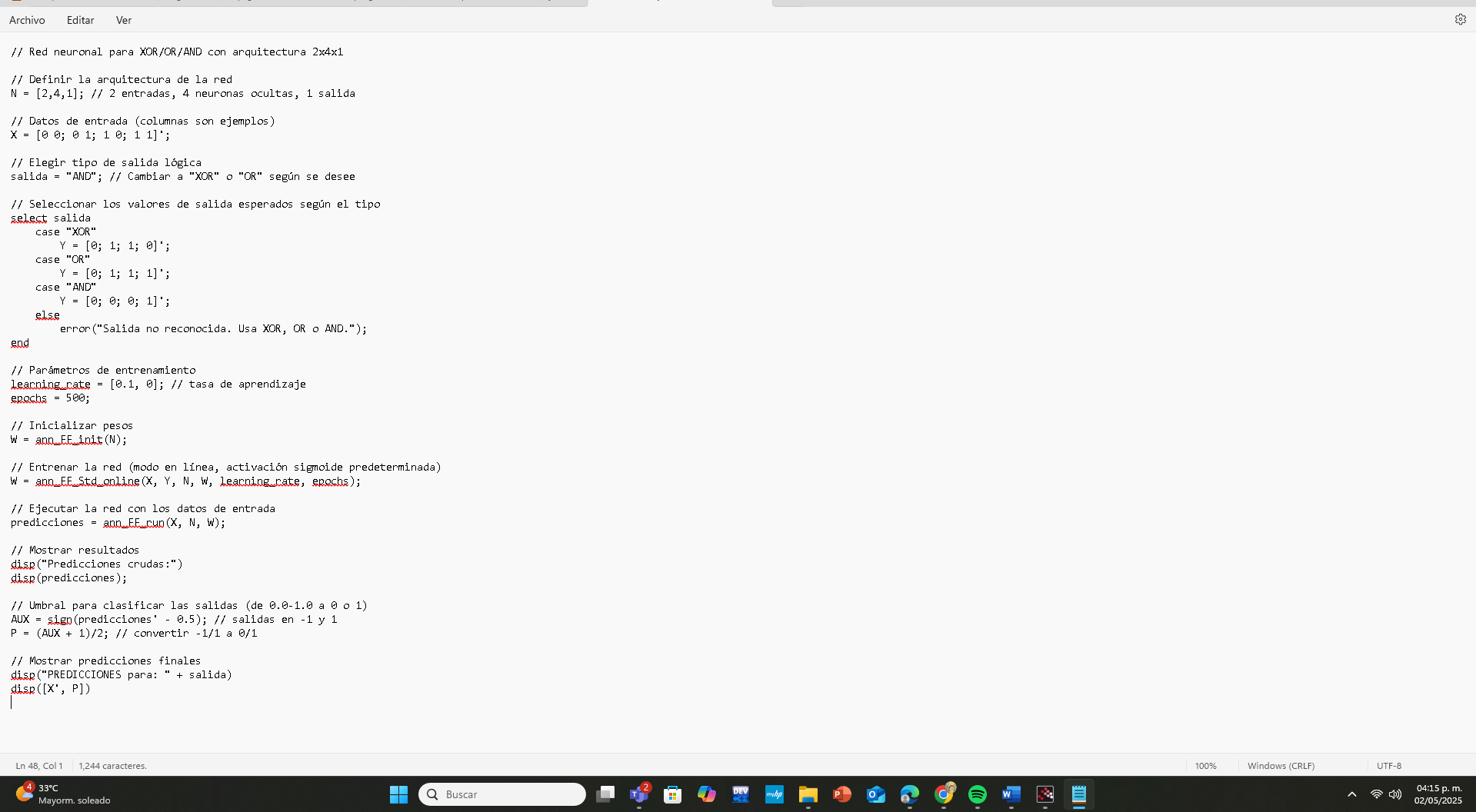
1. Pruebe el código "ANNred2x4x1-OR.sci".
2. Modifique para las salidas de AND y XOR.

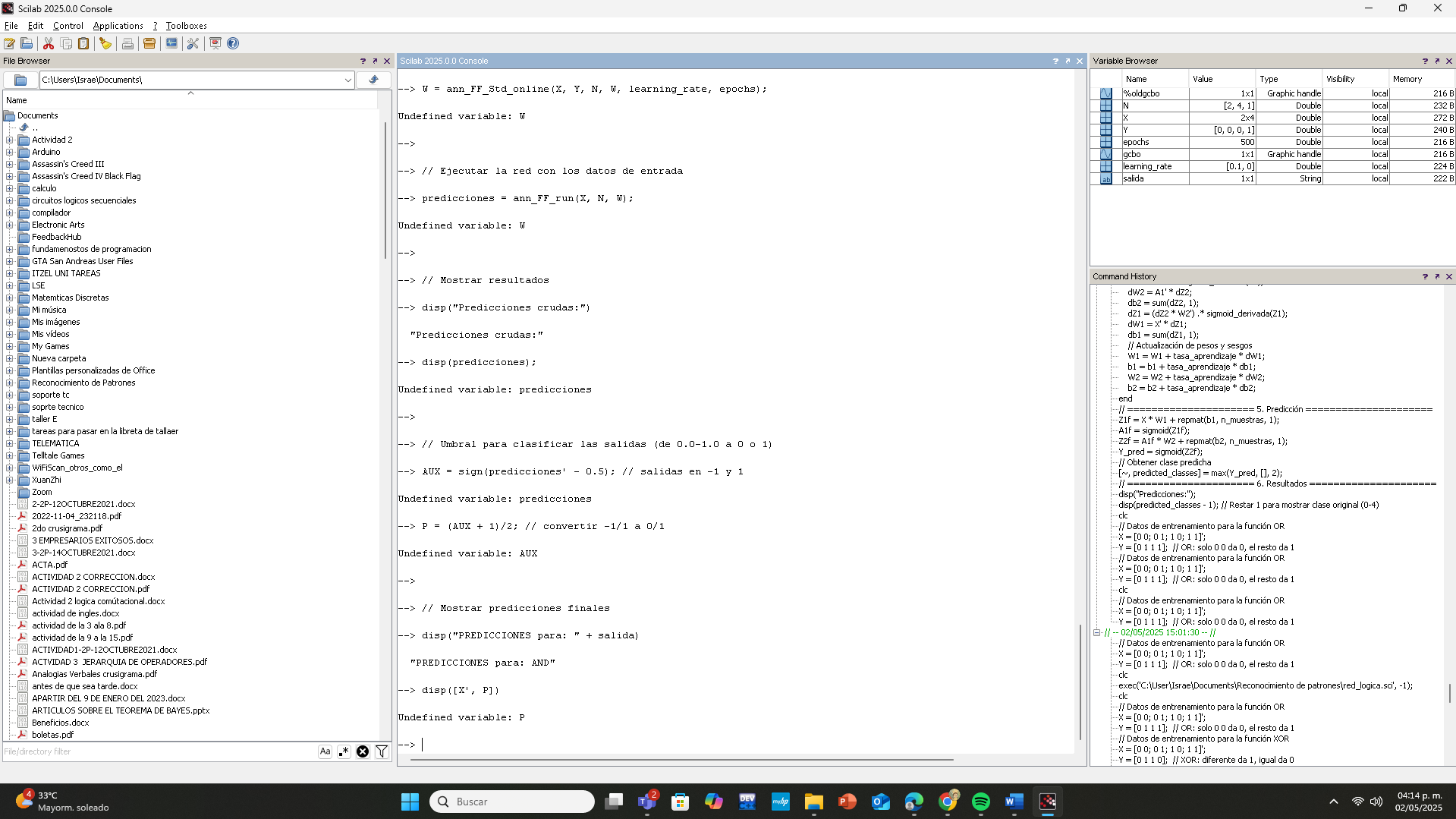
Entregable

1. haga un video que explique el código y sus resultados.
2. suba el video a un repositorio publico y envie la liga al profesor.



Con función  AND y XOR.





**Objetivo del código**

Entrenar una red neuronal en **Scilab** para aprender una función lógica: **AND**, **OR** o **XOR**, usando una arquitectura simple de 2 capas (2x4x1) y evaluar sus predicciones.

|  |  |
| --- | --- |
| Explicación. | Código. |
| * 2 entradas (porque usamos combinaciones binarias de 2 bits). * 4 neuronas en la capa oculta (puedes cambiar esto). * 1 salida (valor entre 0 y 1 que representa la clase). | N = [2,4,1]; |
| Son las **4 combinaciones posibles** de dos valores binarios.  Se transponen (') para que cada **columna** sea un ejemplo de entrada (formato que requiere la toolbox). | X = [0 0; 0 1; 1 0; 1 1]'; |
| * Puedes cambiar este valor para probar otras funciones lógicas. | salida = "AND"; // Cambia por "OR" o "XOR" |
| Según la lógica elegida, se definen las salidas correctas para cada combinación de entrada.  También se transpone (') para que el formato sea 1x4. | select salida  case "XOR"  Y = [0; 1; 1; 0]';  case "OR"  Y = [0; 1; 1; 1]';  case "AND"  Y = [0; 0; 0; 1]';  else  error("Salida no reconocida. Usa XOR, OR o AND.");  end |
| Se inicializan los **pesos aleatoriamente**. | W = ann\_FF\_init(N); |
| ann\_FF\_Std\_online: algoritmo de aprendizaje **en línea** (actualiza los pesos con cada ejemplo).  Usa **función de activación sigmoide** internamente.  Se entrena durante epochs iteraciones. | W = ann\_FF\_Std\_online(X, Y, N, W, learning\_rate, epochs); |
|  Se pasa la entrada por la red ya entrenada.   La salida será un número entre 0 y 1. | predicciones = ann\_FF\_run(X, N, W); |
|  Se aplica un **umbral de 0.5** para convertir la salida continua a binaria.   sign(x-0.5) devuelve -1 para <0.5 y 1 para >=0.5.   Se ajusta a 0 y 1 con (AUX + 1)/2. | AUX = sign(predicciones' - 0.5);  P = (AUX + 1)/2; |
| Muestra las entradas y sus salidas predichas. | disp("PREDICCIONES para: " + salida)  disp([X', P]) |