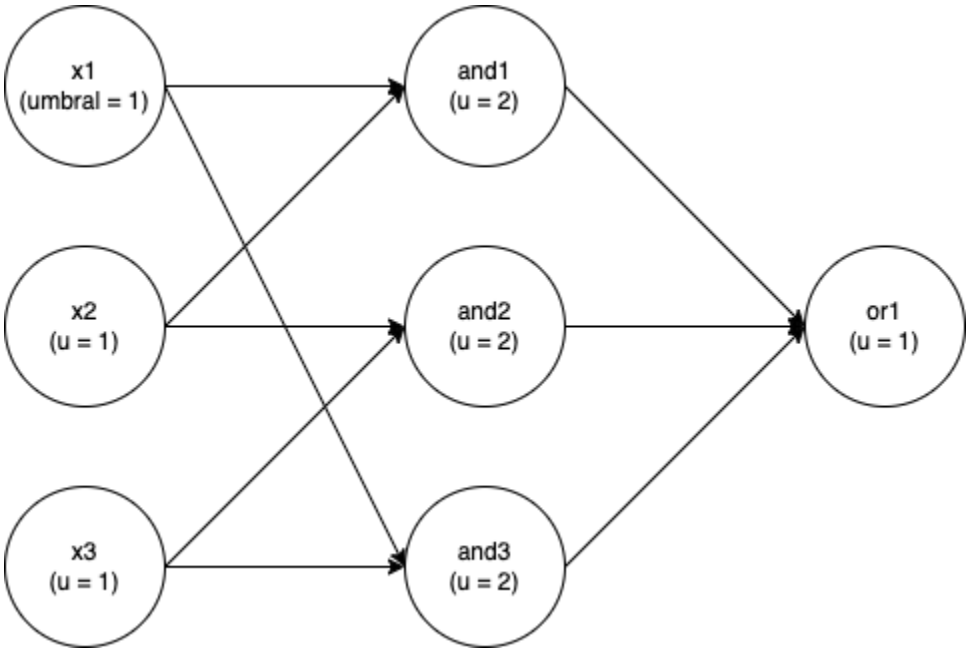
Memoria

2.1.

Incluid el diseño de la red con los sesgos, conexiones y pesos utilizados.



Los pesos de todas conexiones son 1.

Discutid la validez de vuestro diseño e incluid explicaciones de ejemplos que demuestren el correcto funcionamiento interno de cada elemento que conforma el circuito.

Nuestro diseño es válido porque imita las puertas lógicas requeridas. Especialmente para cada neurona de la puerta 'AND', el valor umbral es 2, lo que significa que ambos inputs deben estar activos para activar la neurona. En el caso de la neurona de la puerta 'OR', el valor umbral es 1, lo que significa que si cualquiera de las neuronas de entrada está activa, la neurona 'OR' será activada.

¿Qué calcula la función f(t + 2) para unos $x_1(t)$, $x_2(t)$ y $x_3(t)$ dados?

f(t+2) = (x1(t)^(x2(t)) | (x1(t)^(x3(t)) | (x2(t)^(x3(t))

4.1.1.

Dad las fronteras de decisión de cada algoritmo para los cuatro problemas lógicos en la forma $w_1 \cdot x_1 + w_2 \cdot x_2 + b = 0$.

Las fronteras de perceptron

	w_1	w_2	b
and	2.0	3.0	-4.0
nand	-3.0	-2.0	4.0
or	2.0	2.0	-1.0
xor	0.0	0.0	0.0

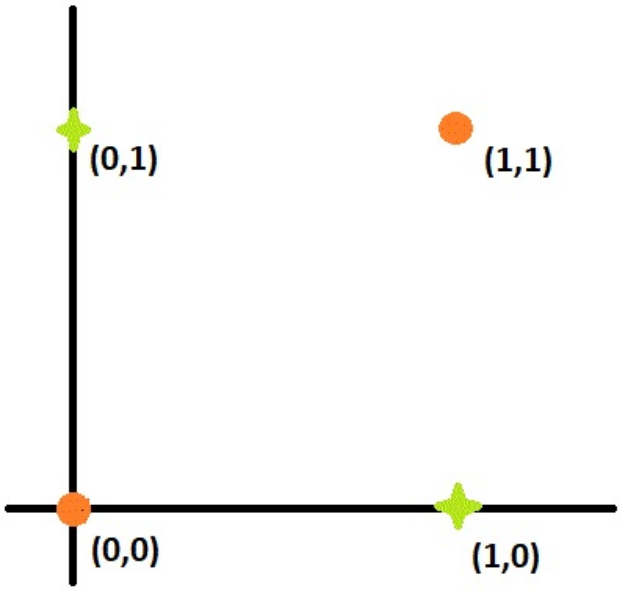
Las fronteras de adaline

	w_1	w_2	b
and	2.0	4.0	-6.0
nand	-4.0	-2.0	4.0
or	2.0	2.0	-2.0
xor	0.0	0.0	0.0

¿Es posible solucionar todos los problemas? En caso de no ser posible: ¿cuál es el problema que no es posible? ¿cómo podría solucionarse?

Es posible resolver todos los problemas linealmente separables sin capas ocultas, por lo tanto, todos los problemas lógicos excepto el xor son solucionables.

Xor no es solucionable porque su gráfica no es linealmente separable:



Esto puede solucionarse añadiendo una capa oculta.

4.2.1.

Describid la implementación de ambas redes neuronales.
El perceptron, siendo monocapa, tiene dos capas input y output. El input viene inicializado con los valores del conjunto de datos y la función propagar se utiliza para calcular el nuevo valor de las neuronas de salida. Después de propagar la función de activación "disparar" viene aplicada sobre las neuronas de salida para convertir los valores en -1/0/1.
Después de calcular y_in para un par x, t se cambian los pesos con la función:
w_nuevo = w_anterior + αtx

La formula cambia también el peso del sesgo, dado al hecho que el valor de las neuronas de sesgo es siempre 1 entonces

b_nuevo = b_anterior + αtx = b_anterior + αt

El adaline tiene un funcionamiento similar pero la función de activación y el cambio del peso es diferente. La activación es binaria con $\mathbf{f(y)} \in \{-1,1\}$
y el cambio de peso es definido con

w_nuevo = w_anterior + α(t-y_in)x

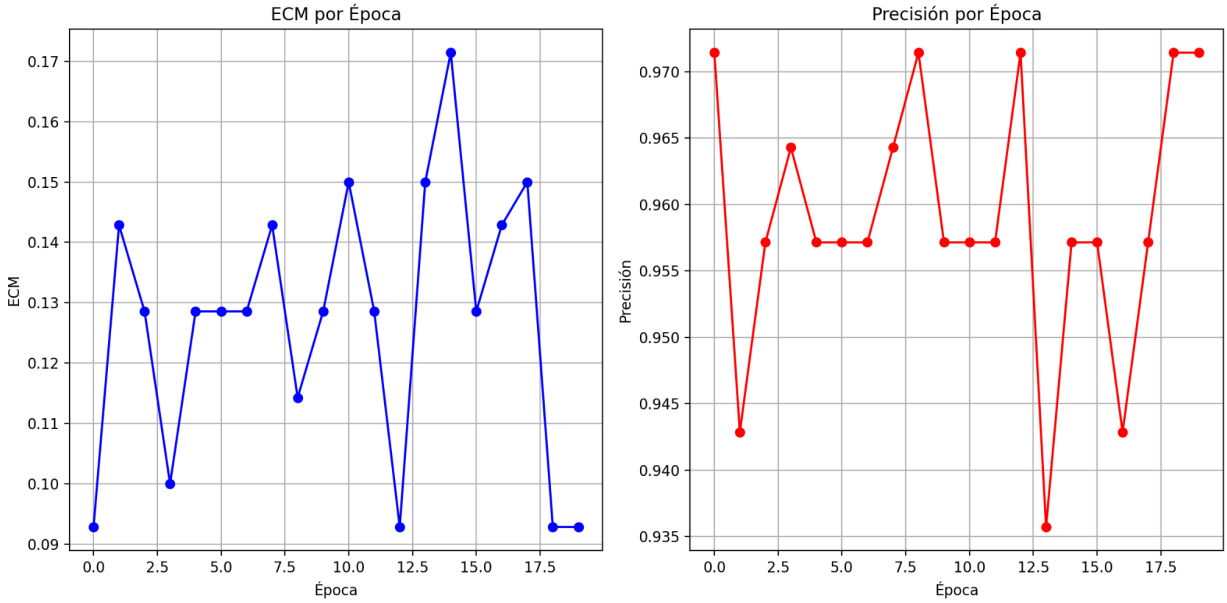
Los parámetros para crear la red neuronal son las dos capas input y output. El entrenamiento se realiza con la función fit con los parámetros X_train, y_train como datos de entrenamiento,

épocas y alfa como parámetros de control y otros parámetros de debug como verbose para imprimir el proceso y ecm para generar el cambio del error por cada época.

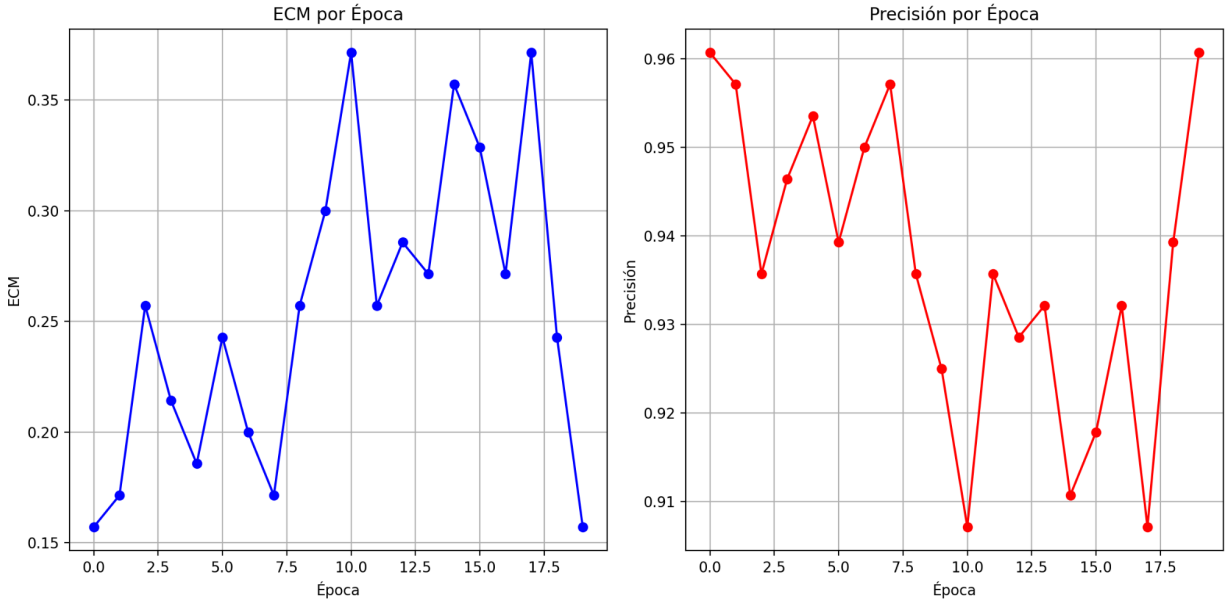
En fin los métodos predict y accuracy se pueden utilizar para comprobar la calidad del modelo.

Compara ambas redes, ¿qué sucede con el Error Cuadrático Medio (ECM) en cada una de ellas?

Con perceptron:



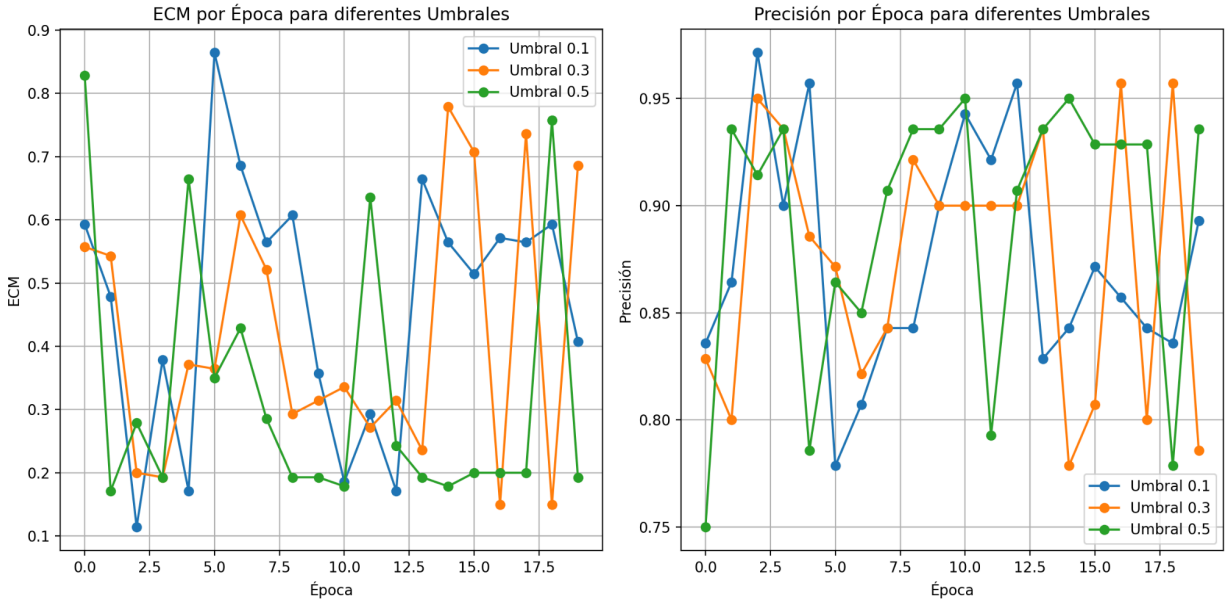
Con Adaline:



As such the ECM of the perception is consistently lower than that of the adaline network.

¿Que influencia presentan en el resultado cada uno de los parámetros que definen el comportamiento de las dos redes? Presentad gráficas que muestren como varía el resultado modificando un solo parámetro de control (p. ej.: umbral). Utilizad `problema_real1` *y el Modo 1.*

Con perceptron:



4.3.1.

Describid la implementación de ambas redes neuronales y de los parámetros usados y resultados.

Las redes neuronales son casi iguales a las descritas en 4.2.1, excepto que tienen 8 neuronas de entrada en lugar de 9.