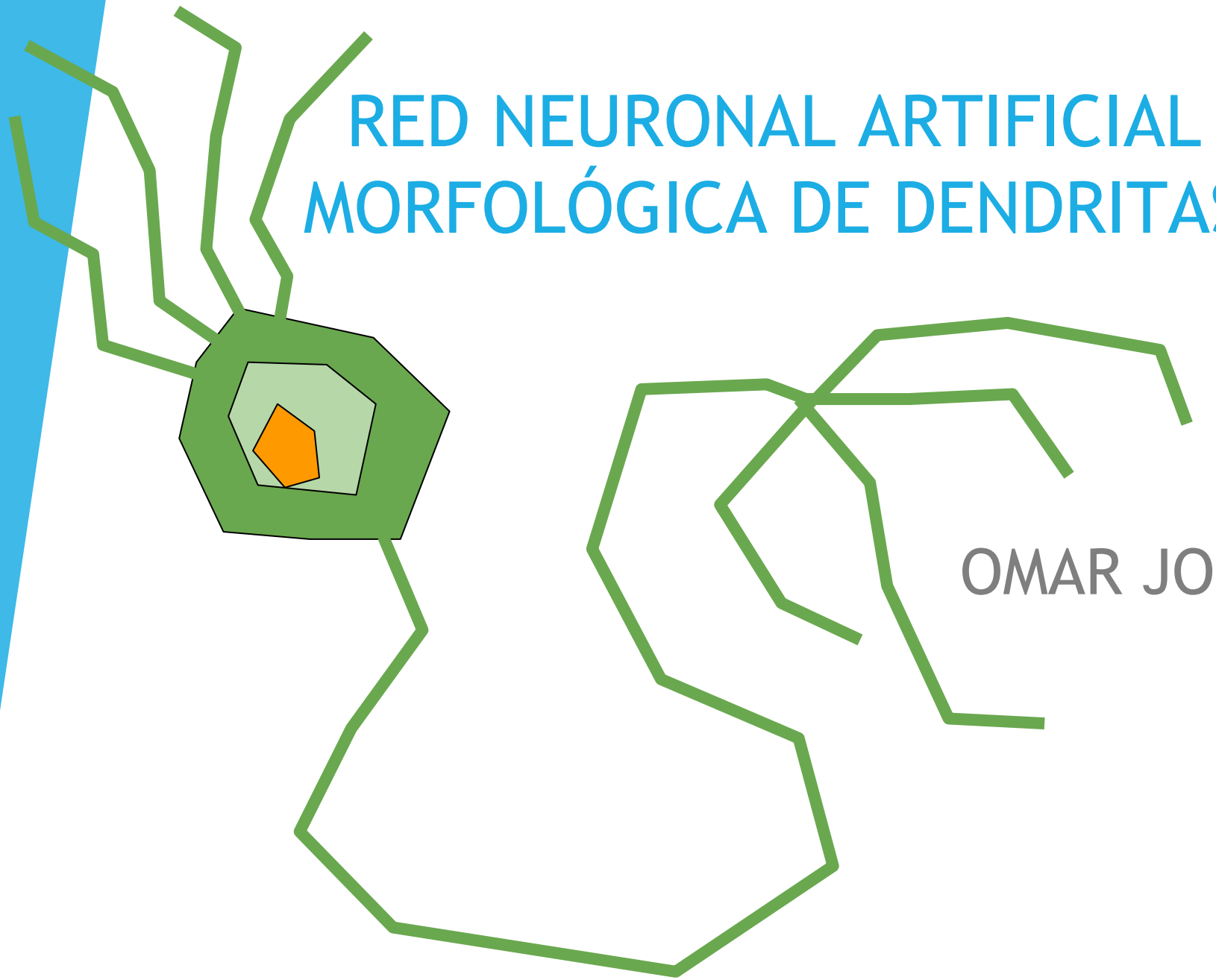
RED NEURONAL ARTIFICIAL MORFOLÓGICA DE DENDRITAS



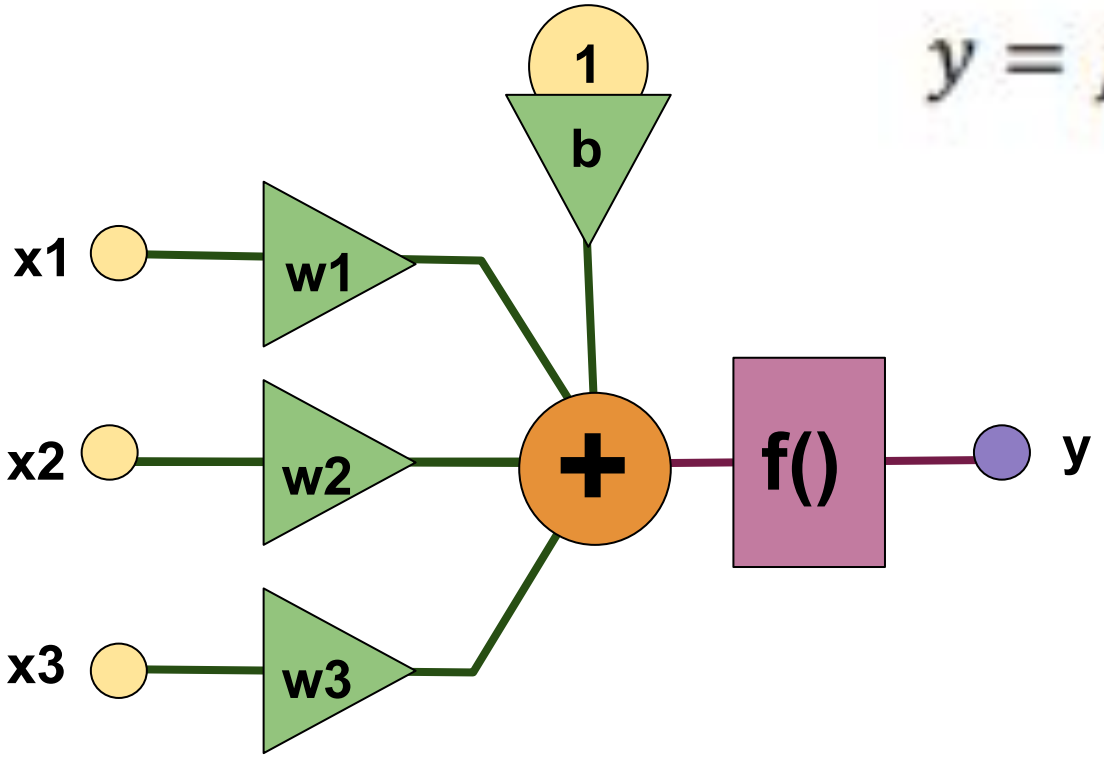
OMAR JORDAN

ÍNDICE

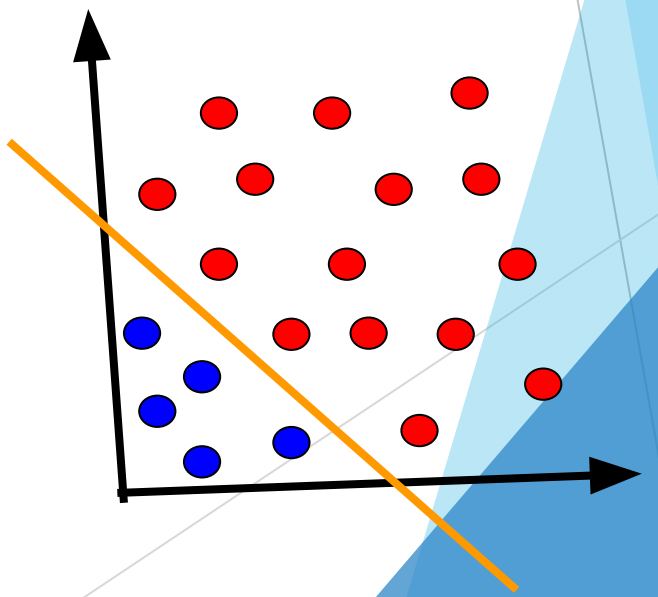
- Qué es DMNN?
- K-means
- Evolución Diferencial
- Optimización Dendritas
- Software

QUÉ ES DMNN?

Recordáis al perceptrón?
el que crea un hiper-plano en el hiper-espacio de soluciones?



$$y = f\left(b + \sum_{i=1}^{n} x_i . w_i\right)$$

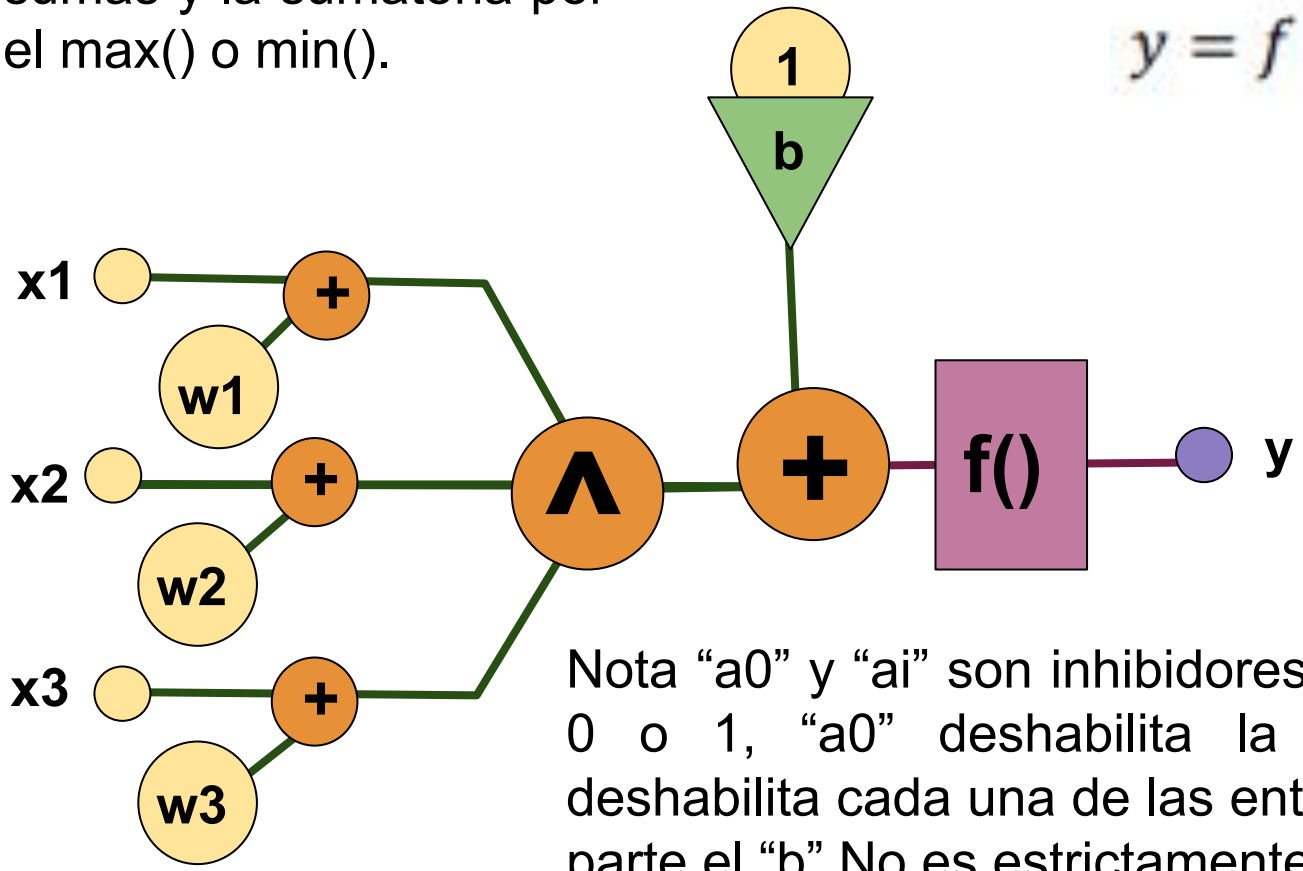


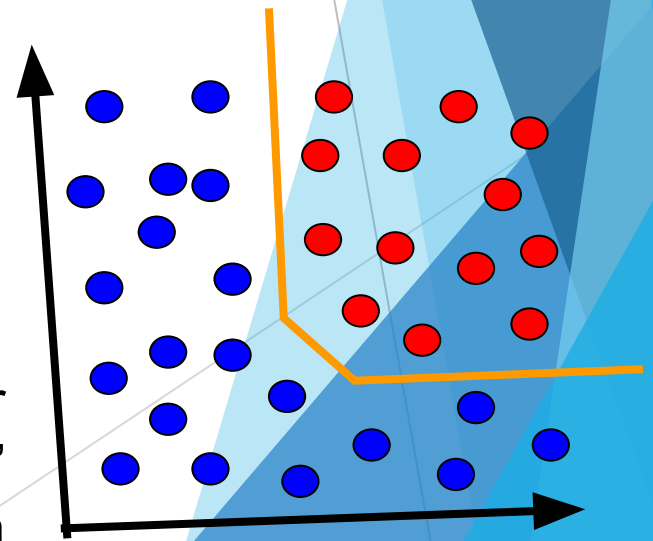
QUÉ ES DMNN?

La red morfológica cambia las operaciones de multiplicación por sumas y la sumatoria por el max() o min().

Esto es la estructura básica de la red morfológica, si pintamos su salida para un gráfico 2D obtenemos:

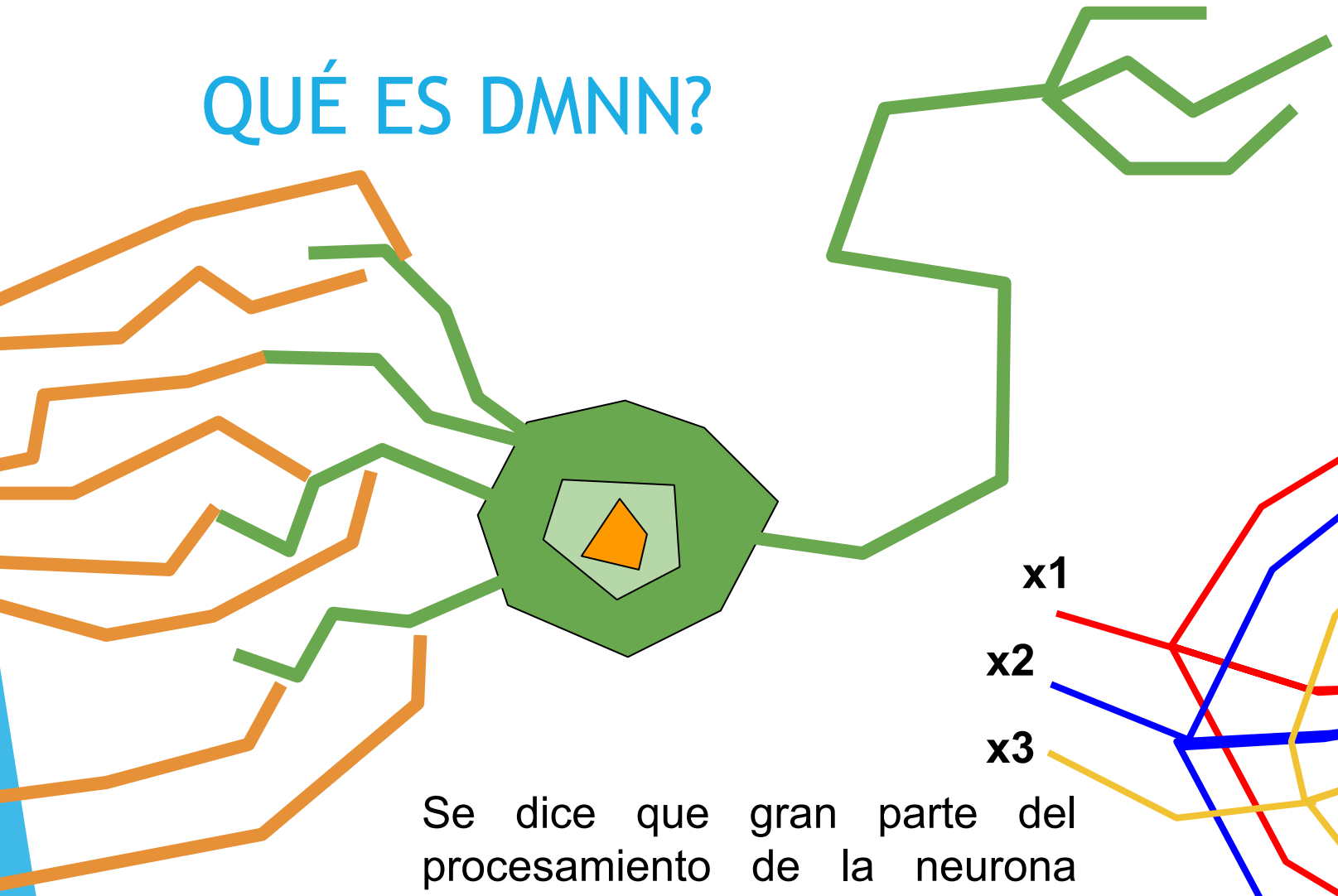
$$y = f\left(b + a_0 . \bigwedge_{i=1}^{n} a_i . (x_i + w_i)\right)$$

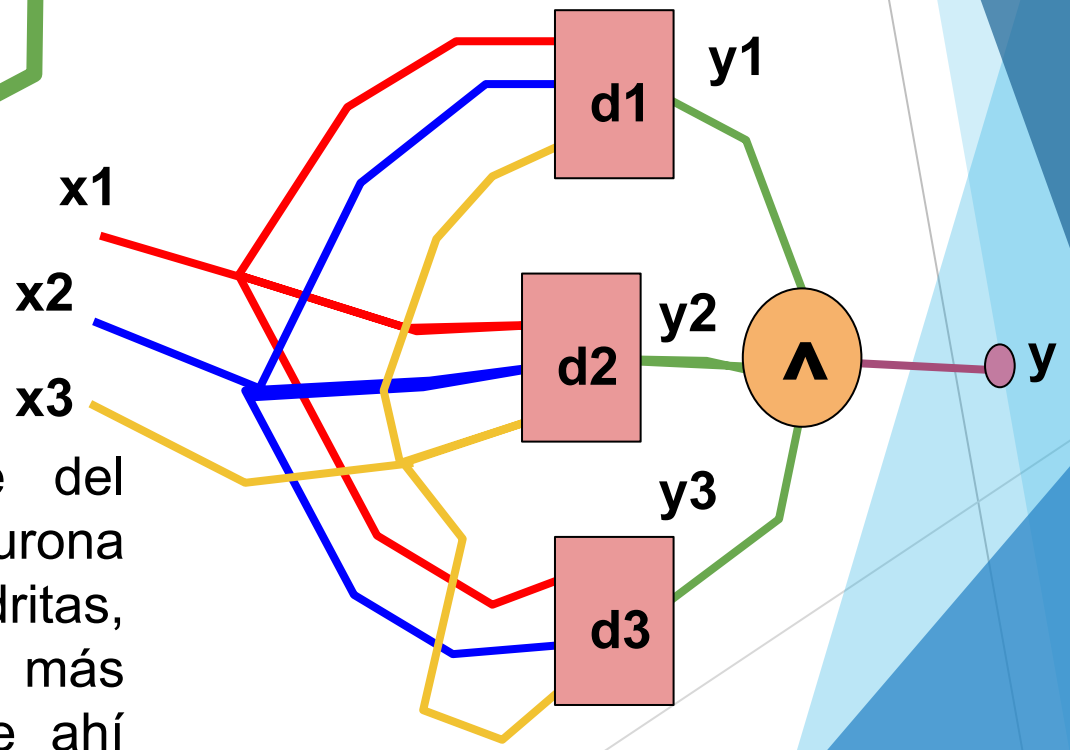




Nota “a0” y “ai” son inhibidores, pueden valer 0 o 1, “a0” deshabilita la salida y “ai” deshabilita cada una de las entradas; por otra parte el “b” No es estrictamente necesario.

QUÉ ES DMNN?





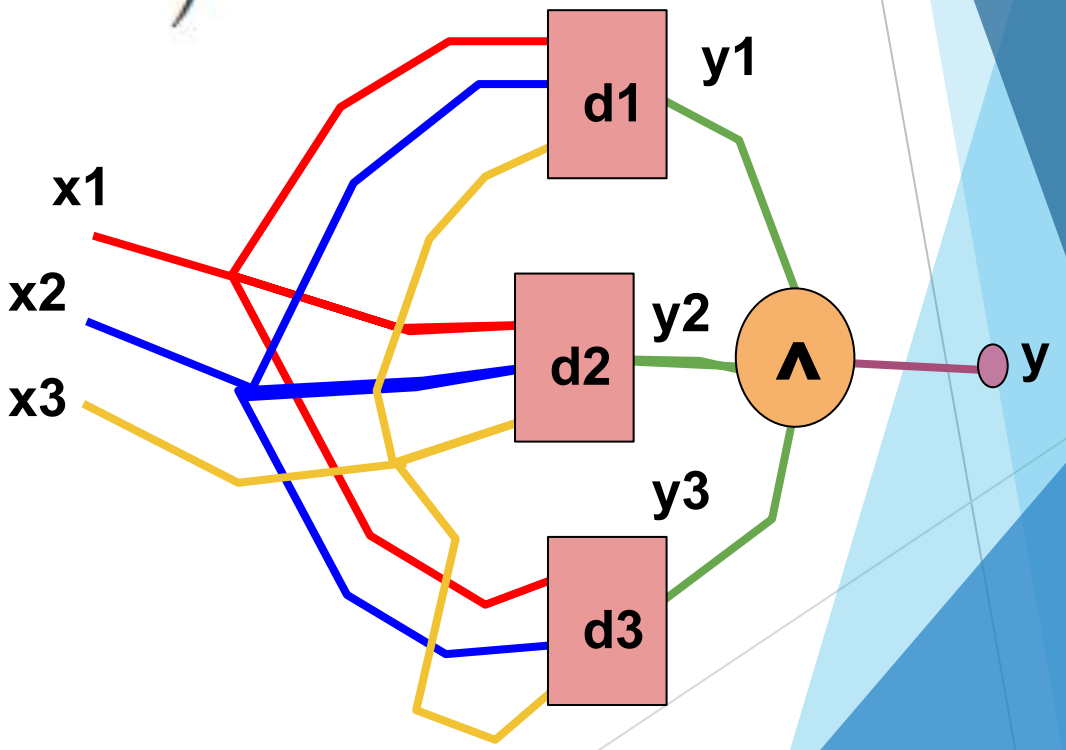
Se dice que gran parte del procesamiento de la neurona biológica ocurre en las dendritas, en cada una puede haber más de una señal entrante, de ahí surge el modelo artificial.

QUÉ ES DMNN?

$$y_k = \overset{1}{\underset{h=0}{V}} \left(\overset{n}{\underset{i=1}{V}} \left((-1)^h . (x_i + w_{ik}^h) \right) \right)$$

$$y = \overset{K}{\underset{k=1}{\Lambda}} (y_k)$$

La “w” no está elevada a la “h”, para toda entrada “xi” independiente de la dimensionalidad del problema hay dos pesos “w high” y “w low” asociados.

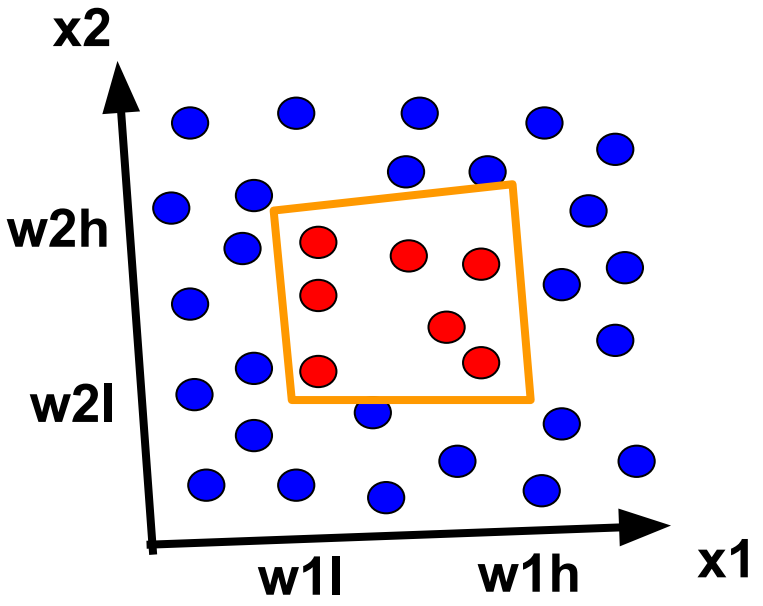


QUÉ ES DMNN?

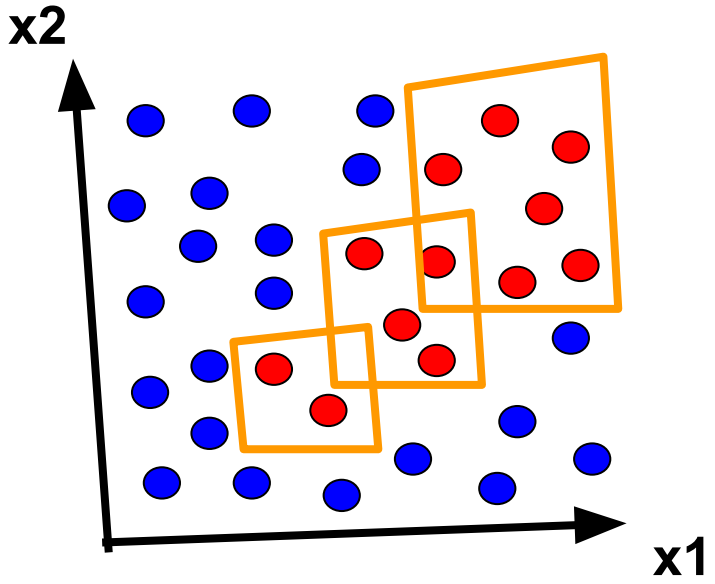
Esto es una dendrita, operada con min().

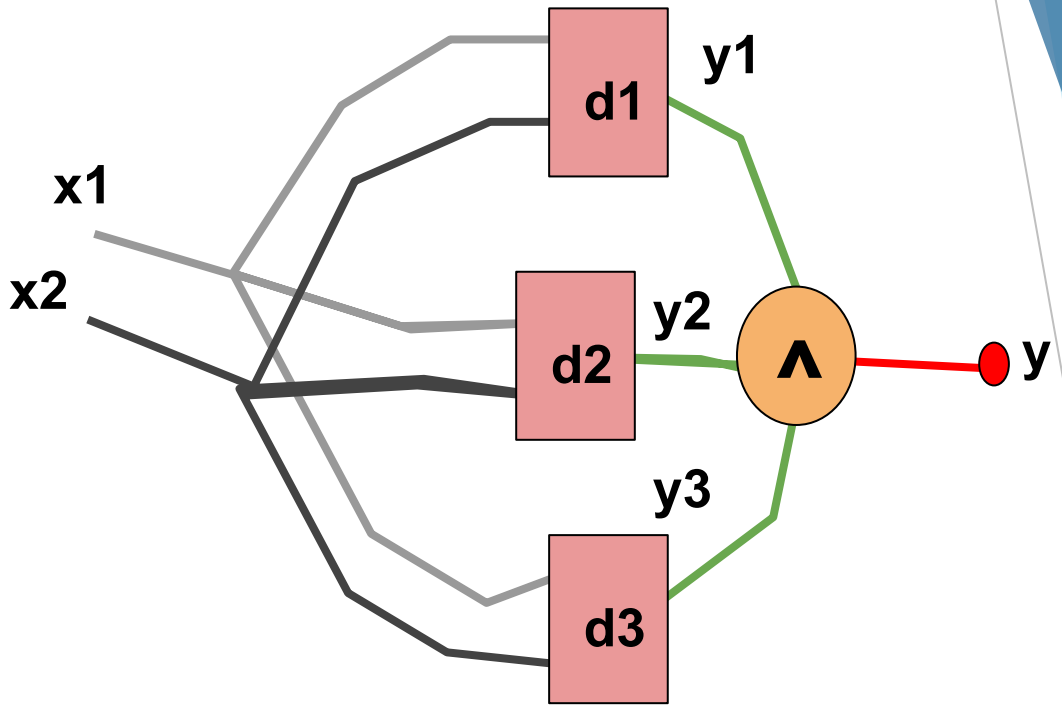
$$y_k = \mathrm{V}\left[\mathrm{V}\left(x_1 + w_1^l, x_2 + w_2^l\right), \mathrm{V}\left(-\left(x_1 + w_1^h\right), -\left(x_2 + w_2^h\right)\right)\right]$$

Con 2 dimensiones de entrada x1, x2 es más entendible; en la gráfica se ve como los “w low” son un extremo de la caja y los “w high” son el otro extremo; esto proyectado hacia las hiper-dimensiones crea hiper-cajas!
pd: puntos rojos son positivos y azules negativos.



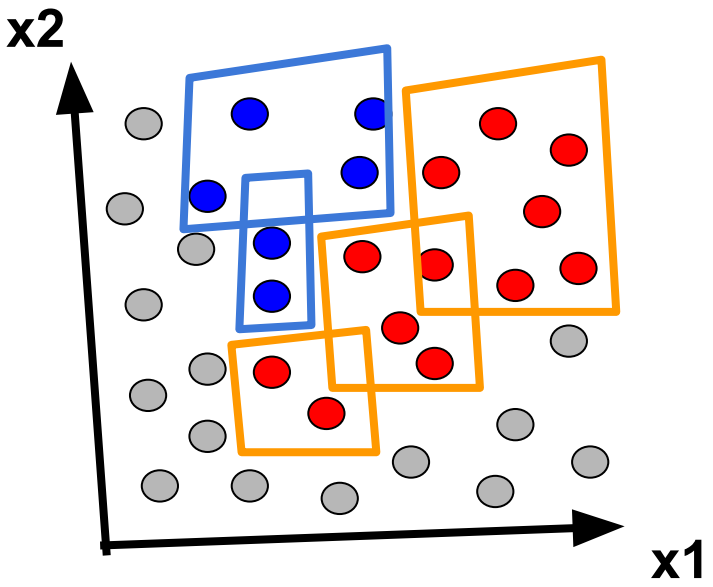
QUÉ ES DMNN?

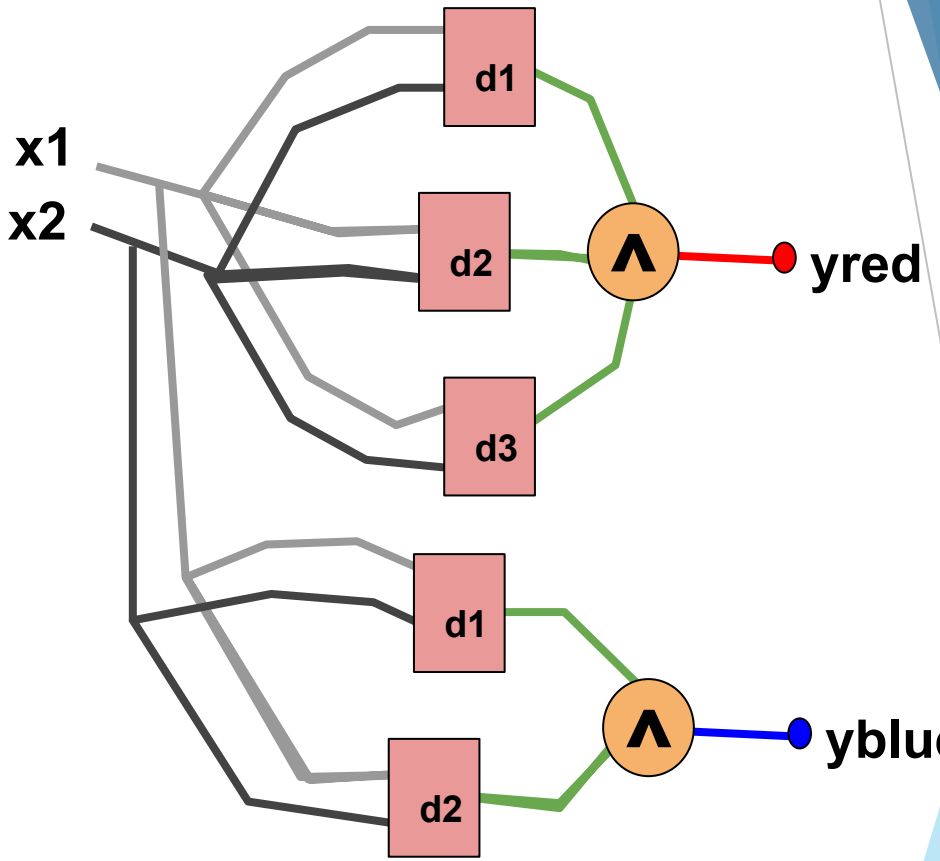




La neurona tiene 3 dendritas y 2 entradas, cada dendrita es una caja en el gráfico 2D, mientras la salida “y” es positiva (roja) siempre y cuando esté dentro de las cajas.

QUÉ ES DMNN?

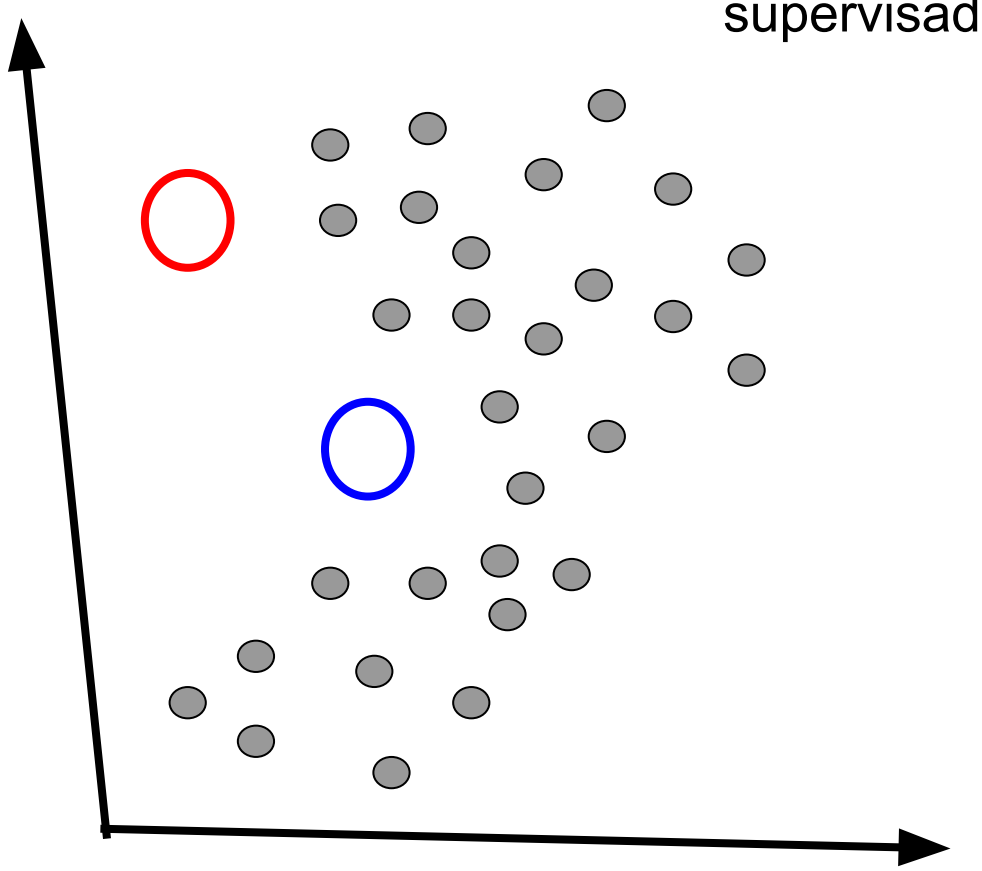




Para el proyecto presentado se utilizó más de una neurona, específicamente hasta 4, representadas con colores rojo, azul, amarillo y verde; a la salida se comparan sus valores y la que tiene el mayor es la clase ganadora, en la gráfica se forza a una clase indefinida (gris) cuando todas las salidas son menores que cero.

K-MEANS

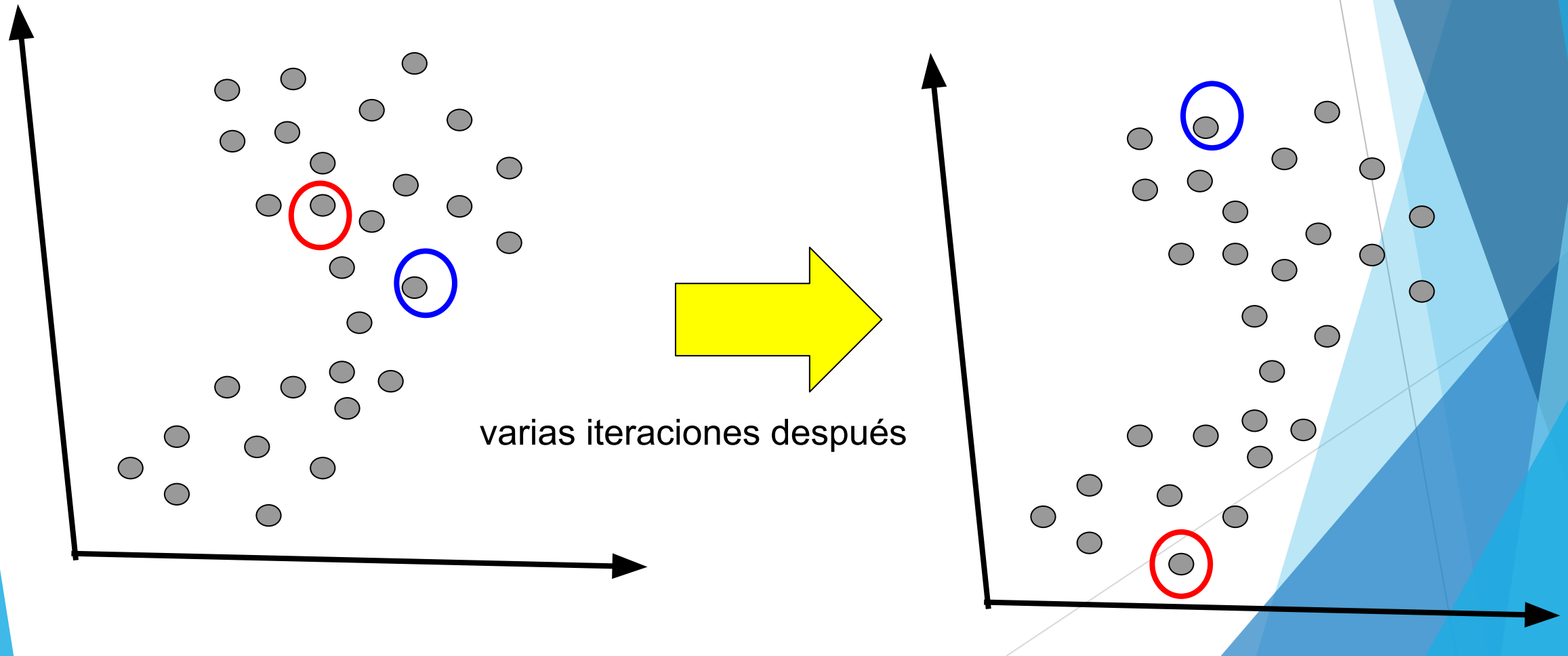
En las redes DMNN no se suelen poner las hiper-cajas al azar puro como se hace por ejemplo con los pesos sinápticos de un MLP.
Para inicializar las cajas de las dendritas se utiliza este algoritmo no supervisado de auto-organización.



En el K-means básico se ponen los centroides al azar, pero es mejor un algoritmo que inicialice los centroides del K-means más adecuadamente.

K-MEANS

Al iniciar se ponen los centroides sobre patrones aleatorios, luego un ciclo de iteraciones hace que busquen al azar patrones desde los cuales la distancia hasta todos los otros centroides sea máxima, evitando también quedar muy lejos de unos pero muy cerca de otros (usar distancia ^⅛).

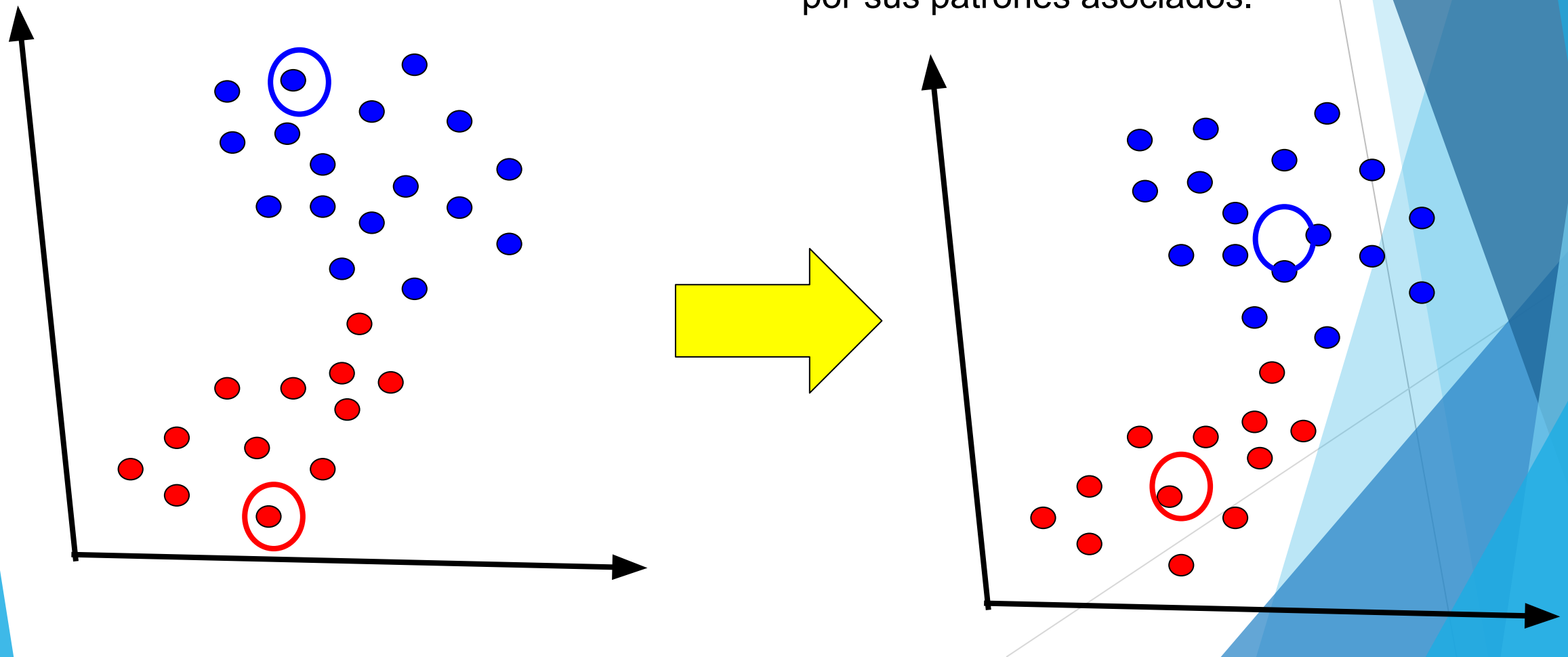


K-MEANS

El ciclo de iteraciones del K-means tiene dos partes:

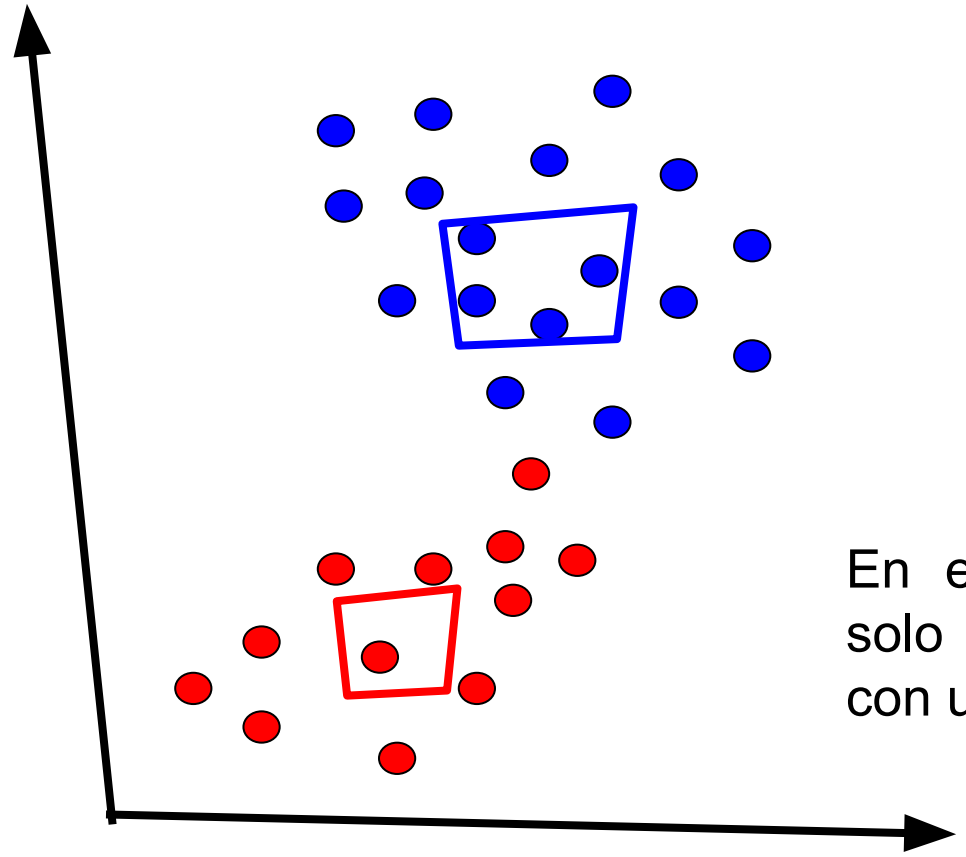
Primero se asocian los patrones al centroide más cercano.

Segundo se mueven los centroides al lugar promedio dado por sus patrones asociados.



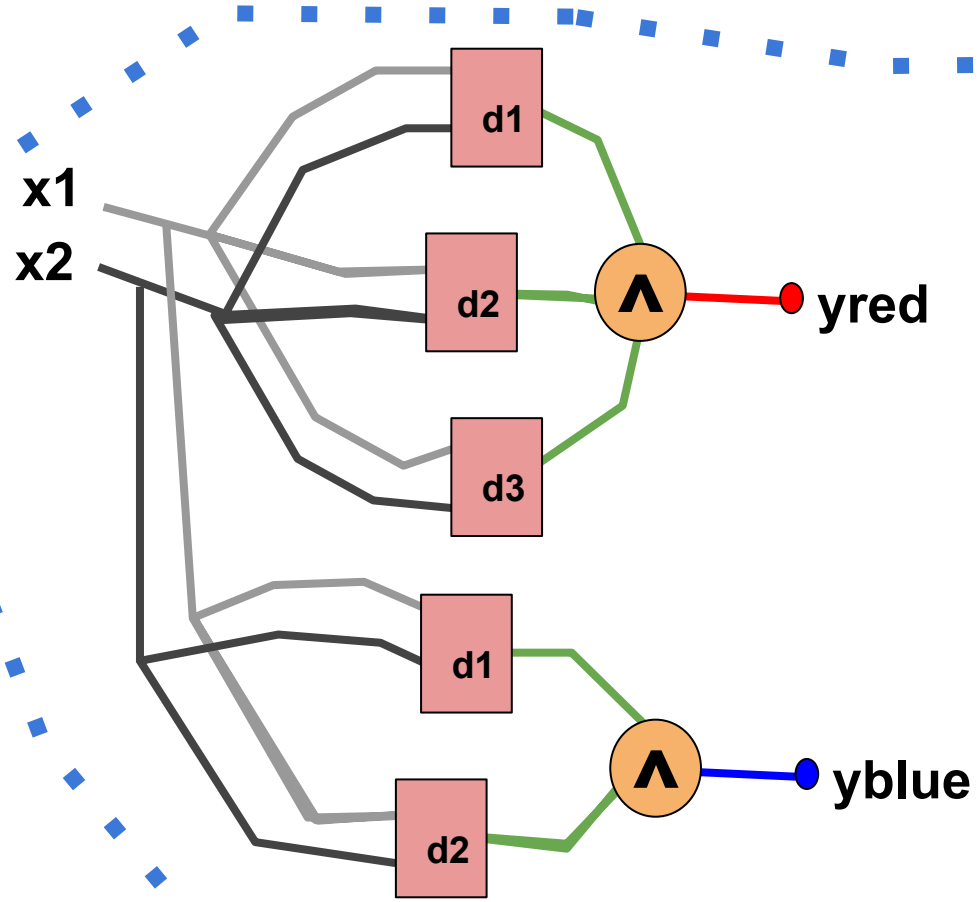
K-MEANS

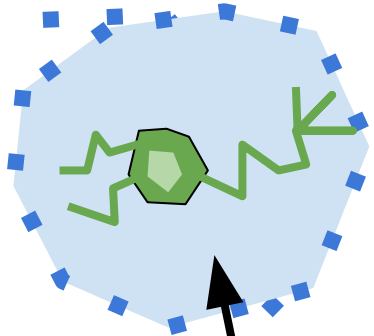
Se crea la DMNN usando los centroides como centros de las cajas de las dendritas, esto se hace sumando y restando al centroide pequeños valores aleatorios.



En este ejemplo tendríamos solo dos neuronas, cada una con una dendrita.

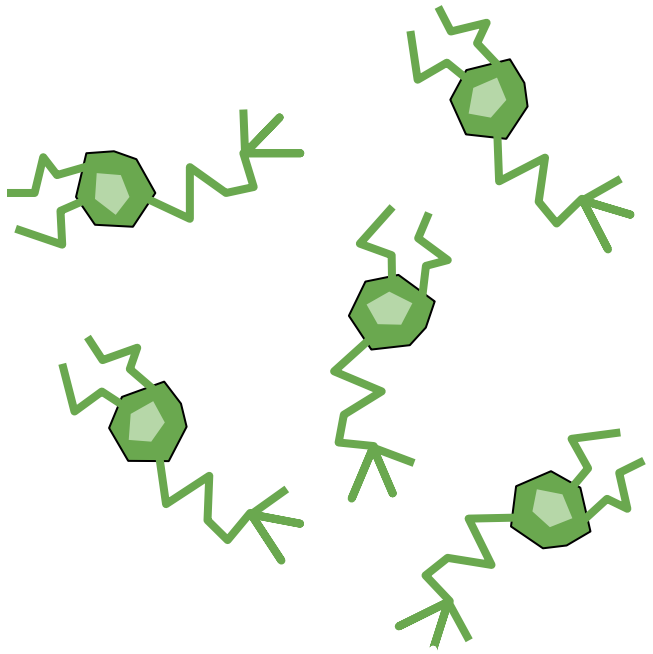
EVOLUCIÓN DIFERENCIAL





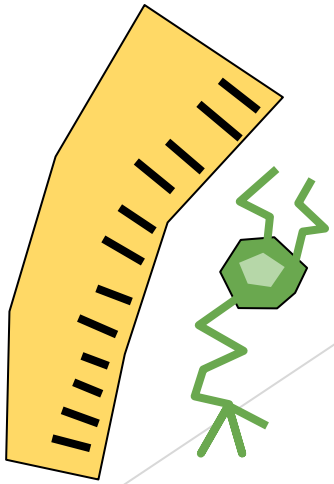
Toda la red neuronal que obtuvimos (con sus varias neuronas y múltiples dendritas), ahora será una unidad a la que llamaremos individuo.

EVOLUCIÓN DIFERENCIAL

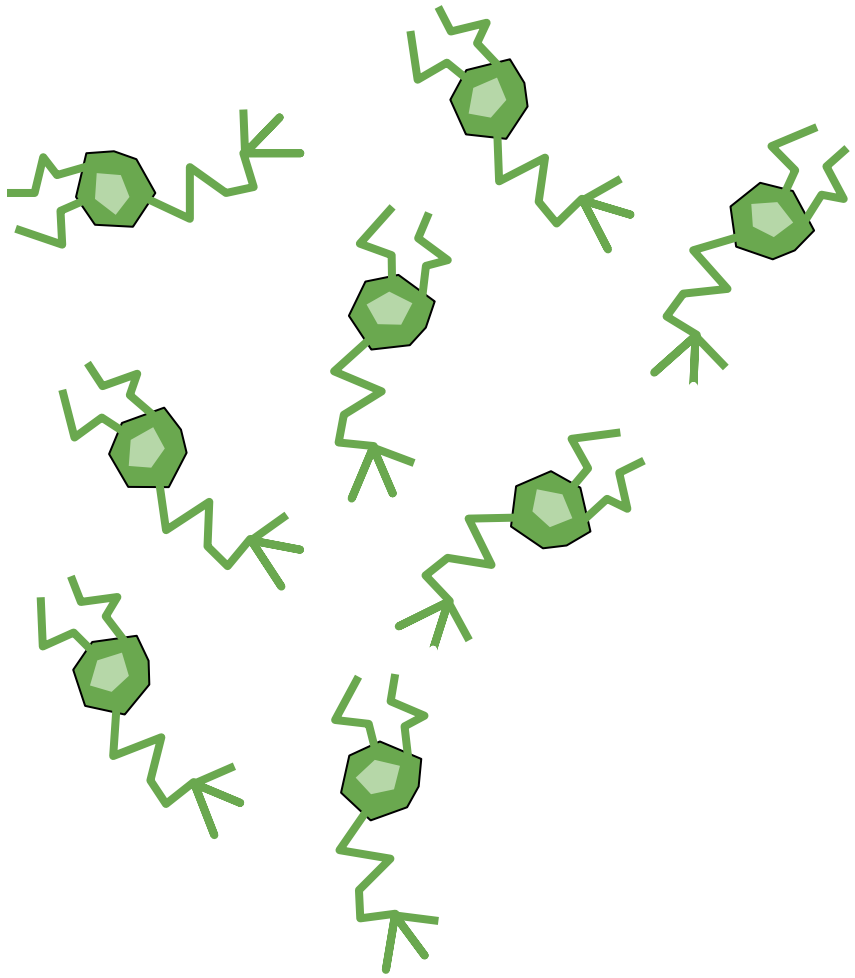


Se crean varios individuos, con parámetros ligeramente cambiados al azar, a este grupo lo llamaremos la primera generación y de ahora en adelante las iteraciones serán llamadas generaciones.

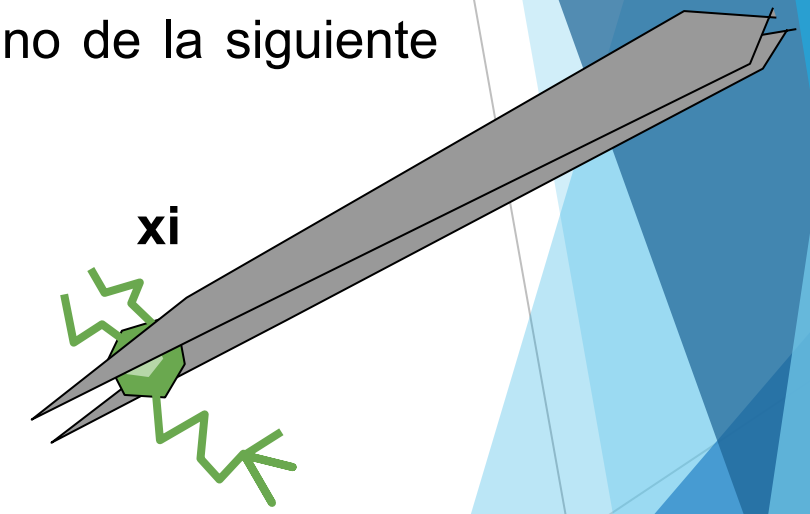
Debemos tener una función de aptitud, fitness o función de error para medir qué tan bueno es un individuo; un ejemplo de estas funciones es el error cuadrático medio que evalúa todos los patrones.



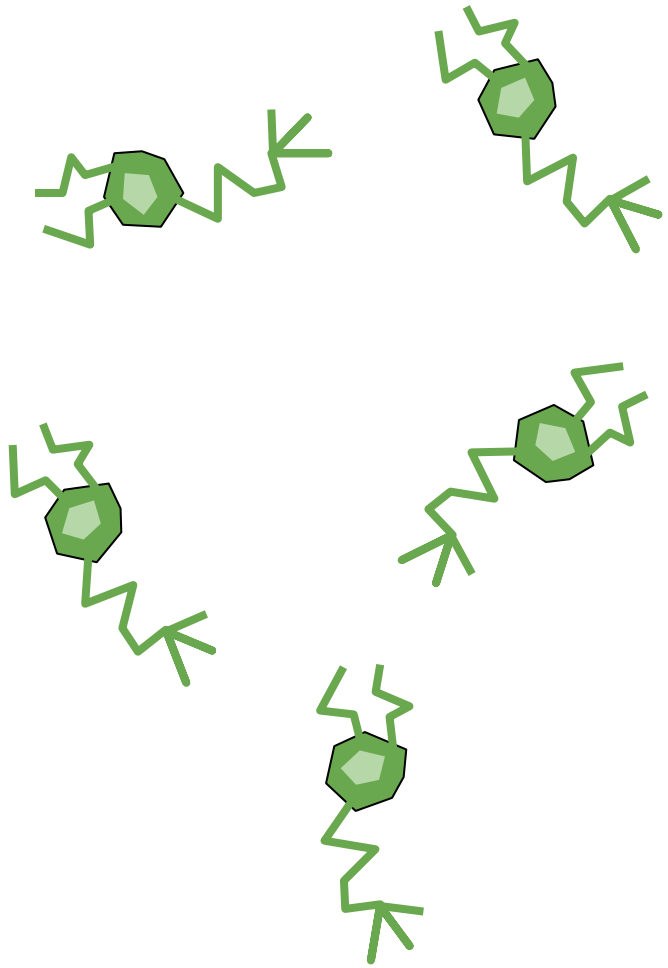
EVOLUCIÓN DIFERENCIAL



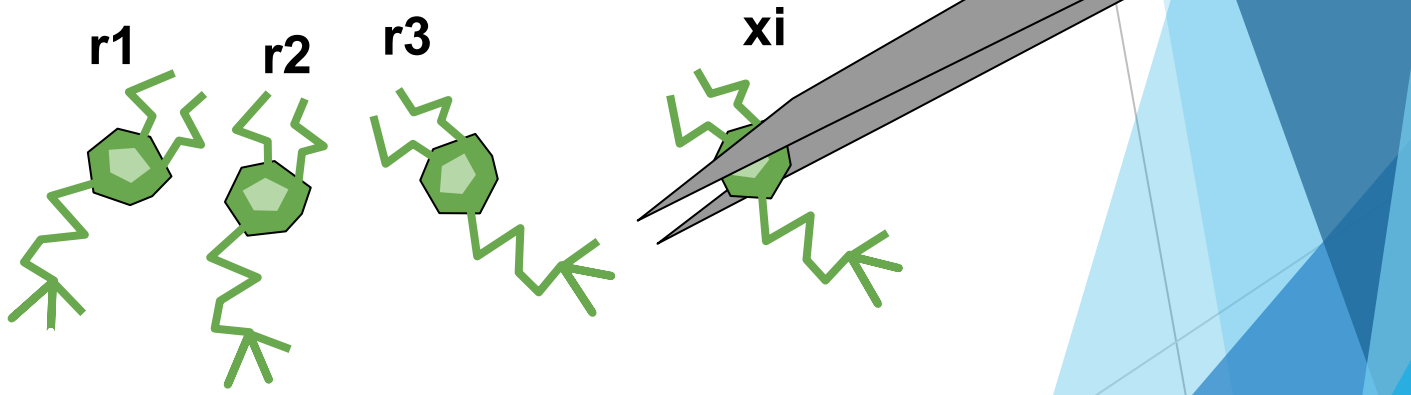
En el ciclo que calcula a la nueva generación vamos primero que todo a tomar a cada individuo i-ésimo uno por uno de la siguiente manera:



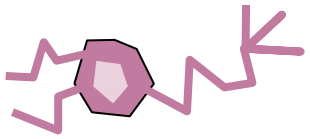
EVOLUCIÓN DIFERENCIAL



Seleccionamos al azar a otros tres individuos (r1, r2, r3) diferentes y mediante una operación matemática vamos a cruzar sus genes (pesos sinápticos), como si formaramos a un individuo sintético vi.

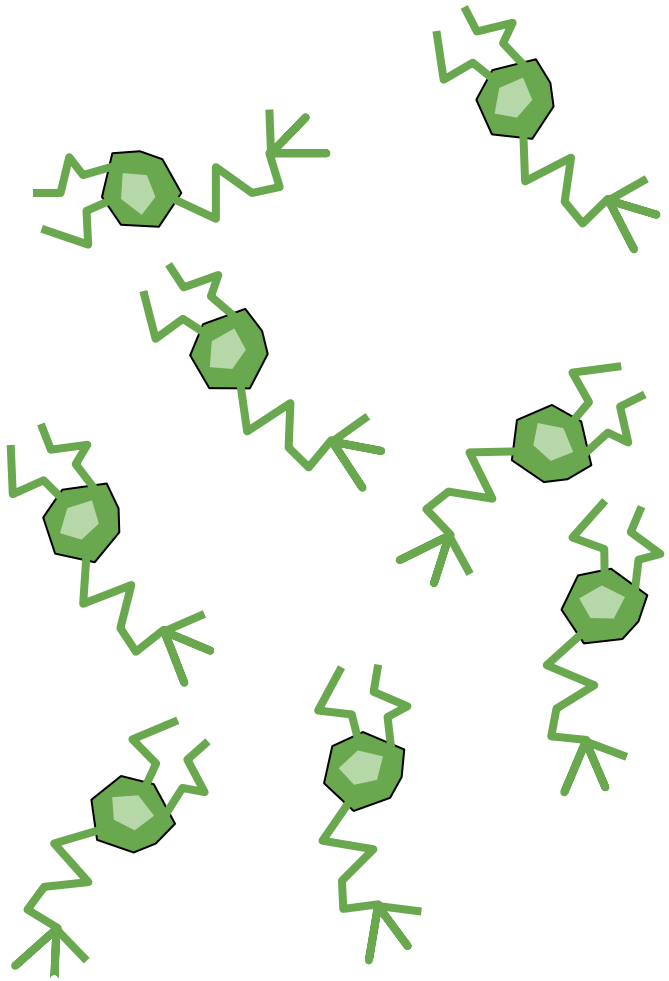


$$vi = x(r1) + h \,.\, (x(r2) - x(r3))$$

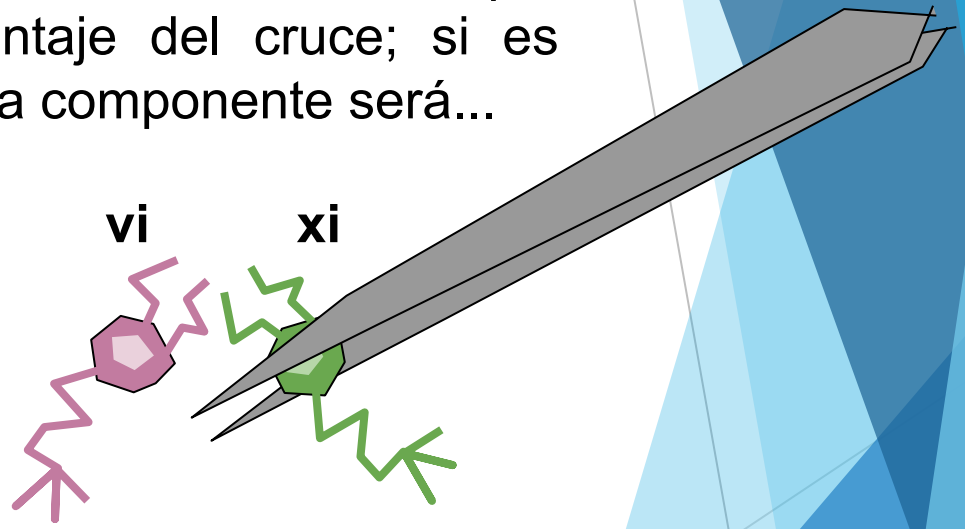


El parámetro “h” es para controlar la fuerza del cruce.

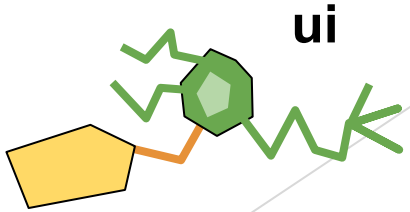
EVOLUCIÓN DIFERENCIAL



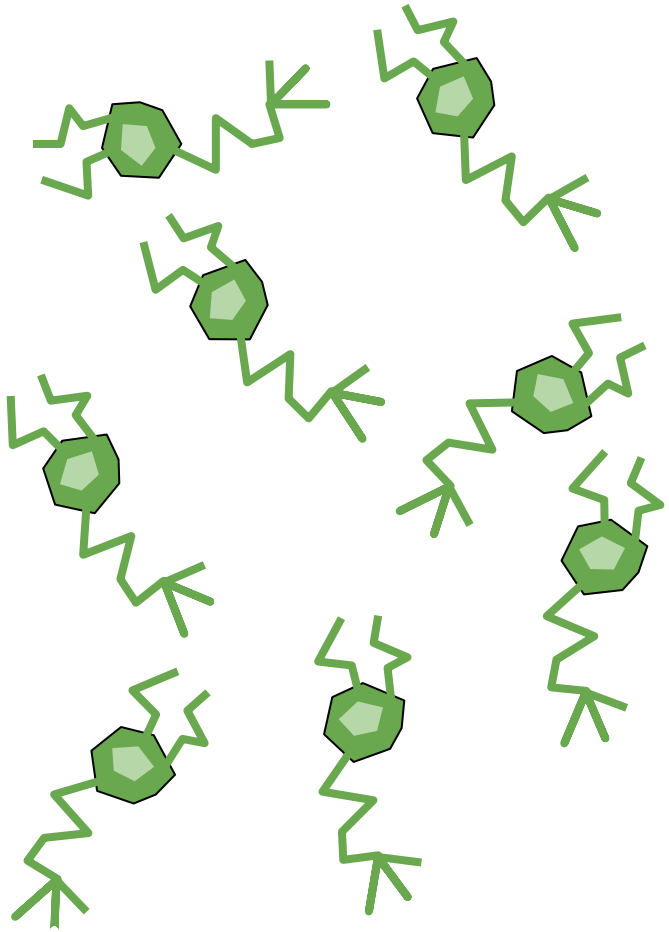
Luego creamos al individuo nuevo ui, recorriendo cíclicamente sus genes, para cada uno comparamos un valor aleatorio con una constante “c” que definirá el porcentaje del cruce; si es menor entonces la componente será...



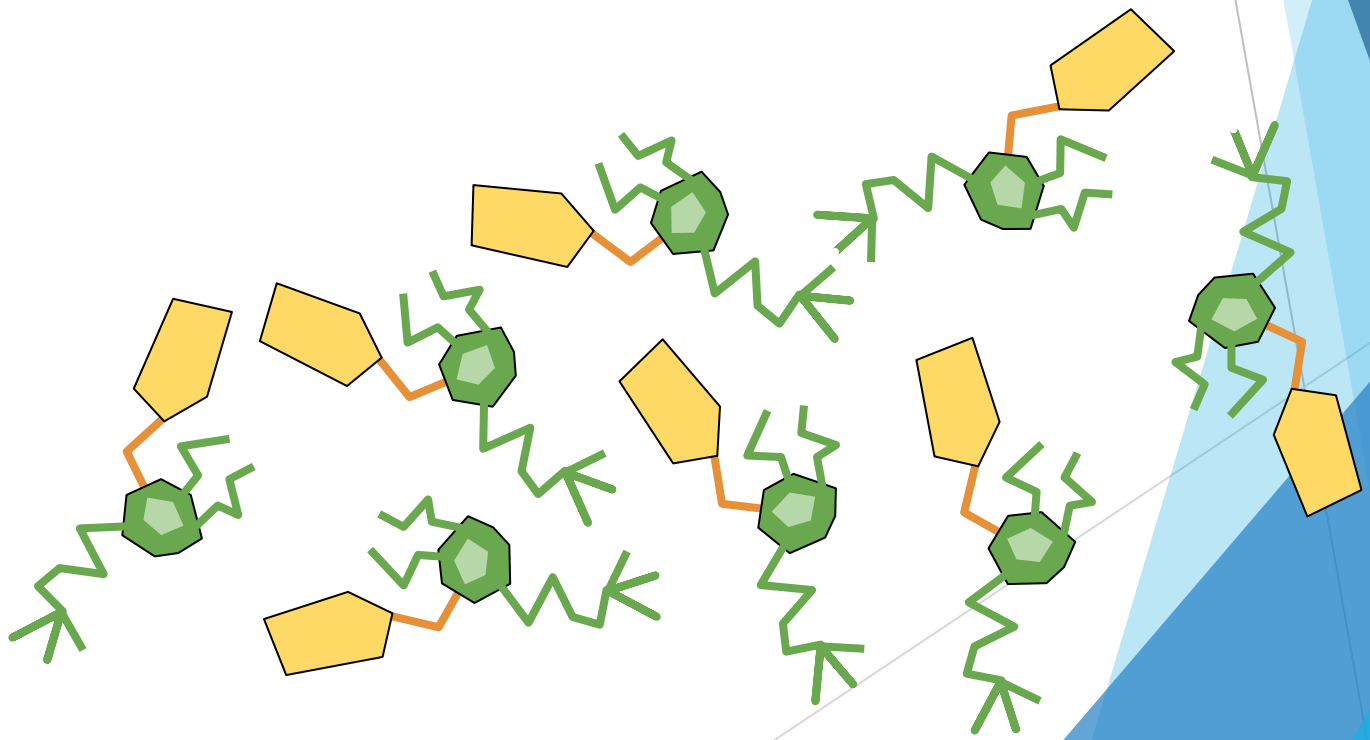
...la de vi, de lo contrario será la de xi.



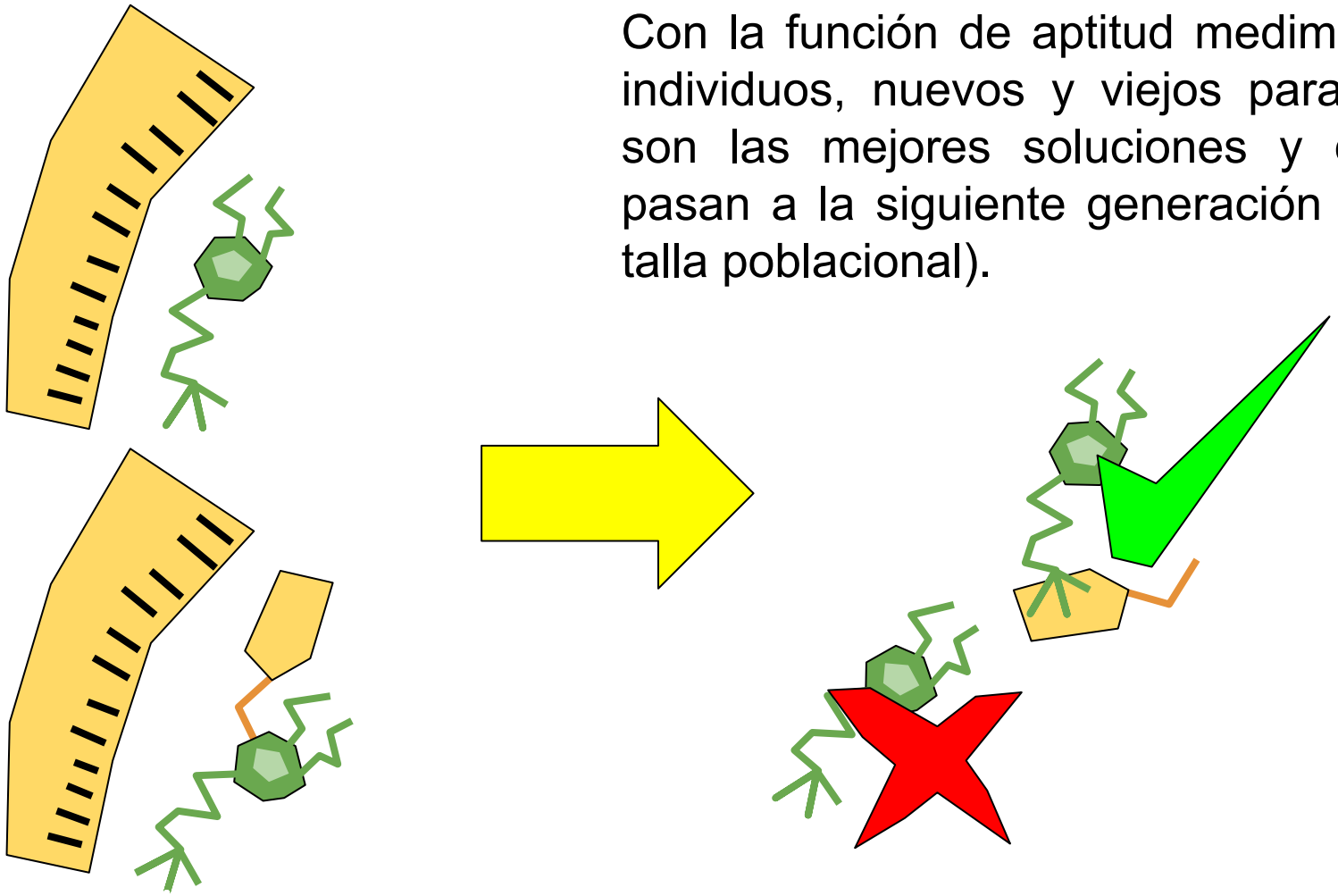
EVOLUCIÓN DIFERENCIAL



Después de hacer lo mismo para todos, tendremos una cantidad igual de “hijos”.



EVOLUCIÓN DIFERENCIAL

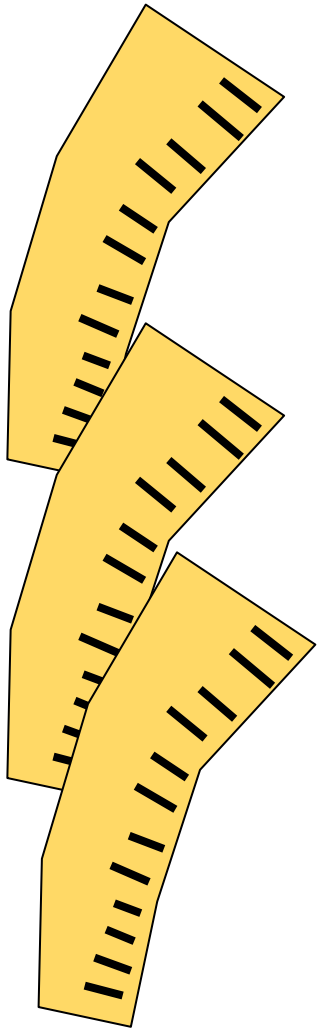


Con la función de aptitud medimos a todos los individuos, nuevos y viejos para saber cuales son las mejores soluciones y que por tanto pasan a la siguiente generación (que conserva talla poblacional).

Siempre mantenemos en mira a la solución (individuo) con menor error; en el caso del software de este proyecto solo esa será graficada.

El ciclo se repite hasta alcanzar error 0 o sea interrumpido.

EVOLUCIÓN DIFERENCIAL



Se utilizaron 3 métodos de cálculo de error, los tres operan todos los patrones, acumulan el error asociado y finalmente dividen por la cantidad de patrones.

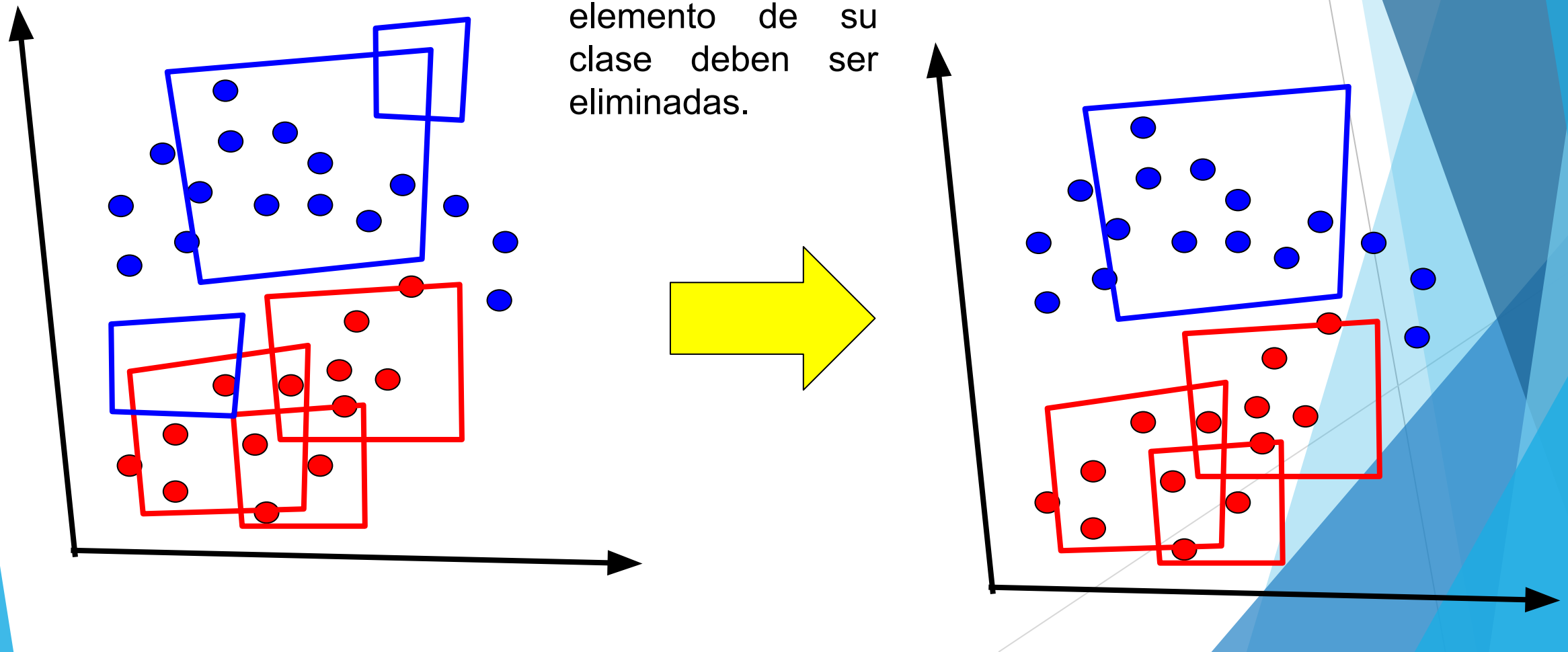
El primero compara directamente la salida deseada con la obtenida y si no coinciden suma 1 al error acumulativo.

El segundo está referenciado a 0, recorre todas las neuronas, si la neurona actual es la asociada a la clase deseada y su valor está por debajo de 0 entonces agrega a la neta del error el valor absoluto de esta salida; si no es la asociada a la clase deseada y su valor es mayor que 0 entonces sumará este valor a la neta del error.

El tercero es igual al segundo pero agrega solo 1nos al error, en lugar de todo el valor de la variable.

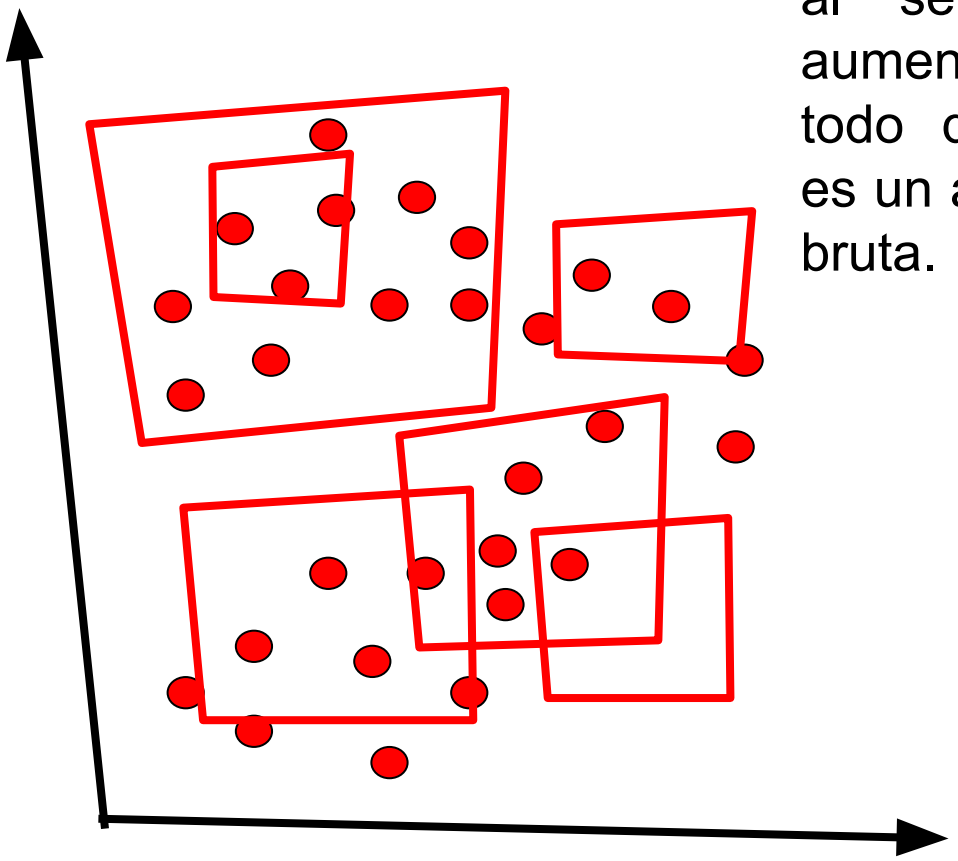
OPTIMIZACIÓN DENDRITAS

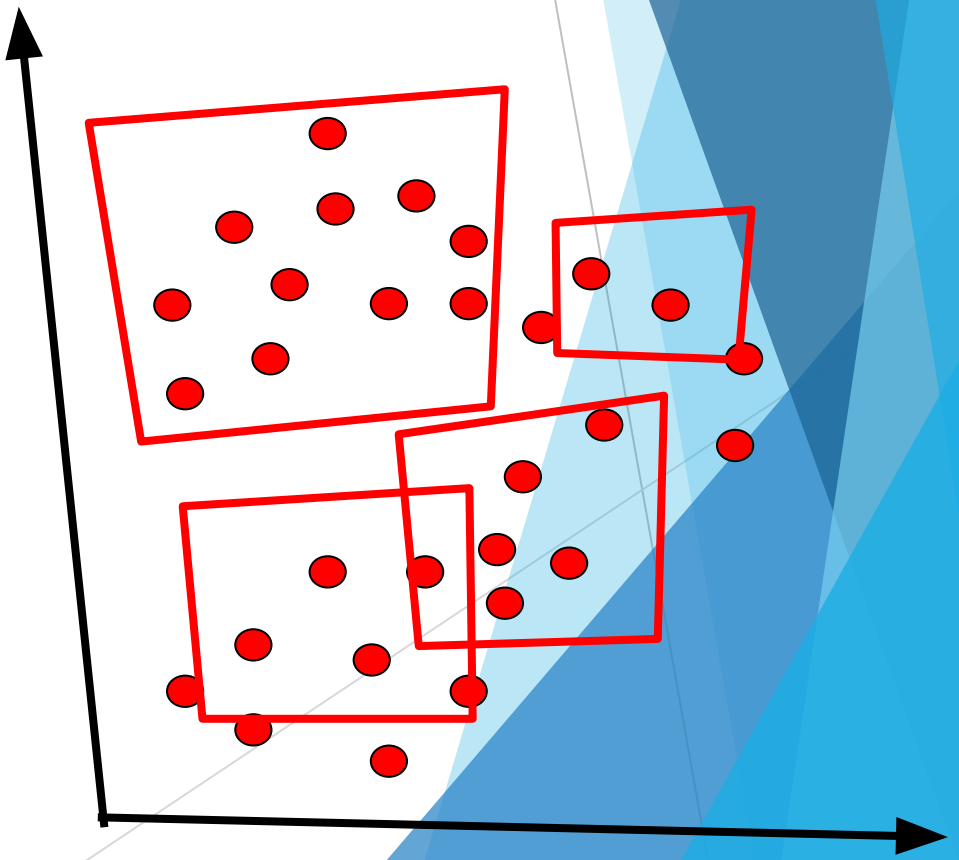
Cajas que no contienen ningún elemento de su clase deben ser eliminadas.



OPTIMIZACIÓN DENDRITAS

Ver primero las cajas más pequeñas, y mirar si al ser eliminadas No aumenta el error (es decir todo queda igual); esto es un algoritmo de fuerza bruta.





SOFTWARE

Construido en Game Maker Studio, tiene 4 paginas: creación de patrones, algoritmo k-means, algoritmo evolución diferencia, testeo.

Se prueban las DMNN con dos entradas x1, x2 por fácil graficación y visualización, pero el potencial de la ANN es hiperdimensional.