Perceptrón Multi Capa (MLP):

Interpretación Geométrica (ejemplo)

Aprendizaje Automático

Dr. Luis Eduardo Falcón Morales
ITESM
Campus Guadalajara

Hemos comentado previamente que en general agregar diferentes capas ocultas o neuronas en dichas capas introduce mayor complejidad a la red y permite resolver problemas de mayor complejidad.

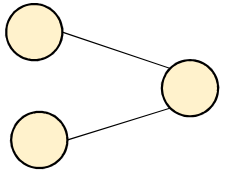
Veamos como se interpretan los pesos de las neuronas que se obtienen tanto en las capas de entrada y salida, como en las ocultas.

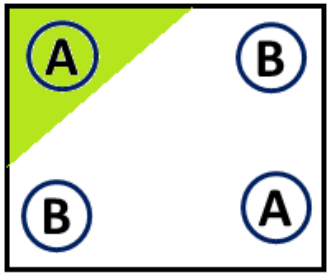
Observa que el perceptrón multicapa será capaz de resolver problemas no lineales, es decir, linealmente no separables, aunque para hacerlo aproxime la solución mediante un conjunto de rectas (hiperplanos).

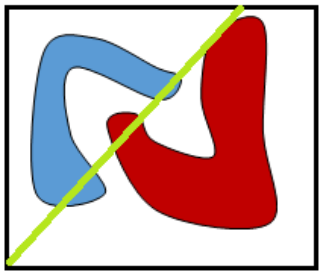
¿Qué significado tienen los pesos de las Capas Ocultas en el proceso de Clasificación y cómo se lleva a cabo dicho proceso?

XOR **Clasificación**

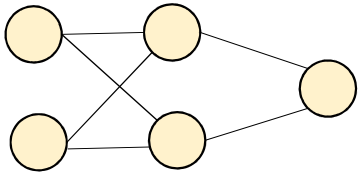
Sin capa oculta:

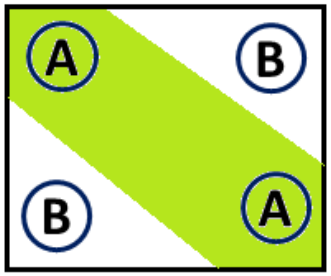


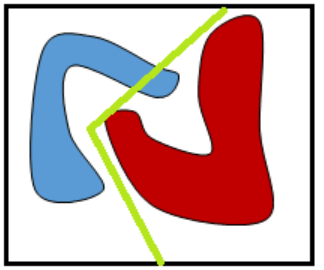




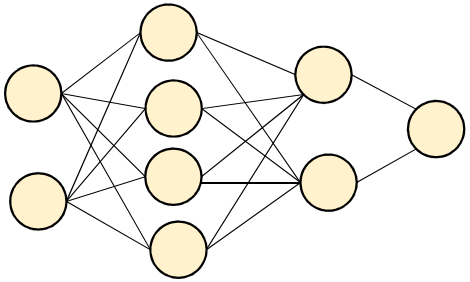
Una capa oculta:

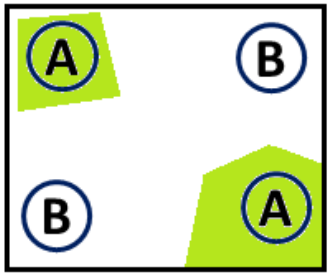


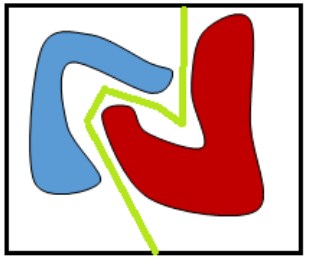




Varias capas ocultas:

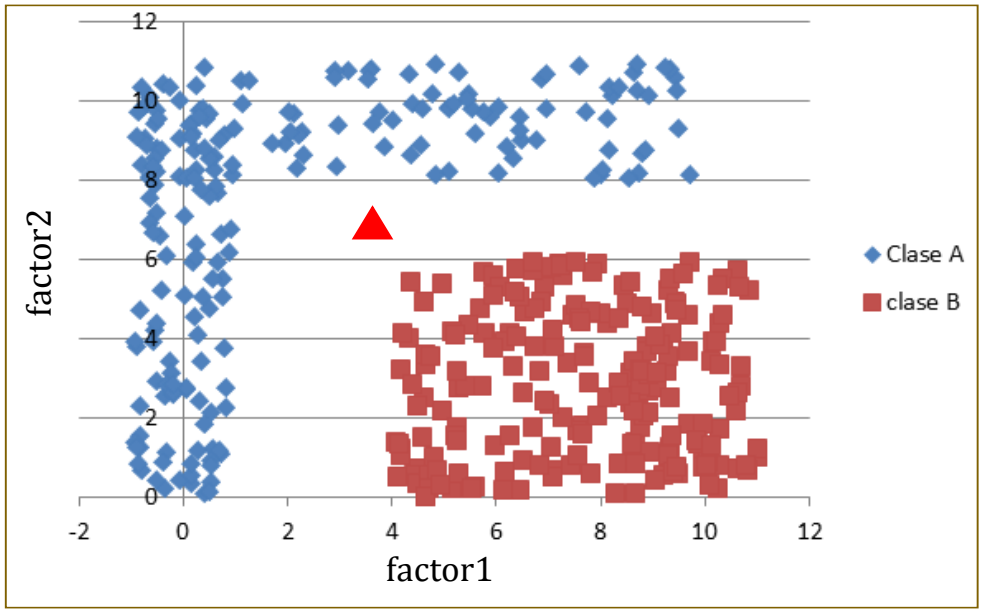






Ejemplo:

Supongamos que tenemos un conjunto formado por datos de dos clases, caracterizados por dos factores como se muestra en la gráfica de dispersión siguiente:



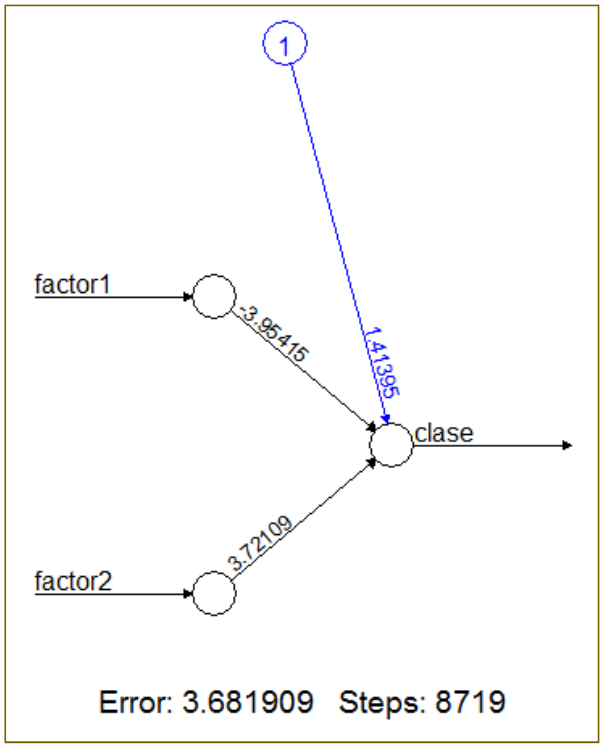
Obtener una solución usando una red neuronal artificial apropiada.

Con base a dicha solución, clasificar el punto en forma de triángulo de la imagen, es decir:

¿El punto $(3.5, 6.8)$ pertenece a la clase A o a la clase B?

Ejemplo (usando R)**:**

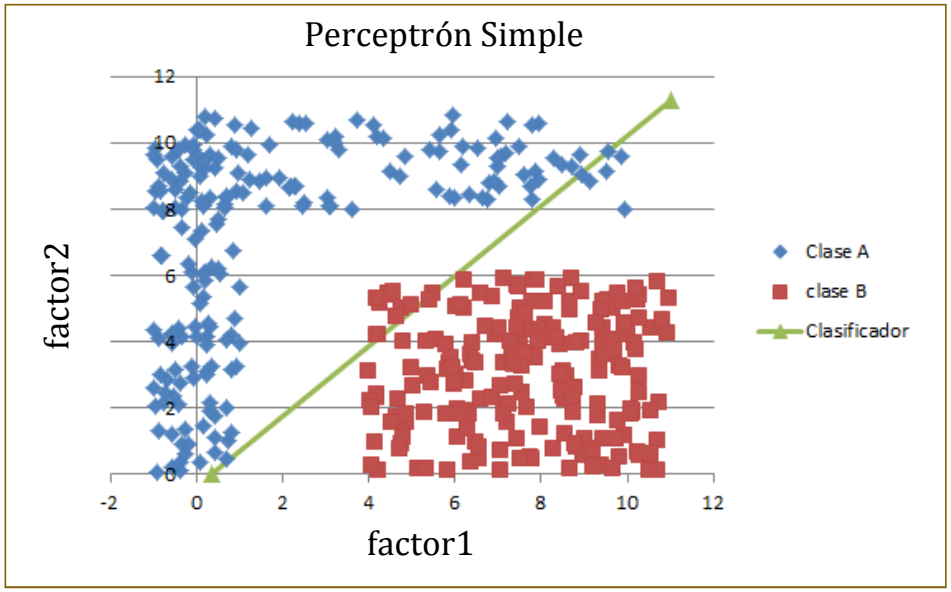
Solución con Perceptrón Simple



Clasificador obtenido:

$$h_{PS}(f_1, f_2) = -3.95f_1 + 3.72f_2 + 1.41 = 0$$

donde f_j denota al factor $j \in \{1, 2\}$. Graficando:



Desde un inicio sabíamos que el Perceptrón simple no podría separarlos al 100%. Aún así, se obtuvo una aproximación bastante aceptable como se observa en la gráfica y del valor de la suma de los cuadrados del error $SSE = 3.68$.

Además, observa en la gráfica que el vector $\vec{u} = -3.95\hat{\imath} + 3.72\hat{\jmath}$ está del mismo lado que los puntos de la clase A, entonces con su porcentaje de error correspondiente, sabemos ahora que:

$$h_{PS}(x) = \begin{cases} \geq 0 & si \quad x \in ClaseA \\ < 0 & si \quad x \in ClaseB \end{cases}$$

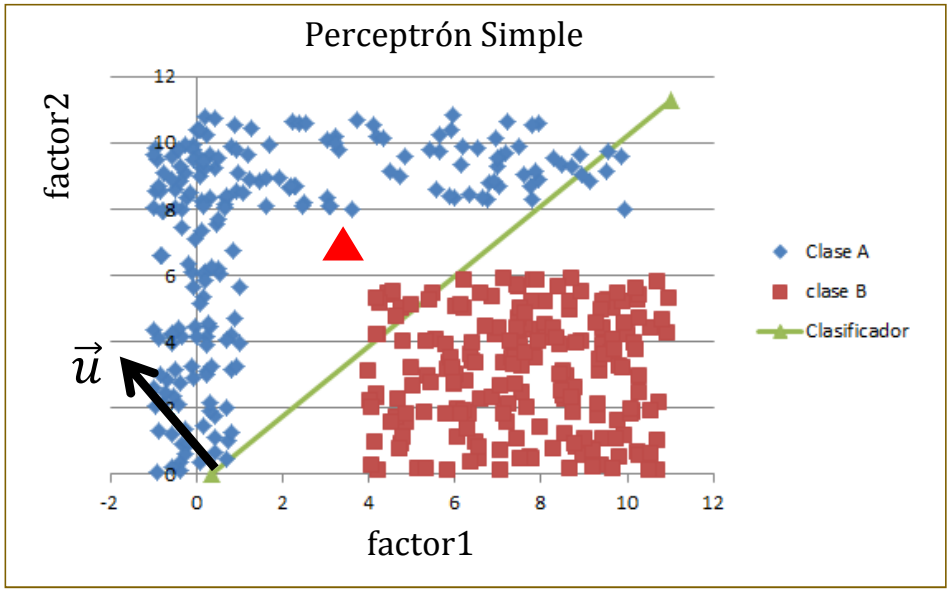
Así, al evaluarlo en el punto de interés $(3.5, 6.8)$, obtenemos:

$$h_{PS}(3.5, 6.8\ ) = 12.88 > 0$$

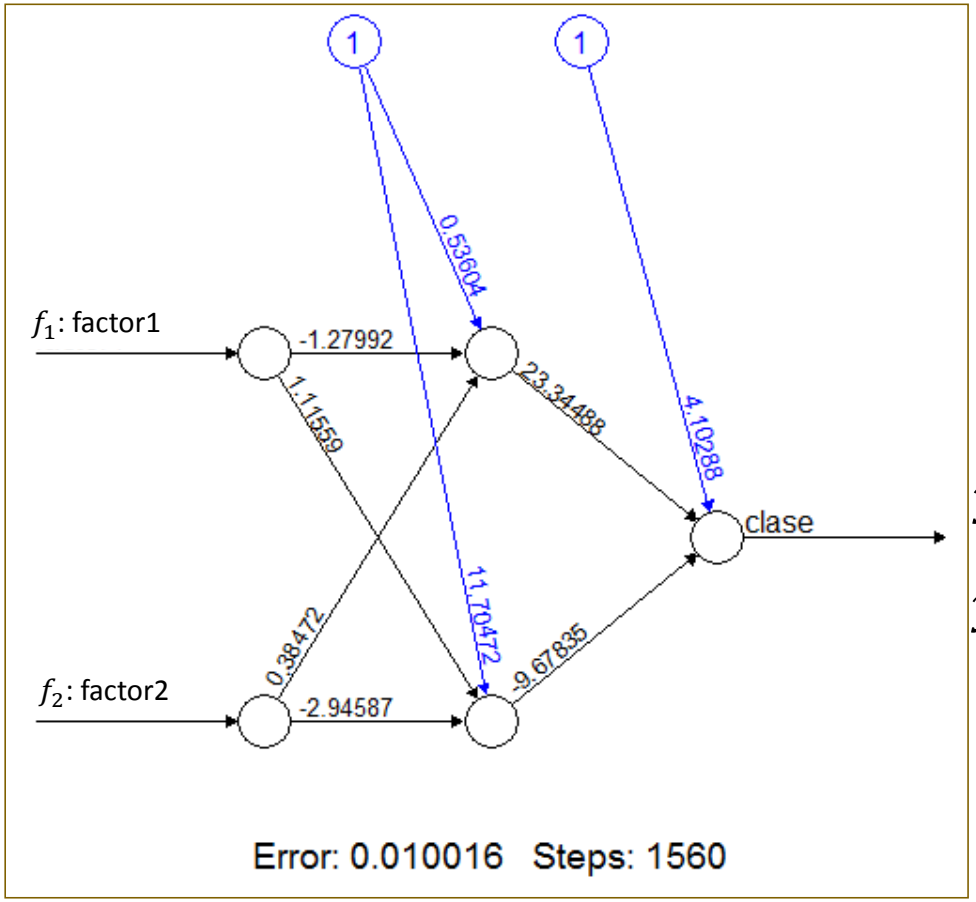
lo cual nos dice que dicho punto se clasificará dentro de la Clase A, como se observa claramente en la gráfica.

Clasificador obtenido con el algoritmo del Perceptrón Simple:

$$h_{PS}(f_1, f_2) = -3.95f_1 + 3.72f_2 + 1.41$$



Solución con MLP:
1 capa oculta con 2 neuronas en ella



Para este caso, vemos que tanto el Error como el total de iteraciones realizadas disminuyó considerablemente con respecto al caso del Perceptrón Simple.

Observamos que en este caso el clasificador está constituido por 3 ecuaciones: los 2 hiperplanos obtenidos de las 2 neuronas de la capa oculta, más el hiperplano de la capa de salida.
Estos lo podemos representar como sigue:

$$y_1 = h_1(f_1, f_2) = -1.28f_1 + 0.38f_2 + 0.54 = 0$$

$$y_2 = h_2(f_1, f_2) = 1.12f_1 - 2.95f_2 + 11.70 = 0$$

$$z = h(y_1, y_2) = 23.34y_1 - 9.68y_2 + 4.10 = 0$$

Solución con MLP:
1 capa oculta con 2 neuronas en ella

Primera Etapa:

Salidas de las dos neuronas de la capa Oculta:

$$y_1 = h_1(f_1, f_2) = -1.28f_1 + 0.38f_2 + 0.54 = 0$$

$$y_2 = h_2(f_1, f_2) = 1.12f_1 - 2.95f_2 + 11.70 = 0$$

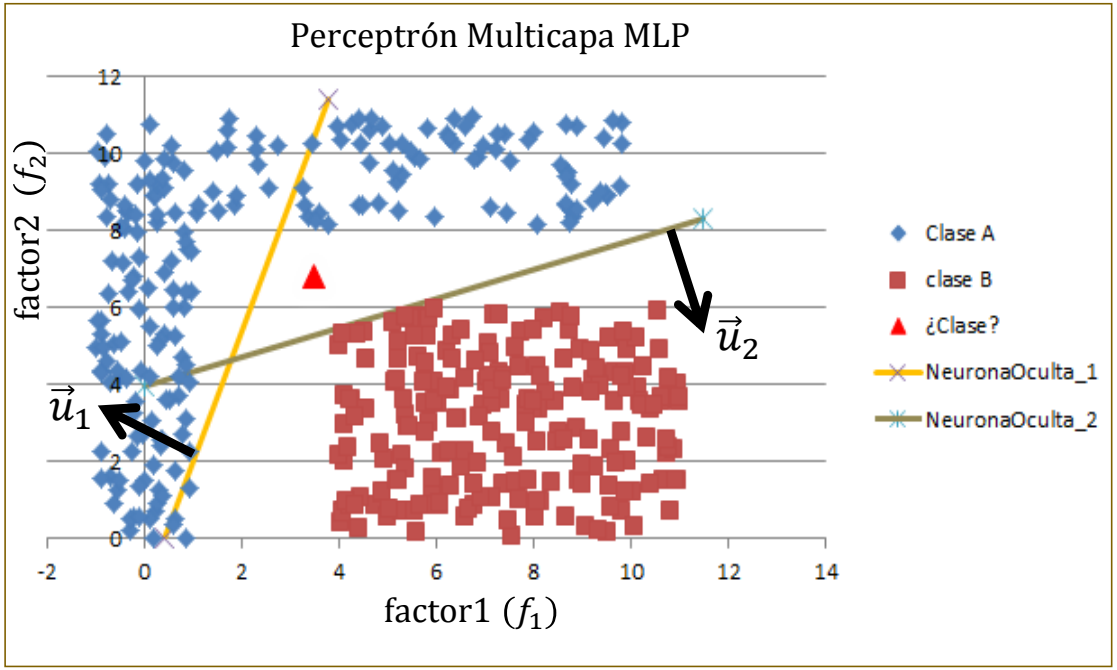


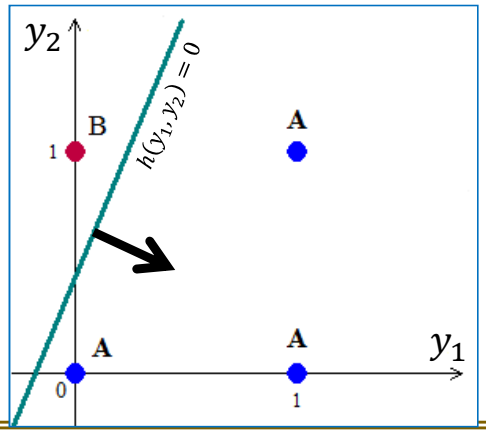
Tabla de Valores de Verdad para las dos Etapas del MLP

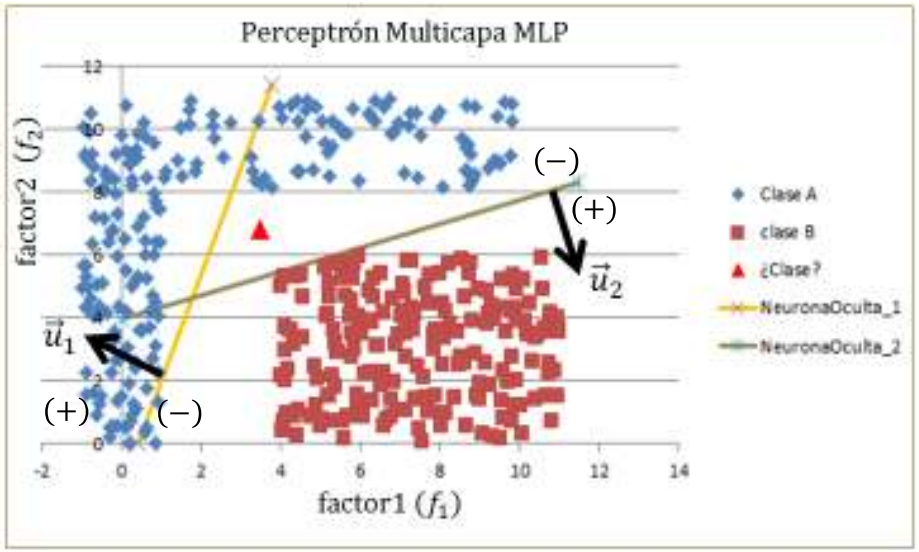
		Etapa 1		Etapa 2
f_1	f_2	y_1	y_2	Etiqueta: z
0	2	1 (+)	1 (+)	A
0	8	1 (+)	0 (−)	A
8	0	0 (−)	1 (+)	B
10	10	0 (−)	0 (−)	A

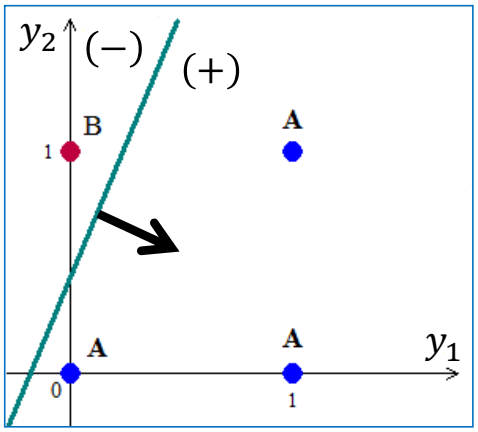
Segunda Etapa:

Resultado de la neurona de la capa de Salida:

$$z = h(y_1, y_2) = 23.34y_1 - 9.68y_2 + 4.10 = 0$$







Finalmente, para determinar la clase a la cual debiera pertenecer el punto $(3.5, 6.8)$, evaluamos primero en las funciones de las neuronas ocultas:

$$y_1 = h_1(3.5, 6.8\ ) = -1.356 < 0$$

$$y_2 = h_2(3.5, 6.8) = -4.44 < 0$$

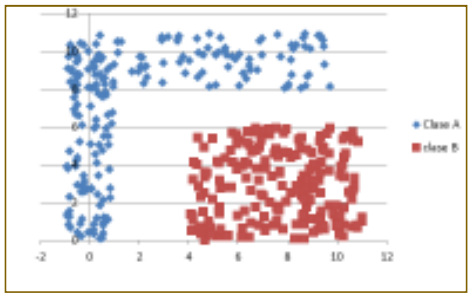
Y estos resultados los evaluamos ahora en la función de la neurona de salida:

$$z = h(y_1, y_2) = 23.34y_1 - 9.68y_2 + 4.10$$

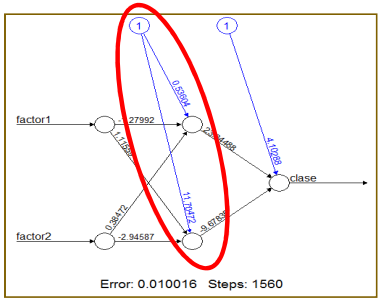
$$z = h(-1.356, -4.44\ ) = 15.43 > 0$$

Es decir, de acuerdo al modelo obtenido, el punto $(3.5, 6.8)$ debe pertenecer a la región positiva de su clasificador $z = h(y_1, y_2)$. Esto lo observamos en la gráfica del plano y_1y_2 que incluimos, donde la región que está asociada a la Clase A es la clase positiva, por lo que el punto dado se clasifica en la Clase A.

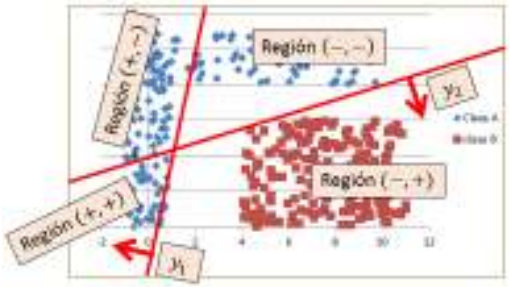
En resumen, podemos decir que la acción realizada por las neuronas de la capa oculta de este ejemplo, fue transformar los datos de entrada a los vértices de un cuadrado unitario, el cual en este caso ya fue linealmente separable.



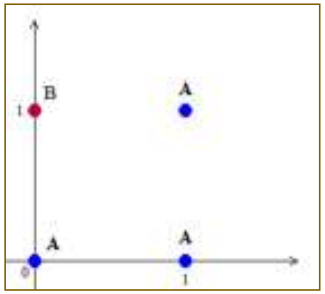
Conjunto no linealmente separable, es decir, con una sola recta (hiperplano) no es posible separar ambos conjuntos A y B.



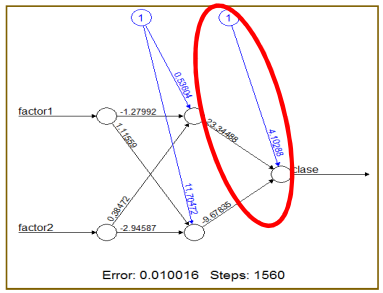
Acción de las neuronas de la capa oculta



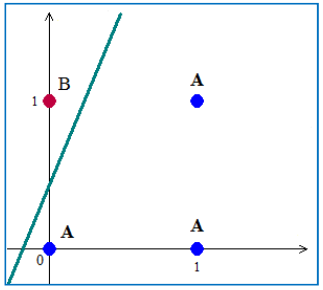
Regiones (y_1, y_2): El espacio de datos de entrada es dividido en 4 regiones.



Las 4 regiones son transformadas a un nuevo espacio donde el conjunto de datos transformado ya es linealmente separable



Acción de las neuronas de la capa de salida

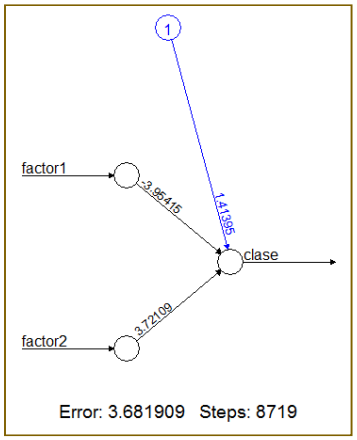


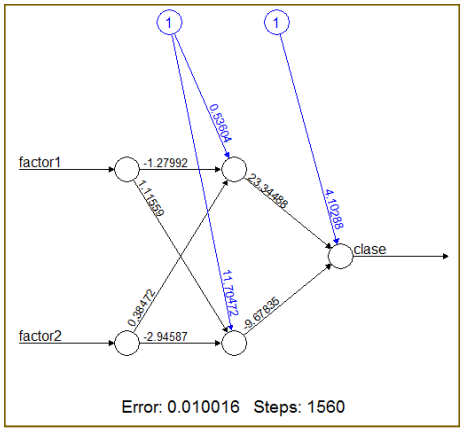
Problema de clasificación resuelto

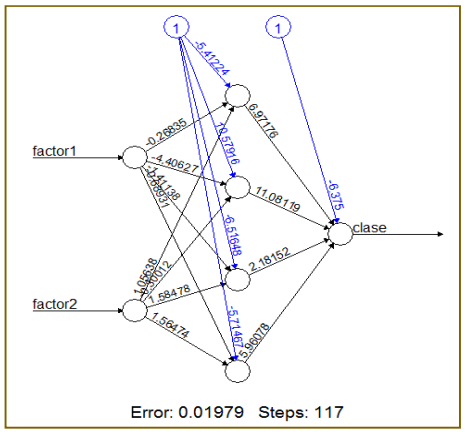
Se pueden seguir agregando más neuronas en la capa oculta, aunque no necesariamente se tendrá cada vez un mejor desempeño, o al menos no de manera significativa.

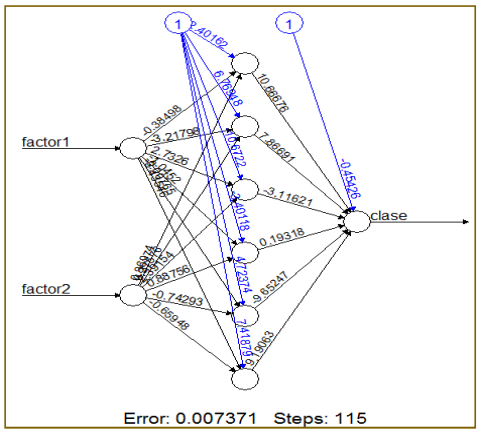
Uno debe evaluar la cantidad de procesamiento realizado con respecto al desempeño obtenido.

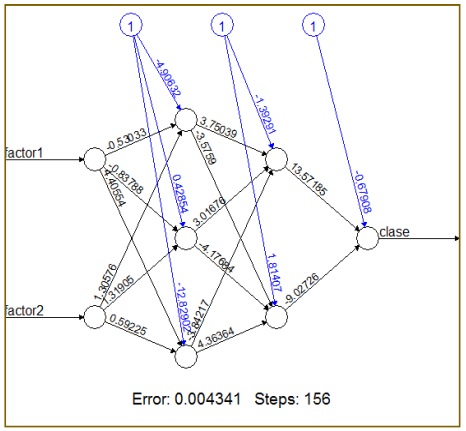
A continuación se muestran las soluciones para diferentes arquitecturas del ejemplo analizado previamente:







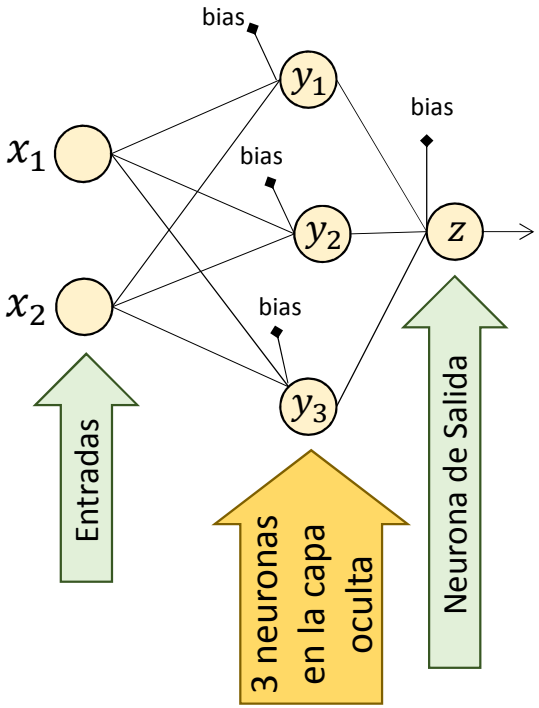




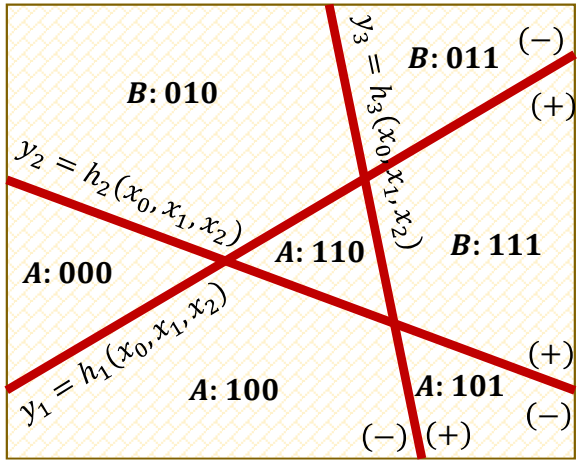
Sin embargo, hay que tener presente que una gran cantidad de capas ocultas o de neuronas en dichas capas ocultas, puede llevar al sobre-entrenamiento.

Supongamos que se tiene un conjunto de puntos con 2 factores independientes y 1 dependiente asociados a dos clases A y B, que no son linealmente separables en el espacio de 2 dimensiones.

Se propone el siguiente modelo MLP:



Como solamente tenemos 2 factores en los datos de entrada, entonces cada neurona en la capa oculta genera una hiperplano (recta) en el espacio de 2 dimensiones.

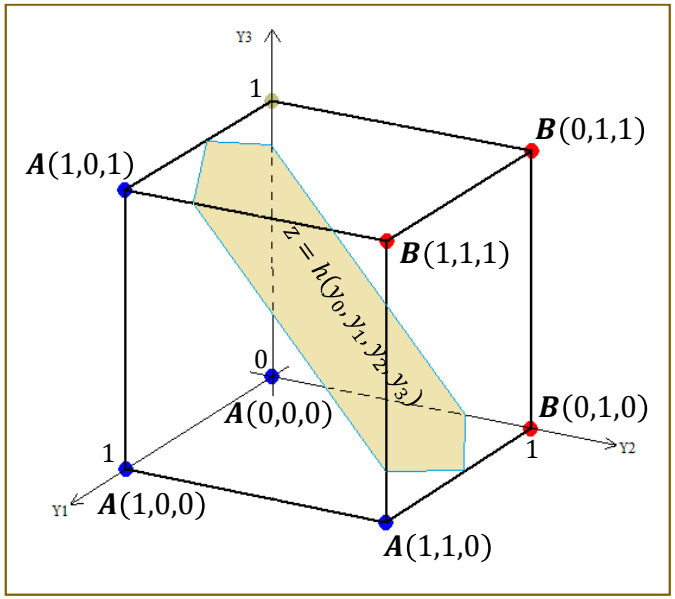


Supongamos que cada región está asociada a la clase indicada en la Figura.

Regiones (y_1, y_2, y_3):

El espacio de datos de entrada es dividido en 7 regiones por estas 3 neuronas ocultas.

Las 7 regiones son transformadas ahora a 7 vértices de un cubo unitario de 3 dimensiones, a través de las 3 neuronas ocultas.



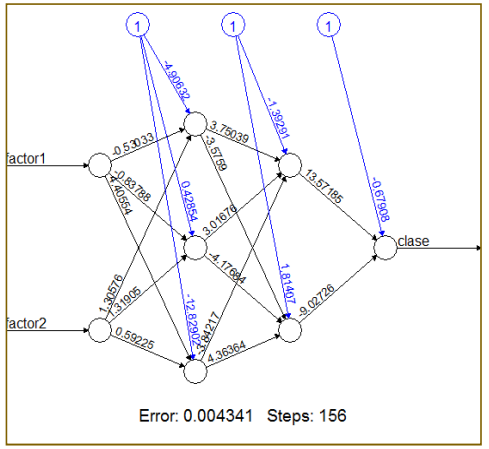
Para este ejemplo, en este espacio de 3 dimensiones los vértices (clases) ya son linealmente separables por el hiperplano de la neurona de salida.

El vértice $(0,0,1)$ del cubo y no asociado a ninguna clase, se dice que pertenece a un *poliedro virtual*.

En general, existen muchos modelos para construir una red neuronal artificial, pero podemos decir que todas ellas se basan en la manera en que se definen las siguientes características:

- **Función de activación**: es la función que transforma los datos de entrada a una neurona en un valor de salida, el cual podrá ser utilizado como entrada en otra neurona.

- **Topología o arquitectura de la red**: indica el número de capas ocultas en la red, así como el número de neuronas en cada capa. Además nos dice la manera en que se conectan las neuronas entre sí y la manera en que se propaga la información en la red: la información se transmite hacia adelante en una red neuronal estándar, o bien, puede viajar en ambas direcciones como en las redes neuronales recurrentes.



- **Forma de entrenamiento/aprendizaje**: nos habla de los métodos que se usarán para ajustar los pesos de la red neuronal. Por ejemplo, el método de propagación hacia atrás de los errores (*backpropagation*): el cual una vez transmitida la información de la capa de entrada a la capa de salida, se mide el error con el valor real de la salida, para posteriormente dicho error propagarlo hacia atrás en la red y modificar los pesos que ayuden a disminuir futuros errores. El método para disminuir dichos errores también deberá ser indicado: gradiente descendente, Levenberg-Marquardt, etc.