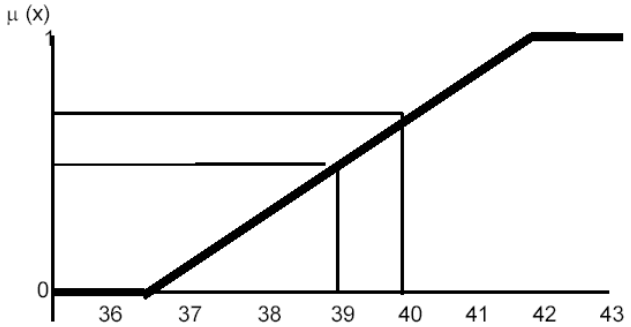
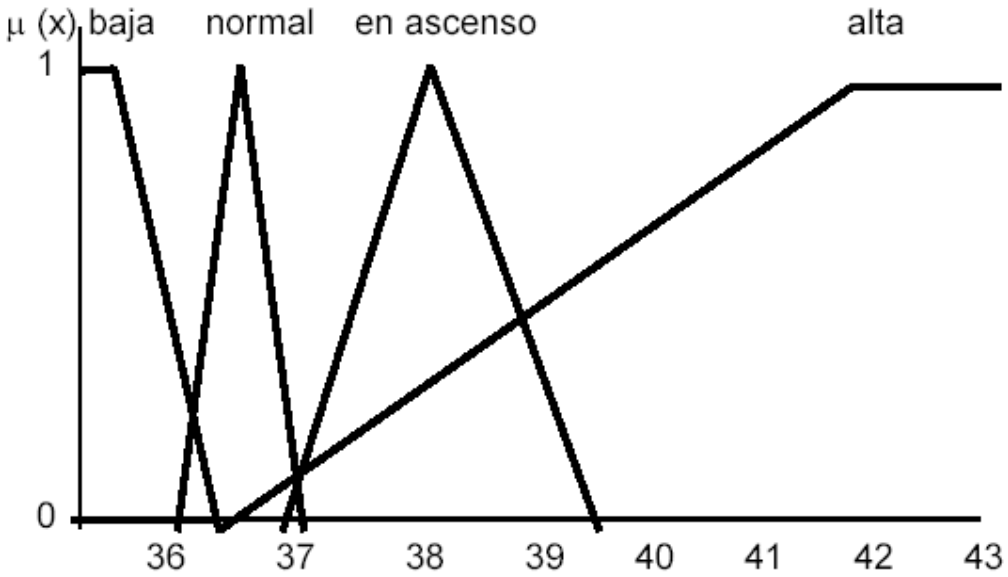
Lógica difusa- FUZZY

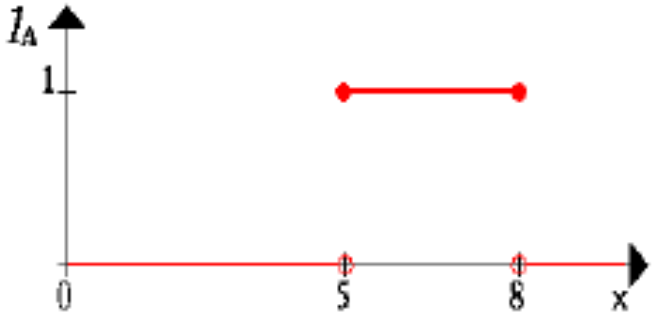
Conceptos de Conjunto Difuso:
Consideremos el caso de definir pacientes con fiebre alta" cuando es igual o superior a 42ºC , habría un conjunto de ellos con - fiebre alta - y otro –sin fiebre alta- , no obstante dentro de éste último conjunto de pacientes los habría con temperatura normal , otros con temperaturas levemente superior a la normal y otros hacia y con hasta casi los 42ºC , por lo que podríamos decir que cada uno de ellos tendría un *cierto grado* de *fiebre alta* , formando un degradé donde es *difuso* cuanto de *fiebre alta* tiene cada uno (de allí el nombre) , salvo el de temperatura *normal* que *no participaría* de este *conjunto difuso* o lo que es lo mismo, que lo *hace* con un *grado cero*.
Si consideramos *todo el conjunto* de pacientes podríamos decir que cada de ellos *participa* con un cierto *grado* de *pertenencia* al subconjunto denominado de *fiebre alta ,* a los grado de pertenencia los denominaremos con la letra μ (x) Estos *grado de pertenencia* puede ser representada por una *función continua,* haciéndose notar que se podría decir que un paciente con 39 ºC tiene un *poco* de *fiebre alta* y otro con 40 º C con *casi tiene fiebre alta*. A través de esta función continua asociamos ciertos grados de pertenencias a ciertas expresiones lingüísticas (*poco , casi* , etc.)



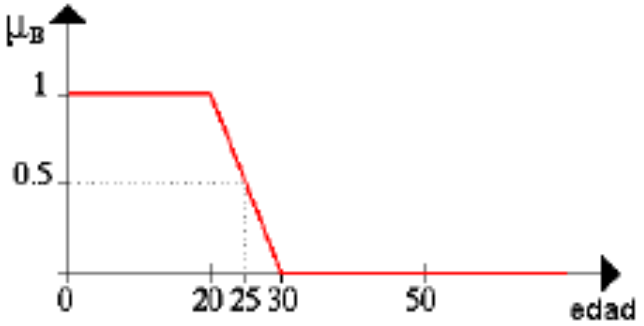
Definimos ahora al conjunto *temperatura alta* como *término lingüístico* que junto con otros términos lingüísticos asociados con la temperatura de los pacientes, como baja, normal y en ascenso y representadas por sendas funciones continuas representando así en forma completa a la variable *lingüística fiebre*. La variable lingüística fiebre nos permite traducir las distintas medidas de la temperatura corpórea , dada en grados centígrados en descripciones lingüísticas.
Por ejemplo, una temperatura corpórea de 37,8 ºC podrá ser evaluada como temperatura en ascenso, casi un poco de alta.
Palabras claves: grado de pertenencia μ (x) , término lingüístico, función continua, variable lingüística.



Conjuntos difusos

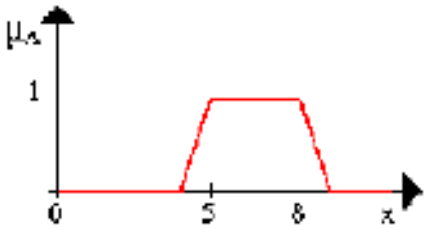


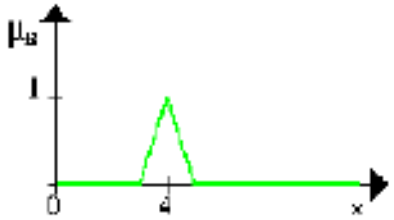
Conjunto de niños



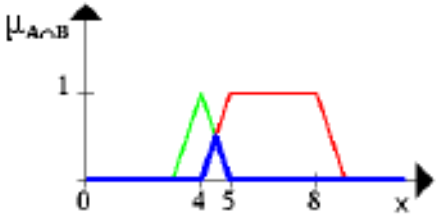
Conjunto de gente joven

Operaciones con conjuntos difusos

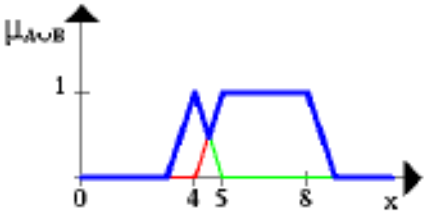




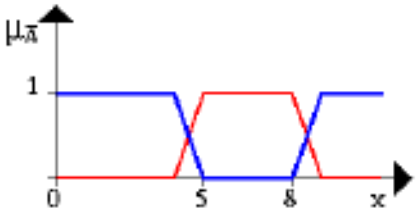
AND

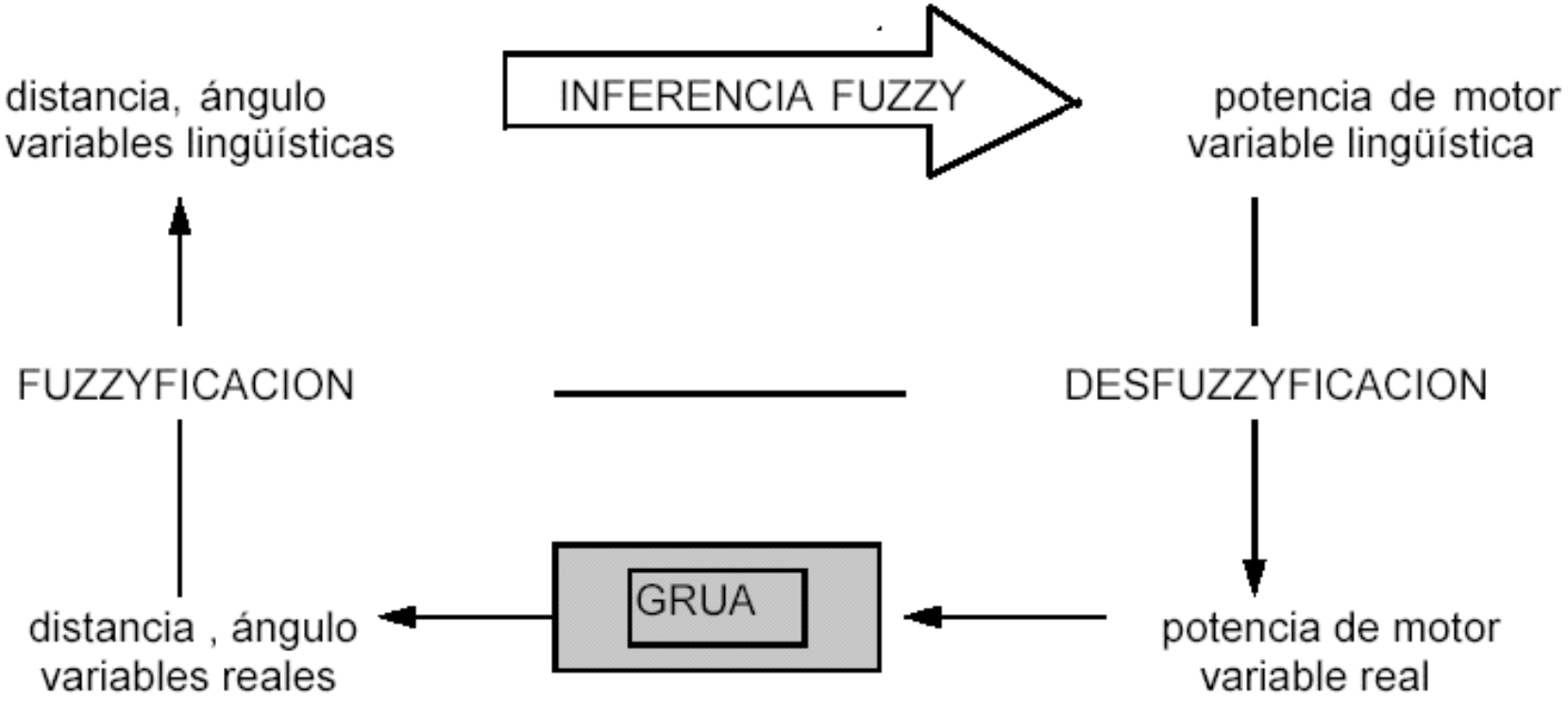


OR



NEGACION DE A





– ***Fuzzification*** **— Borrosificación:**

• Designa a la operación que permite convertir un conjunto nítido en borroso.

– ***Defuzzification*** **— Desborrosificación :**

• Designa a la operación que permite convertir un conjunto difuso en clásico.

– **Función de pertenencia - *Membership function*:**

• Todos los elementos dentro del universo de discurso son miembros de algún conjunto difuso en cierto grado. La función de pertenencia es la curva que define con que grado cada elemento está incluido en el conjunto difuso.

– **Variables lingüísticas - Linguistic variables:**

• Combinan múltiples categorías subjetivas que describen el mismo concepto, así, para el caso de la variable altura existirán las categorías: bajo, mediano, alto y muy alto, que son los términos lingüísticos y representan los posibles valores de una variable lingüística.

• Tiene tres parámetros (x, T(x), U) donde x es el nombre de la variable, T(x) es el conjunto de términos lingüísticos de x y U es el universo.

– **Grado de pertenencia:**

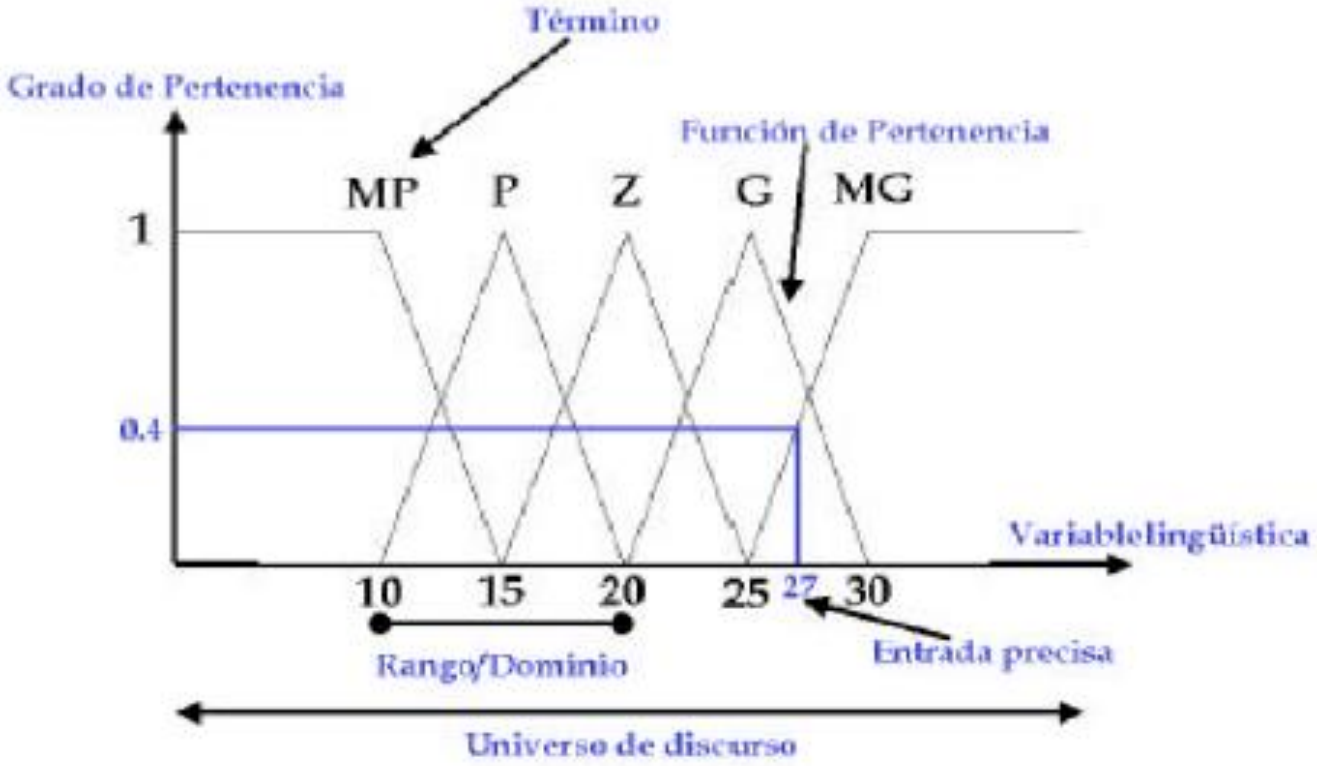
• Es el grado con el que una entrada ***crisp*** es compatible con la función de pertenencia, puede tomar valores entre 0 y 1.

– **Término:**

• Es una categoría subjetiva de una variable lingüística, y consecuentemente, es el nombre descriptivo usado para identificar una función de pertenencia.

– **Rango/Dominio:**

• Es el intervalo sobre el cual se define una Función de Pertenencia. Por ejemplo, una función de pertenencia Alto podría tener un dominio de 1.60 a 1.9 m y su rango sería de 0.3 m



Los **conjuntos difusos** introducidos son una forma matemática para **representar la vaguedad en lingüística**, puede ser considerada como una generalización de la teoría de conjuntos clásica.

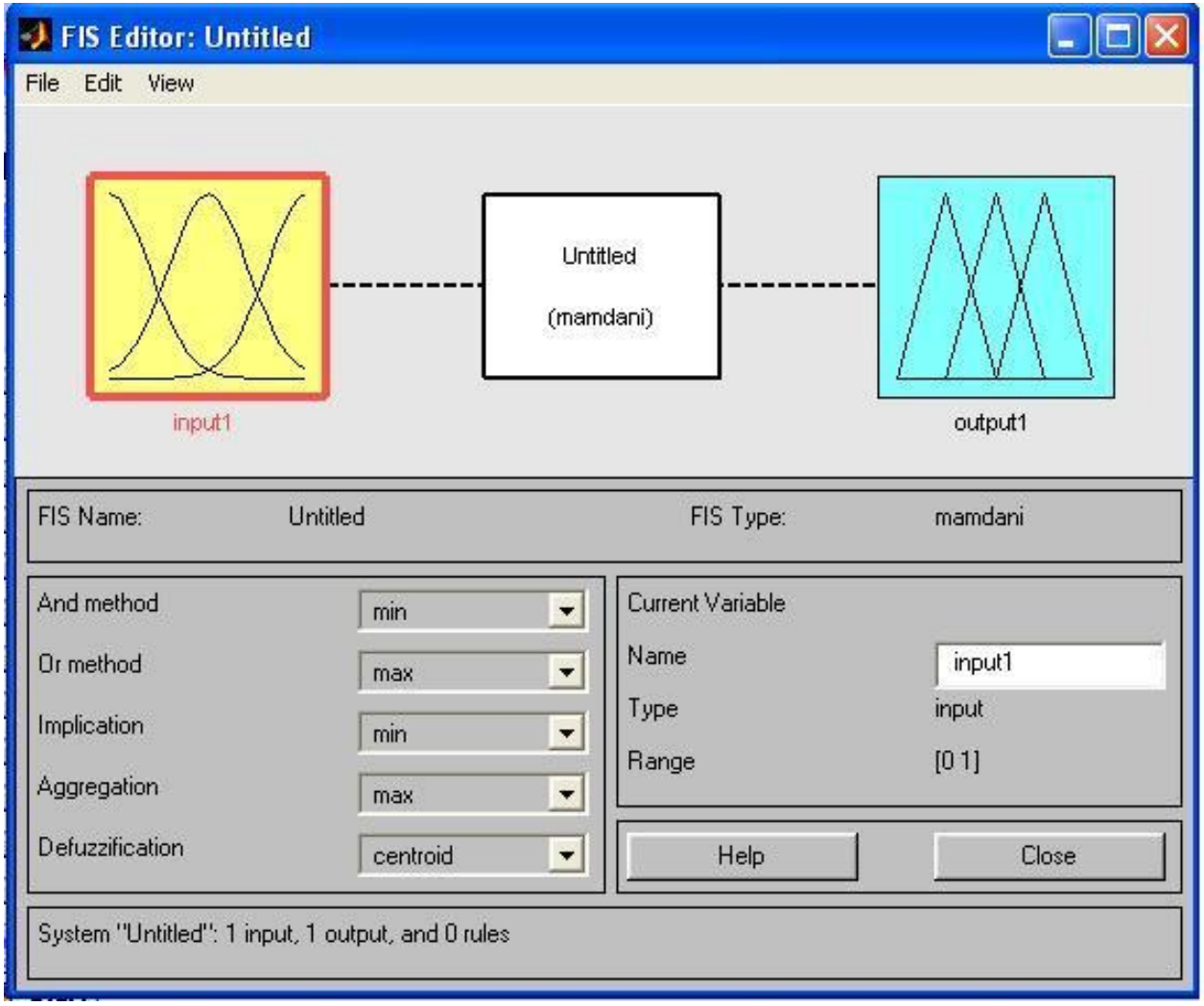
Una de las grandes diferencias entre los **conjuntos clásicos** y difusos es que el primero siempre tiene **funciones de pertenencia únicas**, mientras que todo **conjunto difuso** tiene un **número infinito de funciones de pertenencia** que pueden representarlo.

– **Esto permite que los sistemas difusos puedan ser ajustados a su rendimiento máximo en una situación dada.**

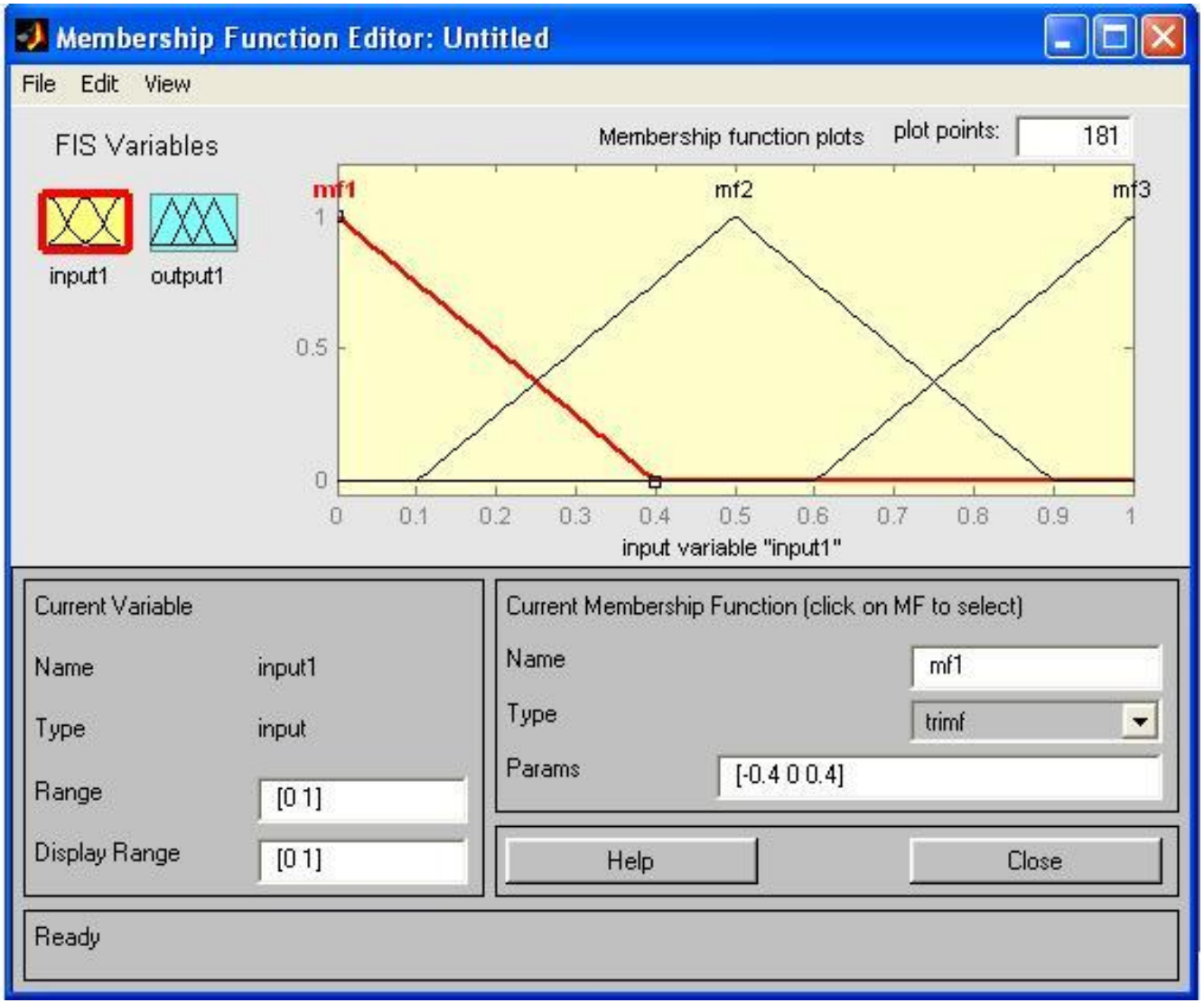
Para obtener un controlador difuso, primero debe crearse un archivo FIS (Fuzzy Inference System), con la ayuda del *FIS Editor* que posee MatLab.

Las diferentes etapas de la creación de un archivo FIS son las siguientes:

1º_ Definición del Sistema de Inferencia



2º_ Definición de las funciones de membresía



3º_ Definición de las reglas difusas

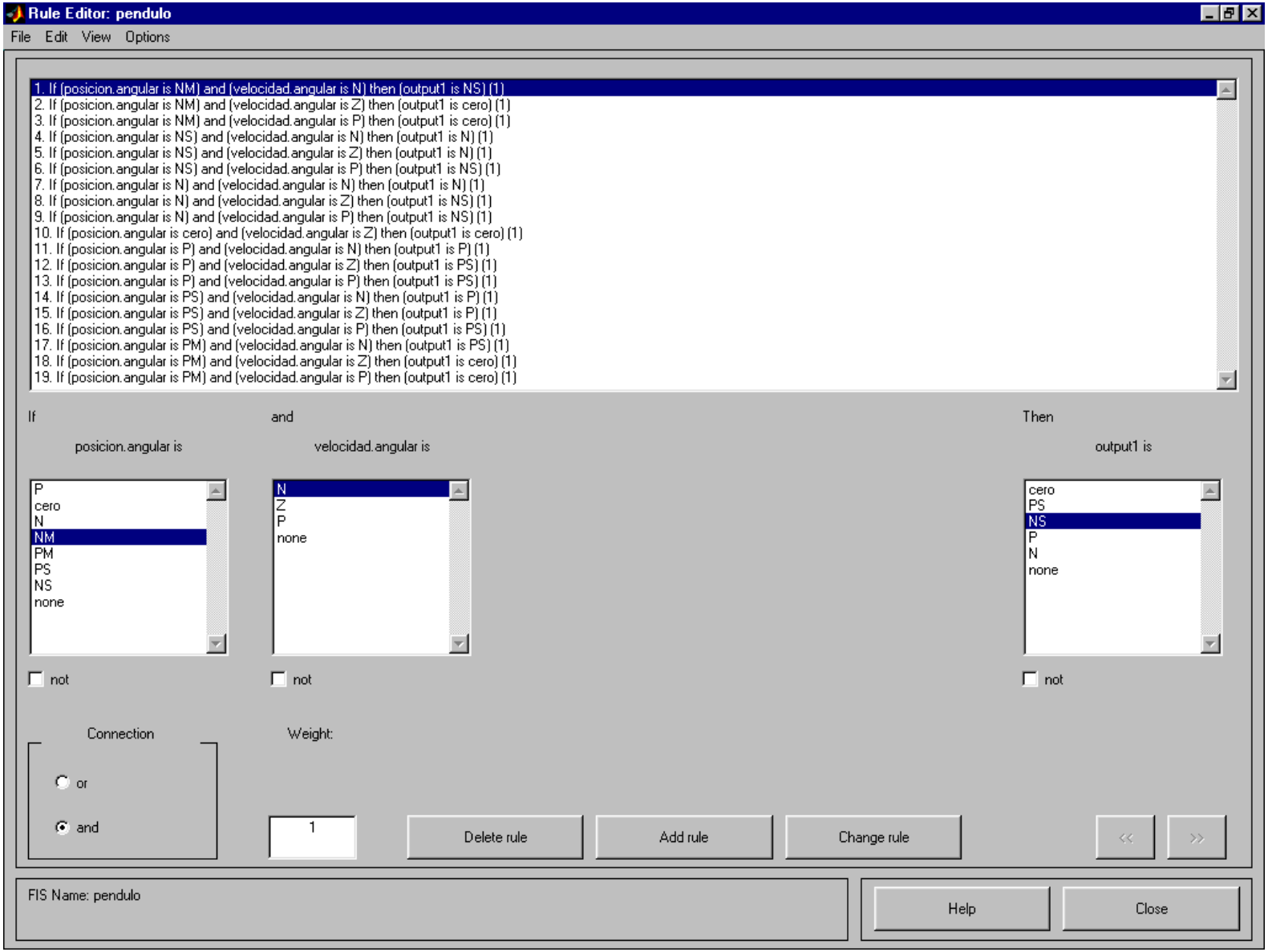
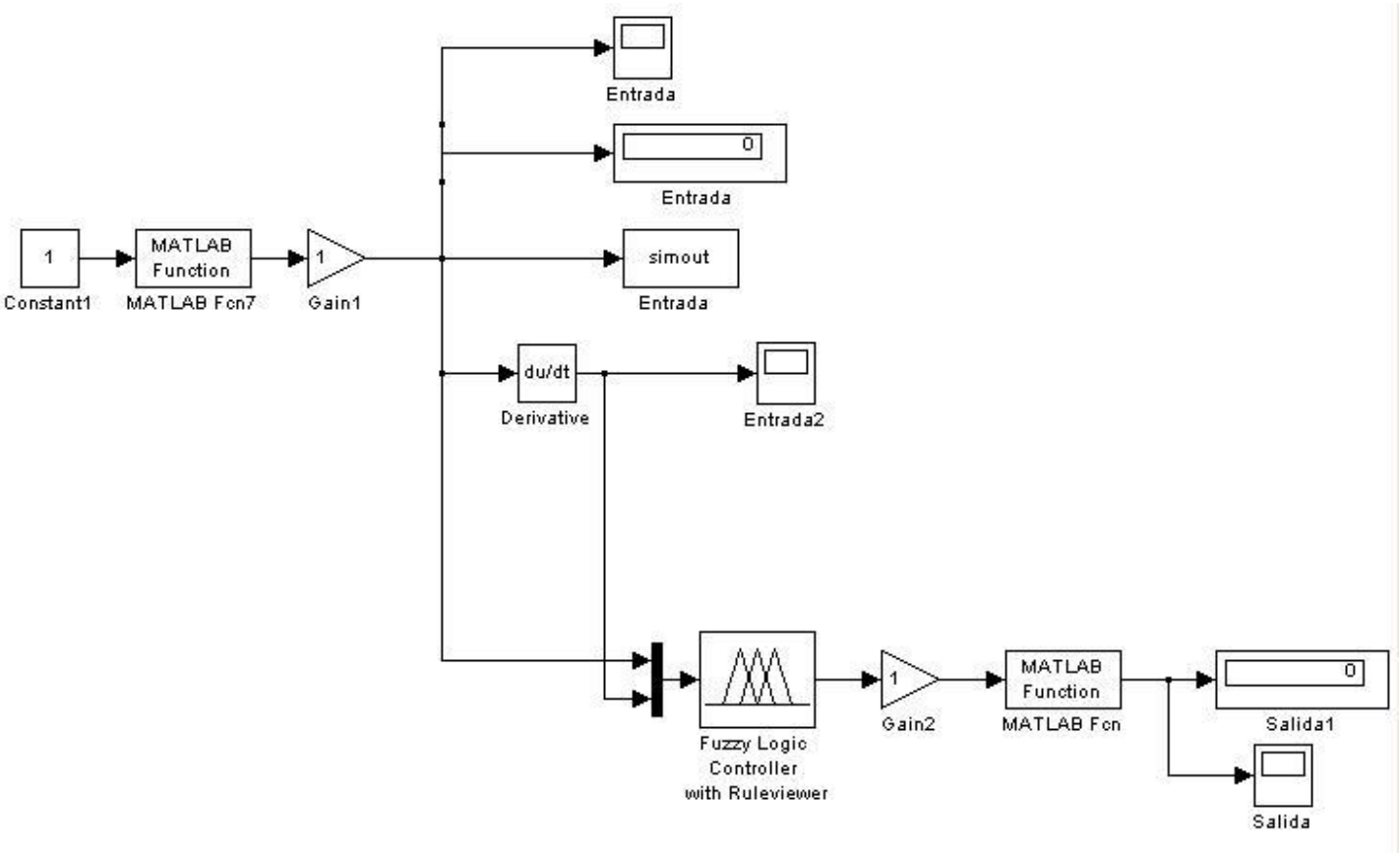
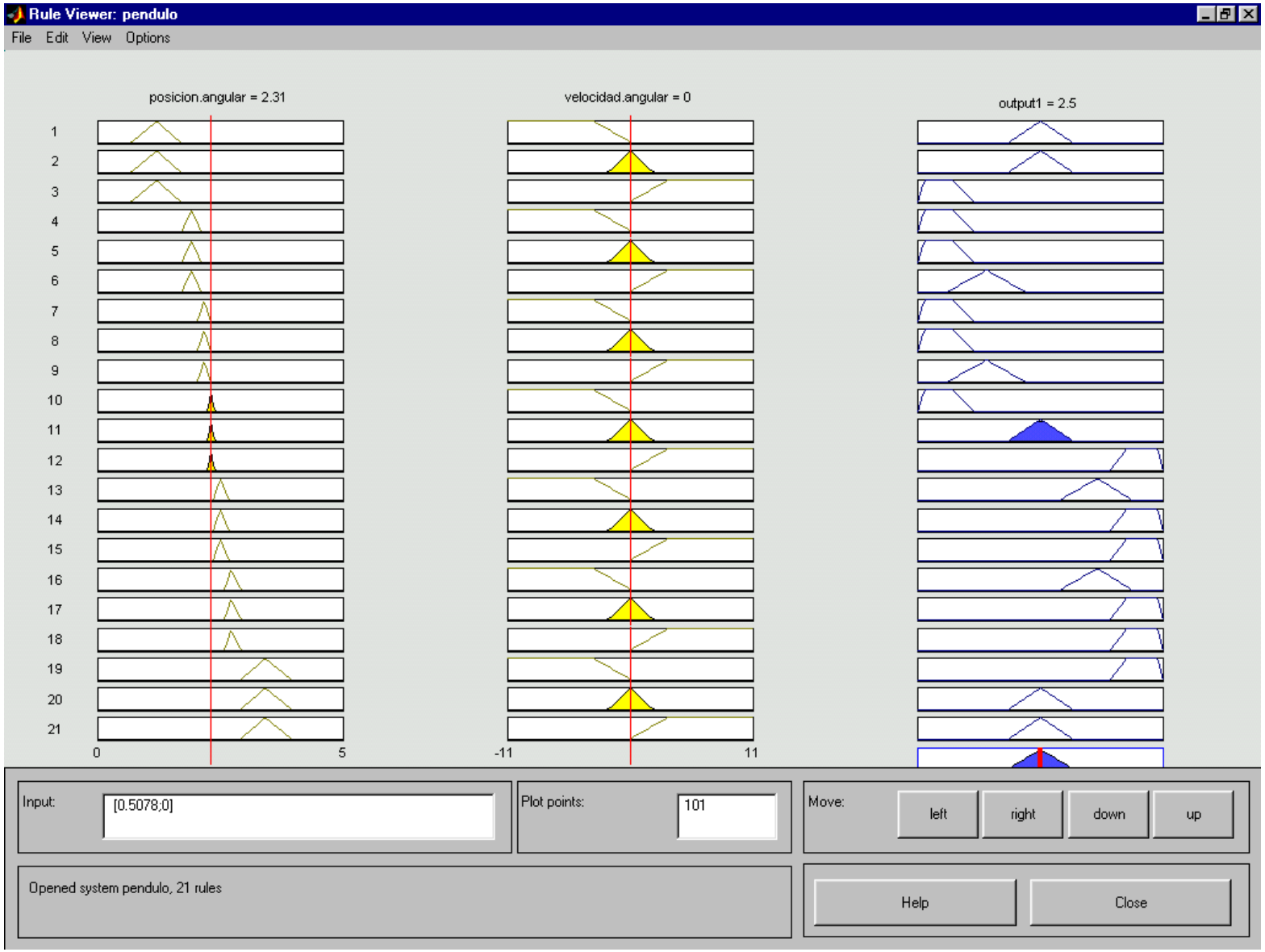


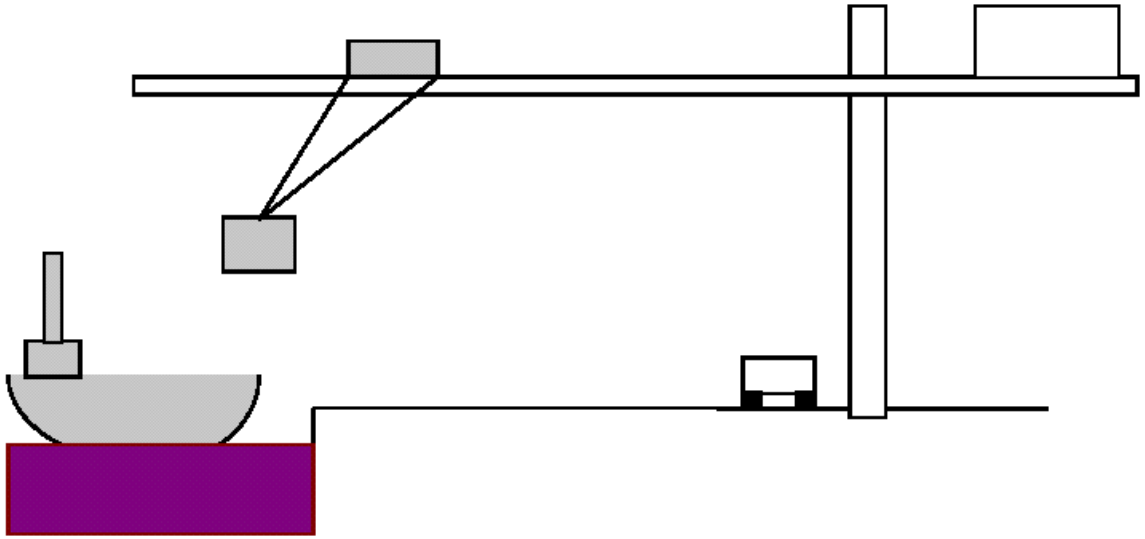
Diagrama de Bloques para el Control por Lógica Difusa:



Visualización de las reglas difusas implementadas:



Las grúas de containers se usan para descargar y cargar de y hacia barcos en la mayoría de los puertos . Una grúa levanta un container mediante cables flexibles que penden desde un extremo de un gran barral horizontal, en el otro está la cabina de comando y control.
Levantado lo suficiente y sobre rieles montados en el gran barral, un carro desplaza el ahora colgante container en forma horizontal hasta un vagón de tren, donde lo deposita.



Las grúas cuentan con un motor eléctrico cuya velocidad puede ser controlada por el operador, que si está bien entrenado es capaz de compensar el balanceo del container y descargar el mismo sobre el vagón en un tiempo aceptable .

Por otro lado veamos como controla el operador : una vez que ha levantado el container, comienza el movimiento horizontal del carro a *media velocidad* , para ver como se balancea el container , dependiendo de dicha reacción ,el ajusta la velocidad de tal forma de dejar un *poco atrás* el container del carro , a partir de ese momento da *máxima velocidad* al motor que impulsa el carro comprobándose mínimo balanceo en esta etapa.

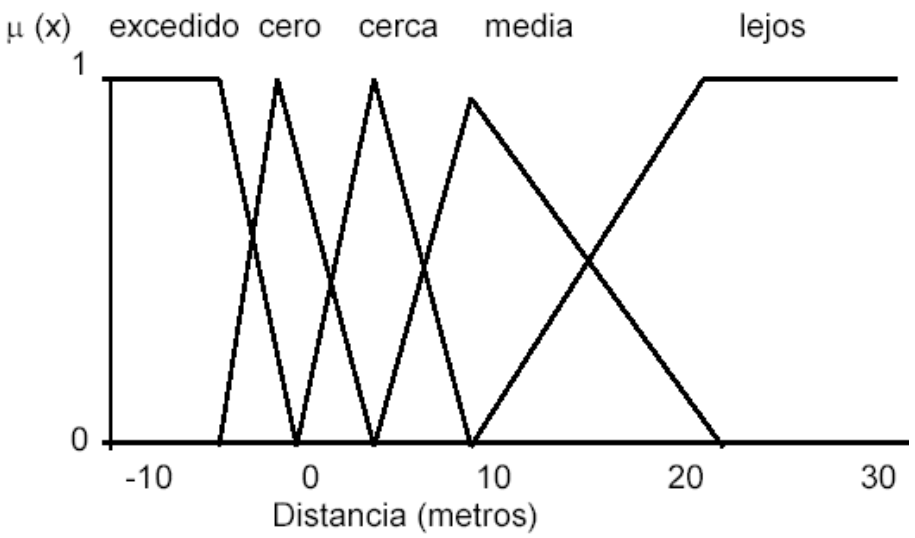
Estando *cerca* del vagón el operador *reduce* la velocidad o aún más , aplica *potencia negativa* para frenar el carro. Hecho esto se ve que el container se adelanta levemente al carro que lo desplaza , a partir de este punto se incrementa la velocidad del motor hasta que el carro está encima del container que no se balancea y ha quedado debajo del vagón.

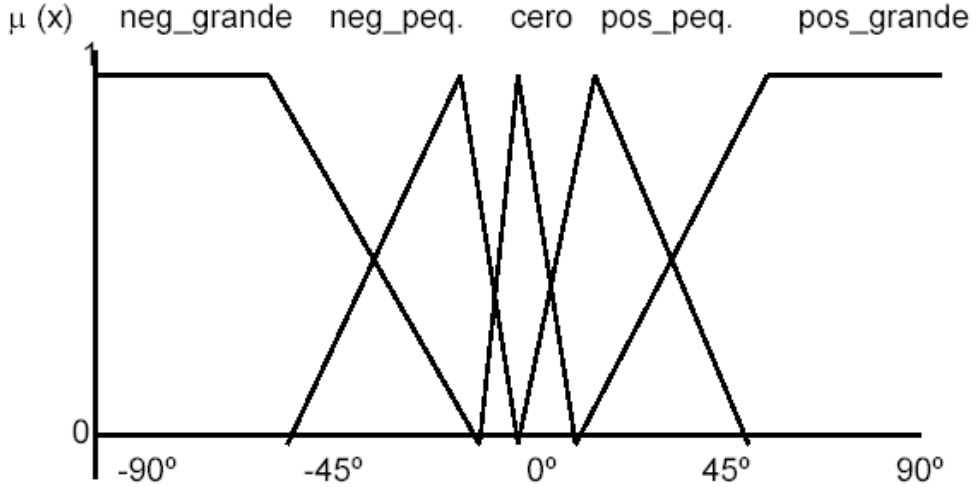
Como vemos las acciones del operador es capaz de compensar los disturbios y las alinealidades del sistema . Un análisis de estas acciones revelan y describen su estrategia de control:

1) Empieza a desplazar el carro con motor a media potencia
2) Estando en esta etapa , lejos del vagón , deja un poco atrás al container y aumenta al máximo la velocidad .
3) Cerca del vagón reduce la velocidad del carro hasta que el container se adelanta un poco respecto del primero.
4) Cuando el container está cerca del vagón , incrementa la velocidad del motor.
5) Cuando el container está sobre el vagón y el balanceo es nulo, para el Motor.

1) IF distancia = lejos AND ángulo = cero
THEN potencia = pos_media (media potencia positiva - hacia el vagón)
2) IF distancia = lejos AND ángulo = neg_pequeño
THEN potencia = pos_maximo (máxima potencia positiva)
3) IF distancia = lejos AND ángulo = neg_grande
THEN potencia = pos_media
4) IF distancia = media AND ángulo = neg_pequeño
THEN potencia = neg_media (media potencia negativa - hacia barco)
5) IF distancia = media AND ángulo = cero
THEN potencia = cero
6) IF distancia = media AND ángulo = pos_pequeño
THEN potencia = pos_media
7) IF distancia = cerca AND ángulo = pos_pequeño
THEN potencia = pos_media
8) IF distancia = cero AND ángulo = cero
THEN potencia = cero

VARIABLE TERMINOS
distancia { lejos , media , cerca , cero , excedido }
ángulo { pos_grande , pos_pequeño , cero , neg_pequeño , neg_grande }
potencia { pos_max , pos_media , cero , neg_media , neg_max }





Veamos ahora la respuesta del controlador fuzzy ante una situación producida en un instante dado , por ejemplo que el container está a distancia de 12 metros del vagón y con una inclinación de 4º (está un poco adelantado respecto de la vertical) en su recorrido hacia el vagón .
La relación que se establece a partir de considerar esta variables , sus términos y grados de pertenencia (ver gráficos) serán:

VARIABLE	TERMINOS
distancia	{ 0.1 , 0.9 , 0 , 0 , 0 }
ángulo	{ 0 , 0.8 , 0.2 , 0 , 0 }

Además vemos que las reglas de aplicación en el formato IF - THEN son las:1), 5) y 6) , que para su aplicación debemos definir dos nuevos conceptos:
Agregación: hace a la evaluación y cálculo de la parte IF de la regla.
Composición: hace a la evaluación y cálculo de la parte THEN de la regla.
Agregación: Debemos decir que la función conectiva AND expuesta en la regla IF - THEN no es la booleana pura sino una que contempla las condición difusa de " mas o menos " , para ello este operador AND en lógica difusa se expresa en términos de tomar como salida de dicha operación el mínimo dado por alguno de los dos operandos, así tenemos:

$$\text{AND}: \quad \mu_{A \wedge B} = \min \{ \mu_A , \mu_B \}$$

Esto se traduce a las reglas IF - THEN que tienen aplicación para este instante, como:

1) IF distancia = lejos AND ángulo = cero ≡ min { 0.1 ; 0.2 } = 0.1

5) IF distancia = media AND ángulo = cero ≡ min { 0.9 ; 0.2 } = 0.2

6) IF distancia = media AND ángulo = pos_pequeño ≡ min { 0.9 ; 0.8 } = 0.8

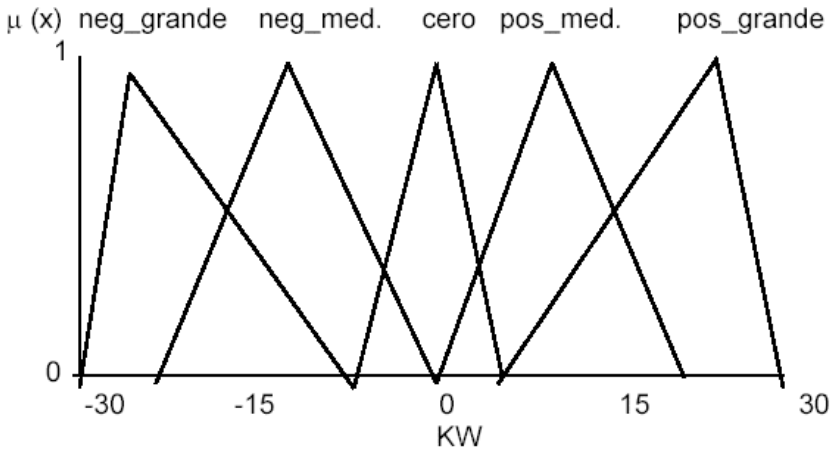
Mínima desviación de las reglas ⟶ Mayor ganancia en la corrección

Vemos que las reglas 1) y 6) indican potencia positiva media , pero con distinto grado de "verdad " mientras que la 5) indica potencia nula , por lo que el paso de inferencia se realiza mediante el operador OR , no el booleano , sino el difuso definido como:

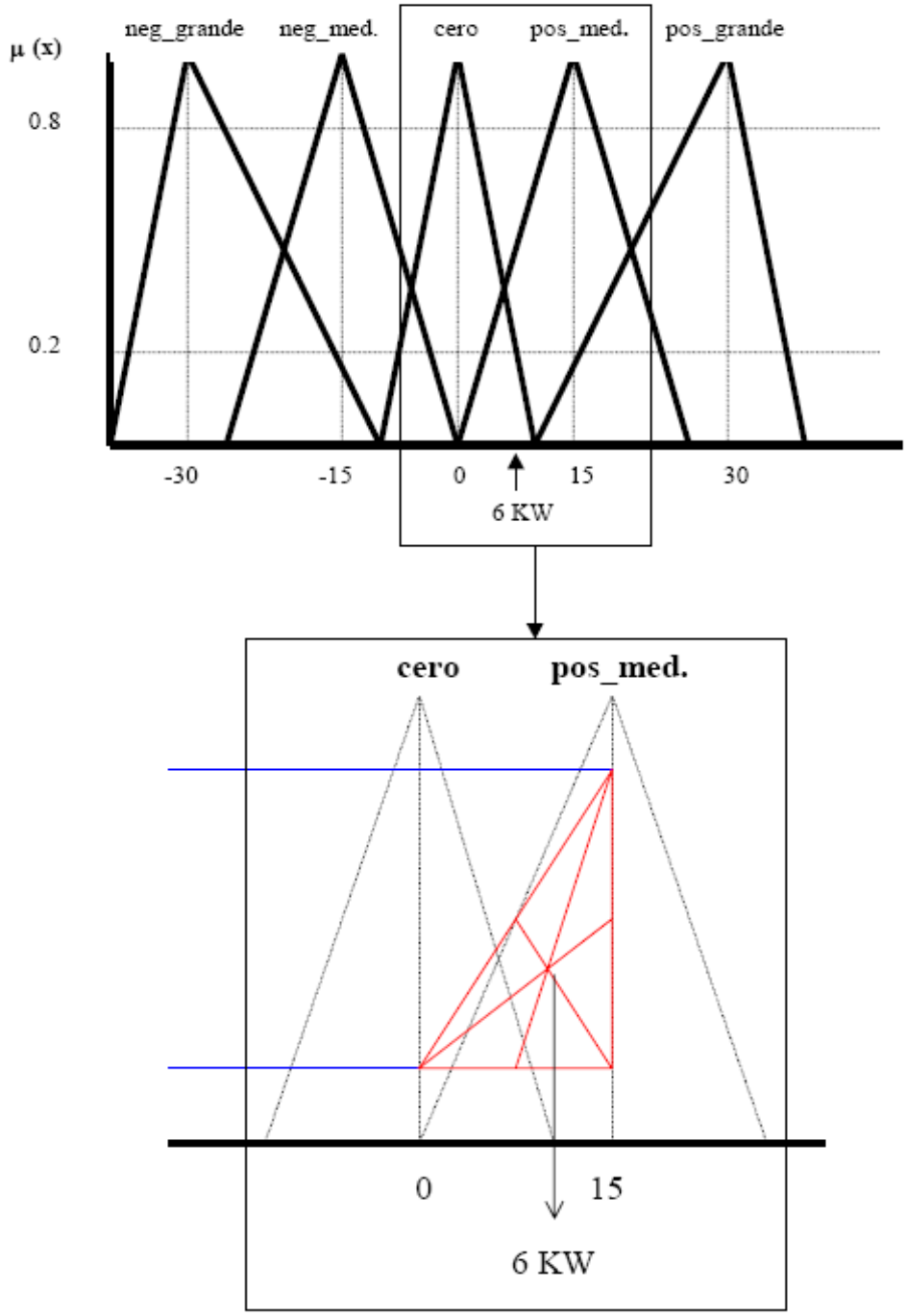
$$OR: \quad \mu_{A \cup B} = \max \{ \mu_A , \mu_B \}$$

Esto se hace que la variable lingüística *potencia* y sus términos , para este instante, resulten en:

pos_grande con un grado de pertenencia 0.0
pos_media con un grado de pertenencia 0.8 (= max { 0.8 ; 0.1 })
cero con un grado de pertenencia 0.2
neg_media con un grado de pertenencia 0.0
neg_grande con un grado de pertenencia 0.0



Mínima desviación de las reglas ⟶ Mayor ganancia en la corrección


μ (x)
0.8
0.2
neg_grande
neg_med.
cero
pos_med.
pos_grande
-30
-15
0
15
30
6 KW
cero
pos_med.
0
15
6 KW

Redes neuronales artificiales

EL MODELO BIOLOGICO

El cerebro humano contiene alrededor de10^{11} neuronas, su célula fundamental.
Cada una de estas neuronas procesa y se comunica con millares de otras en forma continua y paralela. La estructura de las neuronas (nodos), la topología de sus conexiones y el comportamiento conjunto de estas unidades, conforman la base para el estudio de las RNA.

Las redes neuronales naturales posibilitan algunas de las funciones que son patrimonio de la biología como: reconocer patrones, relacionarlos, usar y almacenar conocimiento por experiencia, además de interpretar observaciones.

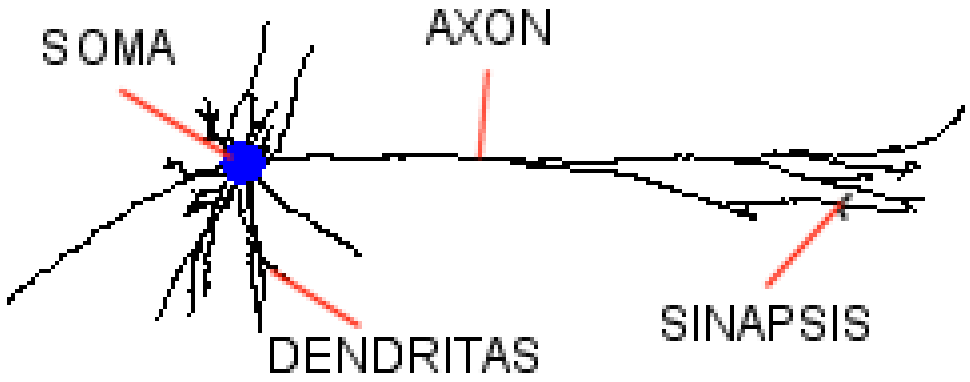
Si bien aún hoy no conocemos con precisión el detalle de las funciones cerebrales sí se sabe de la estructura fisiológica básica y es la que ha dado pie al desarrollo de las RNA.

Las RNA intentan reproducir las funciones de las redes biológicas tratando de emular su comportamiento y dinámica. No obstante de ser bastante diferentes entre ellas.

Como característica común los dos sistemas se basan en unidades de cálculo: las *neuronas*, paralelas y distribuidas que se comunican mediante conexiones llamadas *sinapsis*.

LA NEURONA BIOLOGICA

De forma esquemática una neurona biológica se compone de tres partes principales: el cuerpo de la célula (soma), las dendritas y el axón.



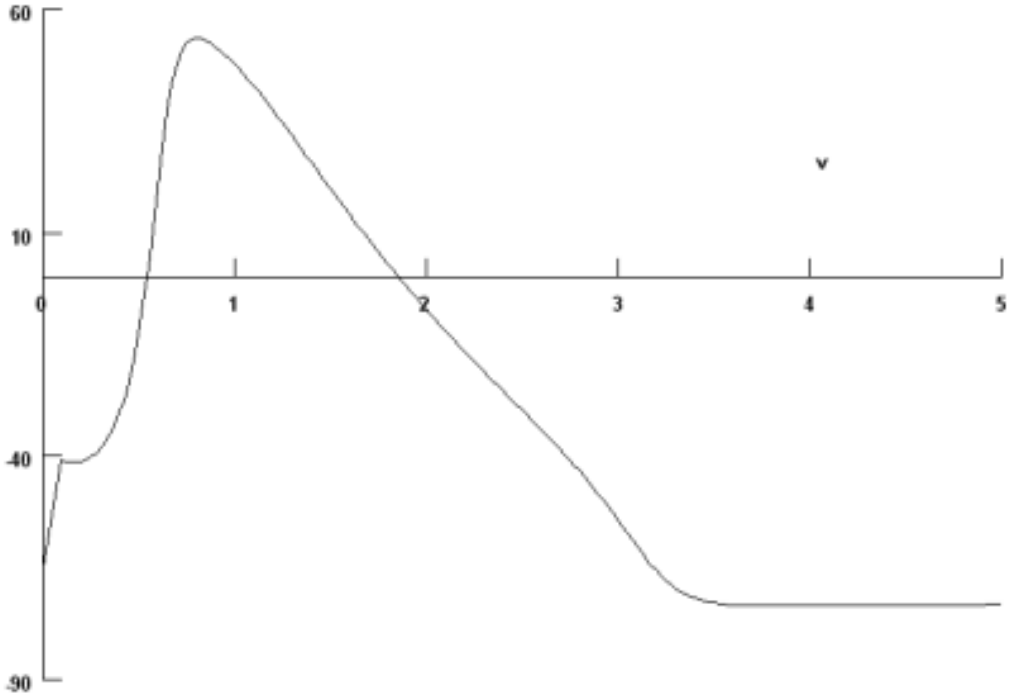
Las *dendritas* tienen como función recibir la información en forma de impulsos nerviosos desde otras neuronas y conducirlos al cuerpo celular el que los procesa, generando como resultado, nuevos impulsos. Estos luego, a través del *axón*, son conducidos a las dendritas de otras neuronas.

En las sinapsis las neuronas se unen funcionalmente formando una red y es aquí donde se produce el control de la transmisión de los impulsos o sea el flujo de información entre ellas.

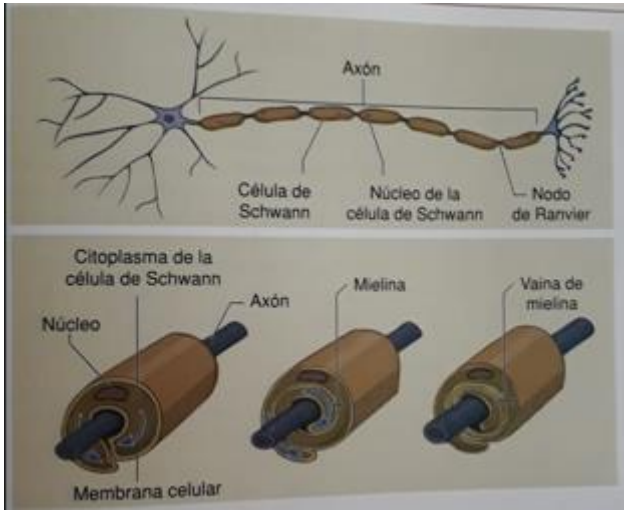
Las señales procedentes de la neurona presináptica son pasadas a la neurona postsináptica donde son sumadas con otras señales similares recibidas por la misma. Si el resultado obtenido, en un intervalo dado de tiempo, es lo suficientemente alto como para superar un nivel de *umbral*, la célula dispara un impulso que es transmitido a la célula siguiente (neurona postsináptica).

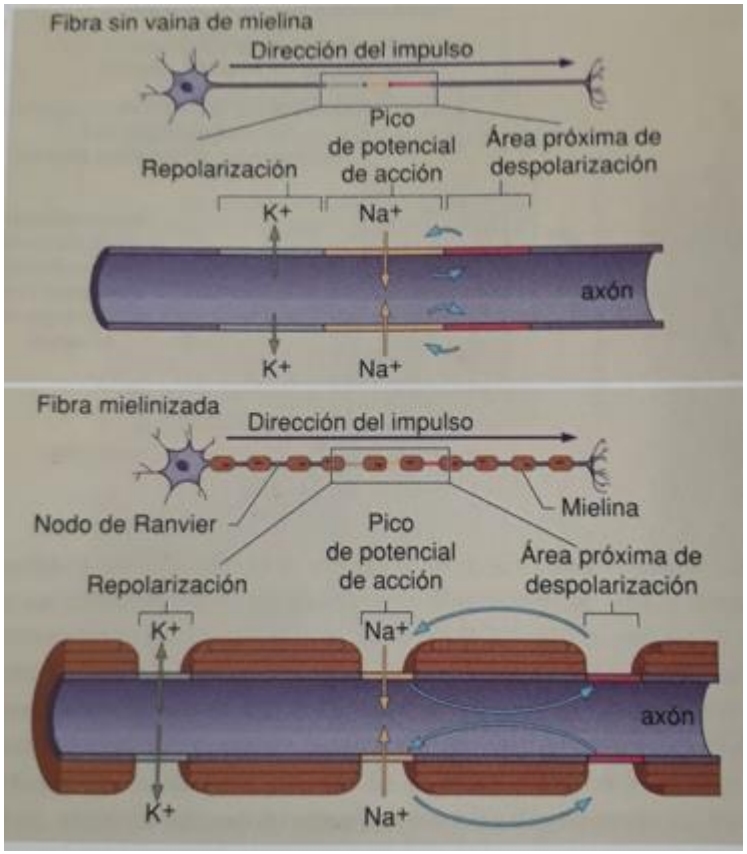
Es la simplicidad de este sistema la responsable de la mayoría de las funciones llevadas a cabo por nuestro cerebro y serían una consecuencia de la operación en paralelo de todas las neuronas del mismo.

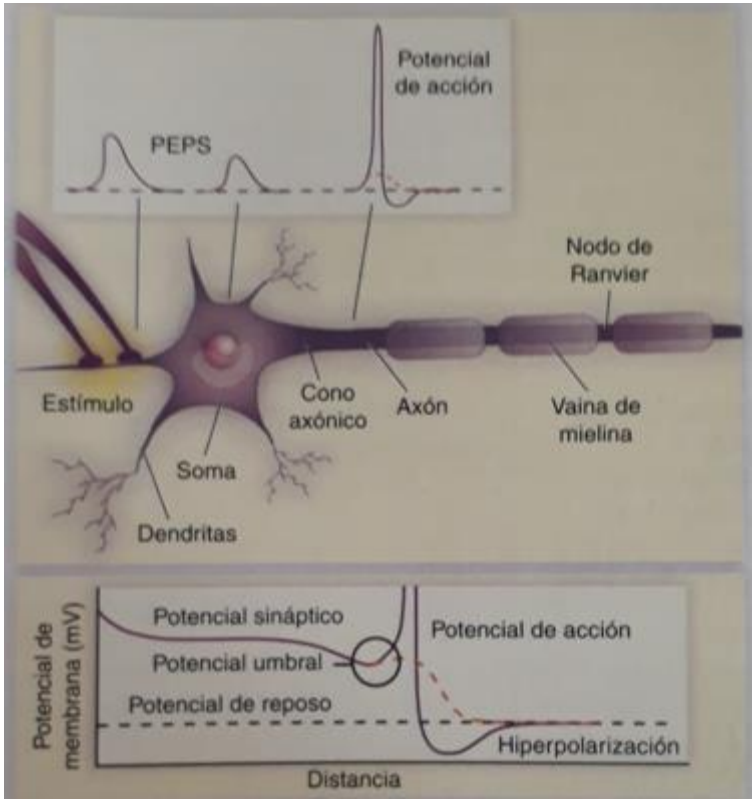
En la célula nerviosa existe una diferencia de potencial entre su interior y el exterior debida a una concentración mayor de K+ interna y una concentración mayor de Na+ externa (*polarización*). Es la concentración intracelular de K+ la principal responsable del *potencial de reposo* (-70mV aproximadamente). Para que la célula dispare un potencial de acción, es necesario que los impulsos sinápticos reduzcan este potencial de reposo a alrededor de –40mV (umbral). En ese instante por pasaje de Na+ hacia el interior y de K+ hacia el exterior mediante *canales específicos* operados por voltaje, se invierte la polaridad celular, haciendo súbitamente el interior celular positivo respecto del exterior (*despolarización*). Esta inversión de polaridad hace que el impulso nervioso se propague a través del axón, hasta sus conexiones sinápticas



Cuando el impulso llega a los terminales del axon, sus canales se abren liberando moléculas neurotransmisoras en el *espacio sináptico* (región entre la membrana presináptica y postsináptica) y el proceso continúa en la neurona siguiente.


Axón
Célula de Schwann
Núcleo de la célula de Schwann
Nodo de Ranvier
Citoplasma de la célula de Schwann
Axón
Mielina
Vaina de mielina
Núcleo
Membrana celular


Fibra sin vaina de mielina
Dirección del impulso
Pico de potencial de acción
Área próxima de despolarización
Repolarización
K+
Na+
axón
K+
Na+
Fibra mielinizada
Dirección del impulso
Nodo de Ranvier
Mielina
Pico de potencial de acción
Área próxima de despolarización
Repolarización
K+
Na+
axón
K+
Na+


Potencial de acción
PEPS
Nodo de Ranvier
Estímulo
Cono axónico
Axón
Vaina de mielina
Soma
Dendritas
Potencial de membrana (mV)
Potencial sináptico
Potencial umbral
Potencial de reposo
Potencial de acción
Hiperpolarización
Distancia

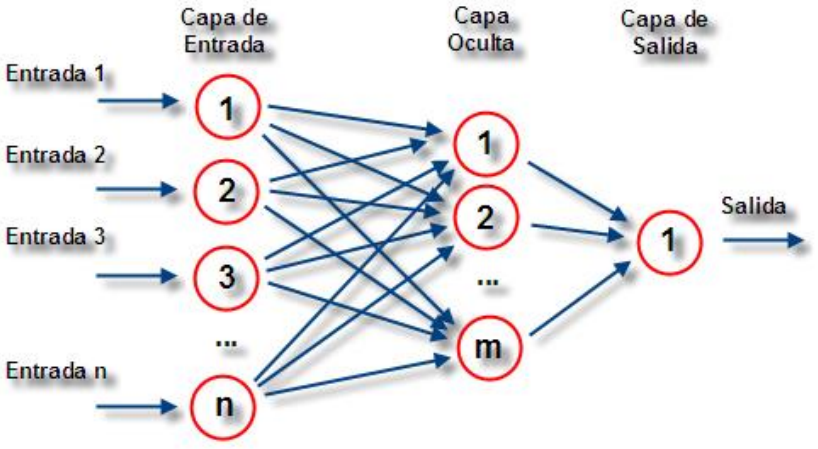
Hay distintos tipos fundamentales de neurotransmisores, aquellos que inducen una despolarización (acción *excitatoria*) y otros que promueven una polarización (acción *inhibitoria*) manteniendo la célula en su estado cercano al de reposo.
La sumatoria de **acciones excitatorias e inhibitorias** de todas las neuronas presinápticas determinarán la generación o no de un impulso por parte de la neurona postsináptica.
Después de generar un impulso, la neurona ingresa en un *periodo refractario* durante el cual no puede ser nuevamente estimulada hasta tanto no alcance su potencial de reposo, momento en el cual queda preparada para producir un nuevo impulso.

EL MODELO ARTIFICIAL

Como definición genérica podríamos adoptar la de Hecht-Niesen que especifica a una Red Neuronal como **"... un sistema de computación constituido por un gran número de elementos simples, elementos de proceso muy interconectados, los cuales procesan información por medio de su estado dinámico como respuesta a entradas externas**".
Si se acepta esquemáticamente la especificación anterior, las redes artificiales entonces, pueden imitar bastante bien varios de los aspectos que son patrimonio del cerebro ya que, pueden aprender de la experiencia, generalizar ante nuevos planteos, abstraer aspectos esenciales de información irrelevante, entre otros.

Redes neuronales artificiales

Al conjunto de las neuronas cuyas entradas provienen de una misma fuente y cuyas salidas se dirigen a un mismo destino , se lo denomina *capa* o *nivel* .

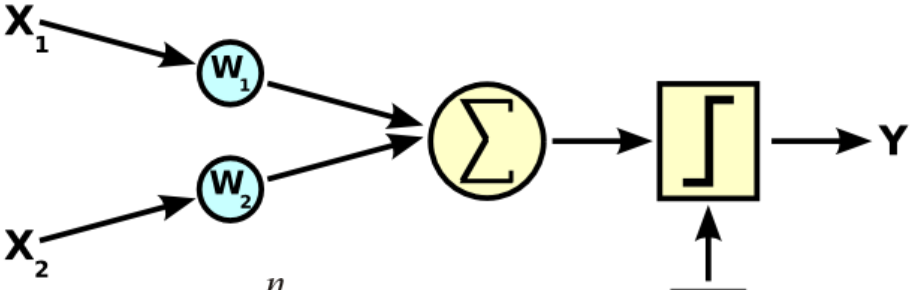


ESTADO DE ACTIVACION

Las neuronas que componen la red se pueden encontrar en uno de dos estados posibles de activación : reposo o excitación. Se le puede asignar a cada uno de ellos un valor discreto (**0** : reposo, **1** : excitado) o continuo si se le da un valor comprendido en un intervalo , usualmente entre **0** y **1** ó entre **–1** y **1**.

FUNCION DE ACTIVACION

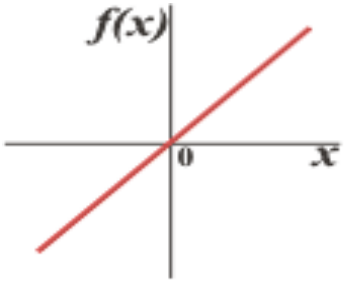
Es aquella que asociada a cada neurona transforma el estado actual de activación en una señal de salida siempre y cuando supere un nivel de umbral . Siguiendo el modelo neuronal artificial desarrollado por McCullooch y Pitts existe una función que activa o no la salida dependiendo de la suma ponderada de sus entradas, dada por la siguiente ecuación



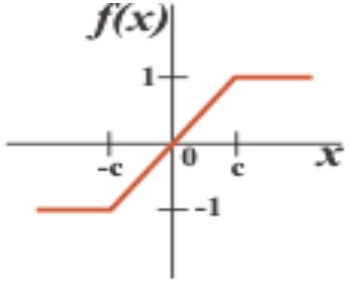
$$\sum_{i=1}^{n} x_i w_i \geq \theta$$

El modelo McCullooch y Pitts tiene algunas particularidades:
- Que al ser de una sola capa sólo resuelve funciones linealmente separables.
- Los pesos asociados a las entradas son fijos, no ajustables

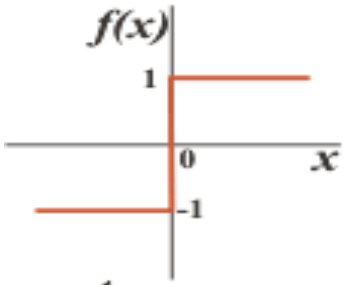
A partir del modelo McCullooch y Pitts se han derivado otras funciones de activación que permiten una salida no necesariamente de 0 ó 1. La figura 5 ilustra otras funciones de activación.



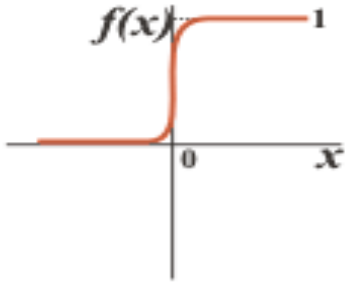
$$y = a \cdot x$$



$$y = \begin{cases} 1 \; si \; x > c \\ a \cdot x \; en \; otro \; caso \\ -1 \; si \; x < -c \end{cases}$$



$$y = \begin{cases} 1 & si \; x \geq 0 \\ -1 & si \; x < 0 \end{cases}$$



$$y = \frac{1}{1 + e^{-x/T}}$$

Funciones de Activación

En cuanto al número de capas se puede tener:

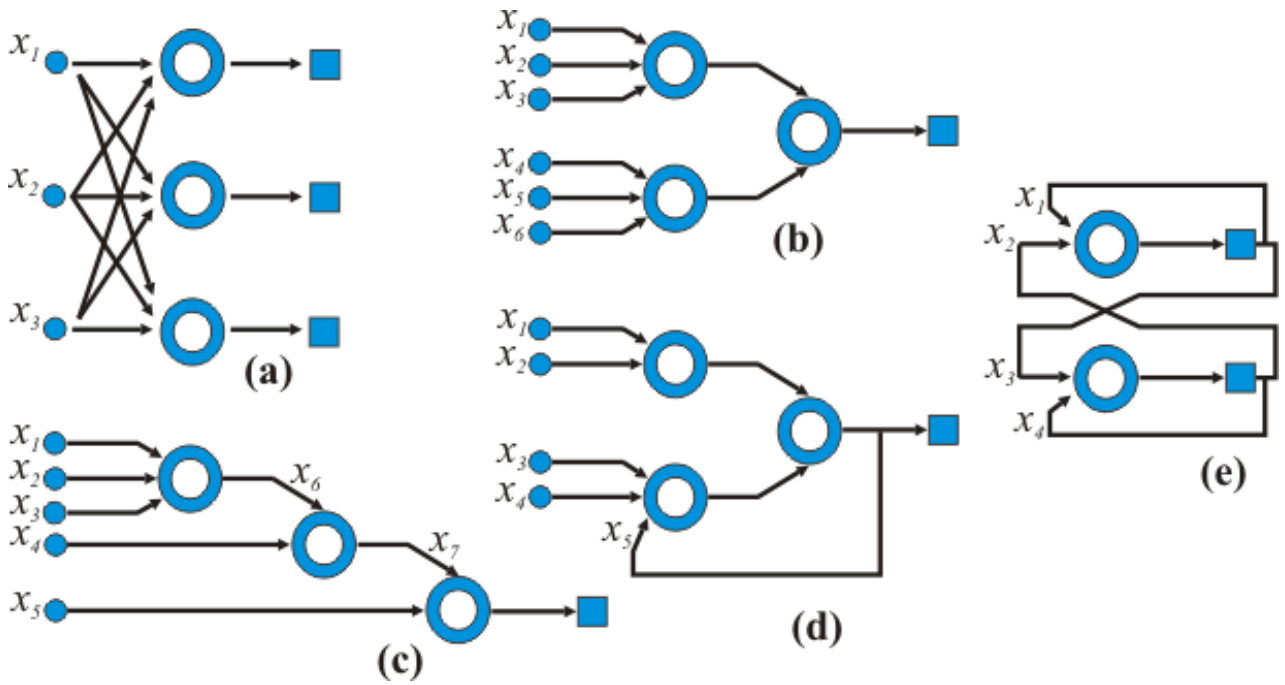
Redes de capa única: Sólo hay una neurona entre cualquier entrada y cualquier salida de la red. (figura 6a).

Redes de capas múltiples: Existe más de una neurona entre alguna entrada y alguna salida de la red (figuras 6b, 6c, 6d).

Las neuronas pueden tener conexiones del tipo:

Feedforward (conexiones hacia delante) *o* ***acíclica (****FF)*: La salida de una neurona en la *iésima* capa de la red no puede ser usada como entrada a neuronas en capas de índice menor o igual (figuras 6a, 6b, 6c)

Feedback (conexiones hacia atrás) *o cíclica (FB)*: La salida de una neurona en la *iésima* capa de la red puede ser usada como entrada a neuronas en capas de índice menor o igual (figura 6d, 6e)



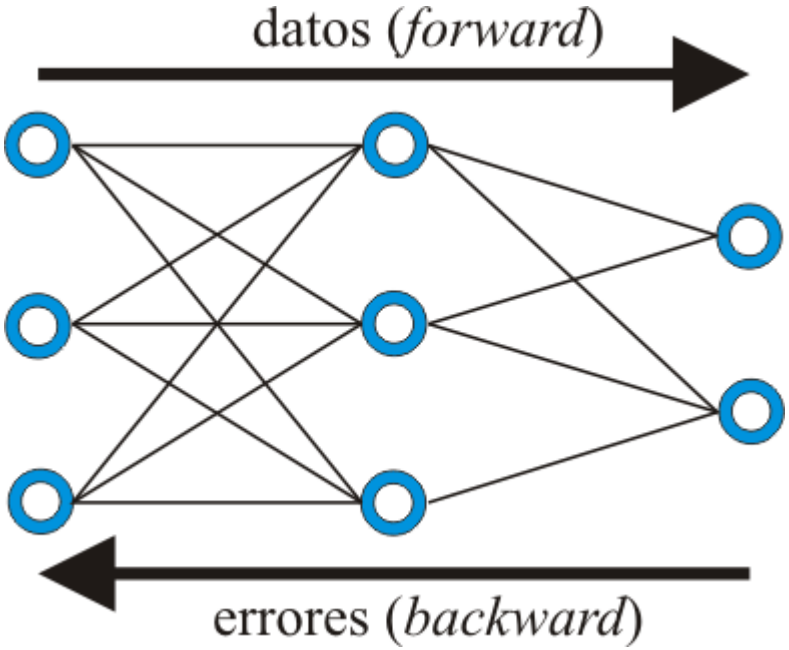
PROCESO DE APRENDIZAJE DE UNA RNA

Los procesos de aprendizaje pueden ser rastreados hasta 1949 cuando Hebb introduce la primera regla con potencial para demostrar los posibles mecanismos que una RNA puede utilizar para aprender.

Es a partir de 1982 que se dan a conocer nuevos paradigmas neuronales y algoritmos de aprendizaje que pueden ser divididos en dos categorias básicas:

Aprendizaje supervisado*:* Requiere de un “maestro” u otra fuente de información que especifique directamente a la RNA la respuesta correcta para cada estímulo definido a la entrada. Por ejemplo el “maestro” puede ser un humano o un proceso real generando casos típicos de salida requerida, en correspondencia a un conjunto definido de entrada. Uno de los más populares es el denominado “*Backpropagation of error*” (BP) - (propagación hacia atrás del error) dado que usa patrones que especifican directamente la salida deseada.

Aprendizaje no supervisado*:* Requiere que la RNA determine alguna estructura a partir de la información que se le presenta; en este caso no hay un “maestro” dictando las respuestas correctas y la red debe determinar por sí misma cómo generar las salidas correctas a partir generalmente, de una redundante presentación de patrones de entrada.



RNA Tipo *Feedforward* con aprendizaje *Backpropagation of error*

Es la más comunmente usada . El algoritmo BP es una regla de aprendizaje que se puede aplicar en modelos de redes con más de dos capas de neuronas. Una característica importante de este algoritmo es la representación interna del conocimiento que es capaz de organizar en la capa intermedia u oculta para obtener la correspondencia entre la entrada y la salida de la red

Recordemos que el modelo McCullooch y Pitts de una sola capa de neuronas sólo resuelve problemas lineales, en cambio, al incluir una o más capas ocultas, se puede implementar cualquier función continua.

Primera fase: Se aplica un patrón a la capa de entrada, como estímulo, que se propagará a través de la red hasta generar una salida. Luego se compara la misma con las salidas que se desean obtener (objetivos) y se calcula un valor de error, para cada neurona de salida.

Segunda fase: Los errores determinados se transmiten hacia atrás partiendo de la capa de salida hacia todas las neuronas de la capa intermedia, recibiendo el porcentaje de error aproximado a la participación de la neurona intermedia, en la salida original. Este proceso se repite, capa por capa, hasta que todas las neuronas de la red haya recibido un error que describe su aporte relativo al error total. En base al valor del error recibido, se reajustan los pesos de conexión de cada neurona, de forma tal que la próxima vez que se le presente el mismo patrón, la salida esté más cercana a la deseada, es decir que, el error disminuye.

La importancia de la red *Backpropagation of error* radica en su capacidad de autoadaptar los pesos de las neuronas de las capas ocultas para aprender la relación que existe entre un conjunto de patrones dados como ejemplo y sus correspondientes salidas, esto permite, después del entrenamiento, que nuevos vectores de entrada incompletos o con ruido pero parecidos a los presentados durante el entrenamiento, den una salida similar a las propuestas durante el aprendizaje.

Una característica importante que se exije de los sistemas de aprendizaje, es la capacidad de *generalización* entendiendo como tal, a la facilidad de la RNA de brindar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento, sirviendo de proceso de *validación* del comportamiento de la misma.

Las RNAtambien permiten estimar nuevas salidas para datos nuevos.

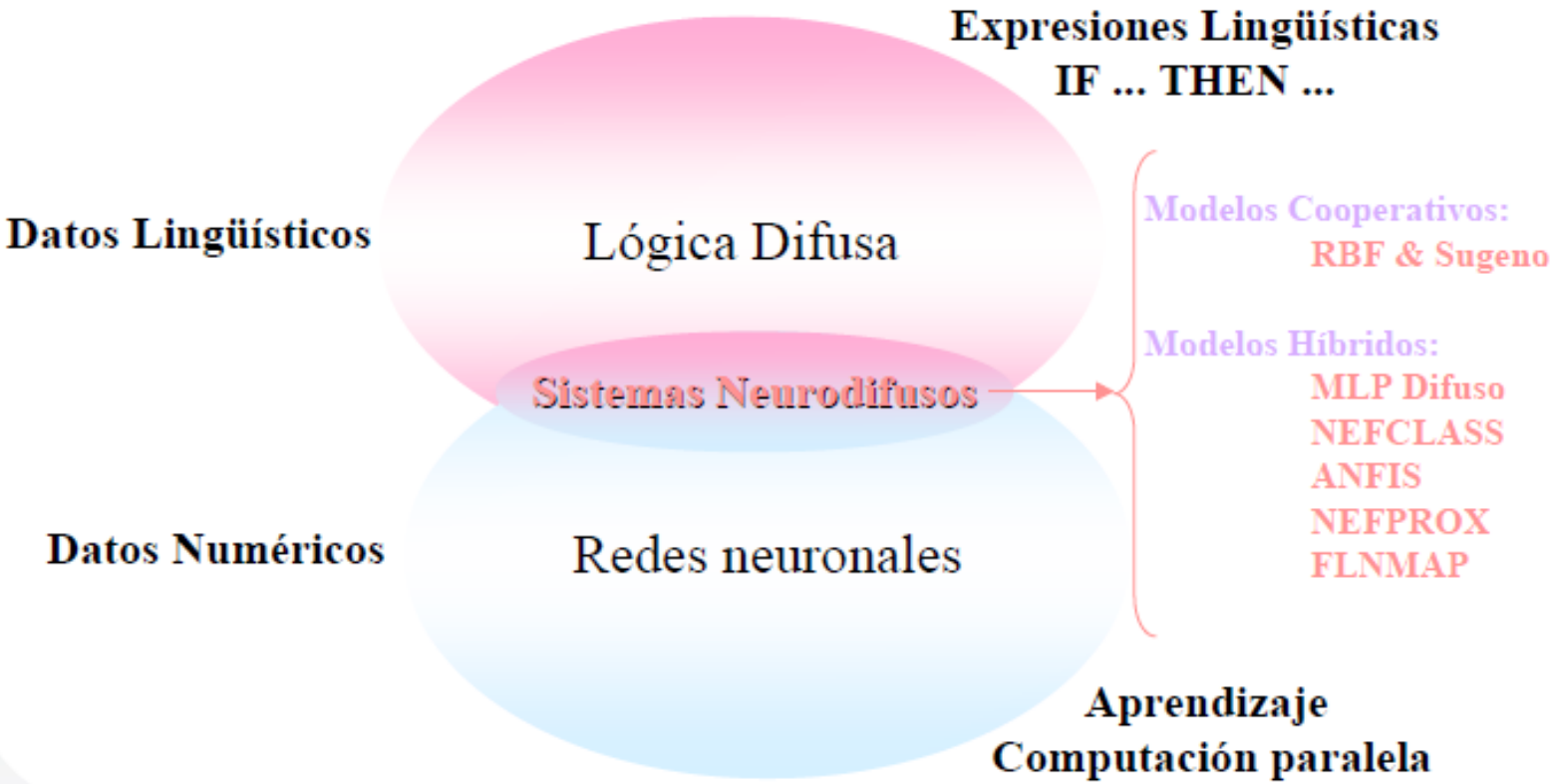
• La **lógica difusa** está basada en la forma en que el cerebro maneja **información inexacta**.

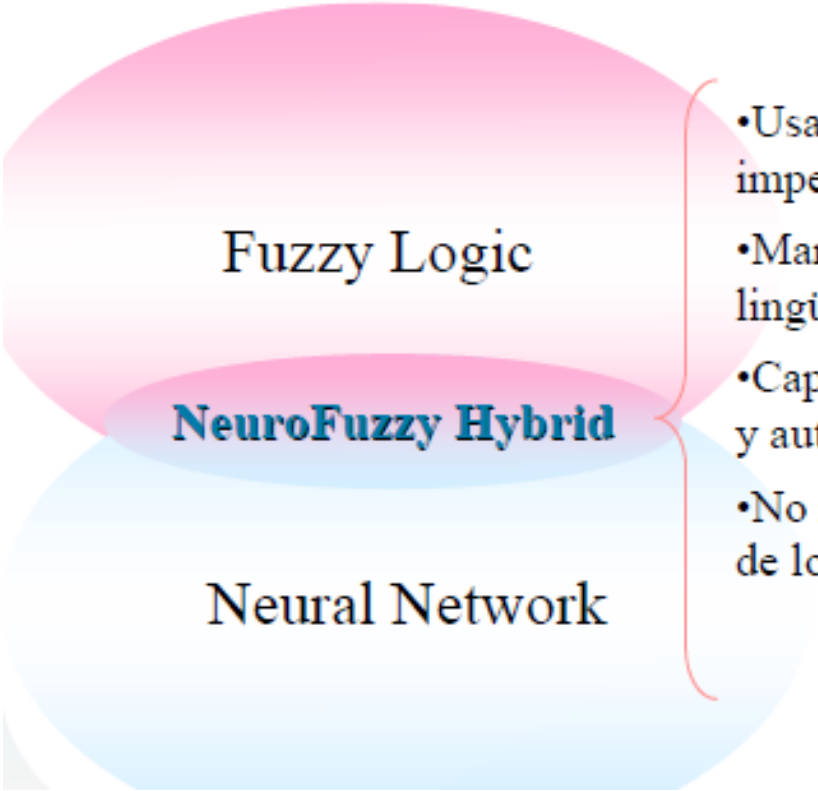
• Las **redes neuronales** están modeladas en la arquitectura física del cerebro.

• Las **redes neuronales artificiales** (ANNs/RNA) son un paradigma para la detección de patrones basado en la interconexión de unidades denominadas **neuronas**.

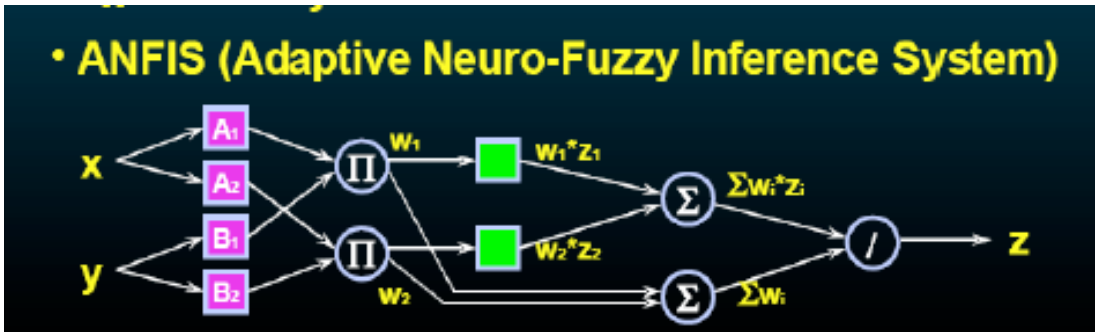
• La neurona artificial es un modelo basado en los complejos sistemas nerviosos de los animales y seres humanos con su gran cantidad de interconexiones y paralelismo

Tipos


Expresiones Lingüísticas
IF ... THEN ...
Datos Lingüísticos
Lógica Difusa
Modelos Cooperativos:
RBF & Sugeno
Modelos Híbridos:
MLP Difuso
NEFCLASS
ANFIS
NEFPROX
FLNMAP
Sistemas Neurodifusos
Datos Numéricos
Redes neuronales
Aprendizaje
Computación paralela



- Usa información imprecisa, parcial, vaga o imperfecta
- Maneja cualquier tipo de información (numérica, lingüística, lógica, etc.)
- Capacidades de Auto-aprendizaje, auto-organización y auto-ajuste
- No necesita un conocimiento a prior de las relaciones de los datos



ANFIS

• Adaptive Neural Fuzzy Inference System • Red neuronal adaptativa con parámetros difusos o sistema difuso con funcionamiento distribuido
• Aplicaciones:
– Procesamiento y filtrado de señales
– Control adaptativo
– Clasificación de datos.

• Red Feed Forward
• Nodos adaptivos
• Aprendizaje similar a redes neuronales
– Ajustar los parámetros de los nodos adaptativos
• Conocimiento previo: Topología de la red (reglas difusas)

• **La forma de las funciones de pertenencia dependen de los parámetros**, y cambiando estos parámetros cambiará la forma de la función de pertenencia.

• **La función de pertenencia puede ser ajustada automáticamente usando ANFIS**.

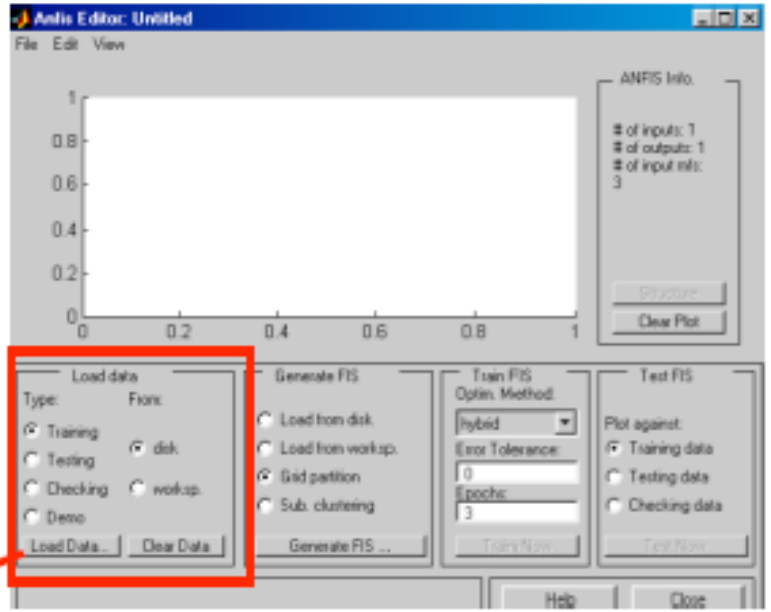
• Esta técnica provee un método para el procedimiento de modelado difuso que **permite aprender información sobre el conjunto de datos**, en orden a computar los parámetros de las funciones de pertenencia que mejor asociación entre los datos de entrada/salida obtenga para el sistema de inferencia difuso.

• **Uno de los problemas** con la validación del modelo construido usando técnicas adaptativas es la selección del **conjunto de datos** que sea **representativo** para los datos.

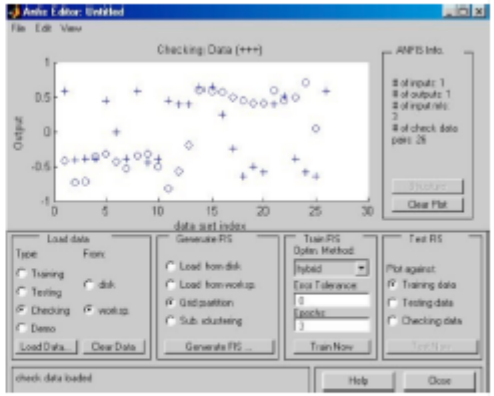
• La idea cuando usamos una **verificación de datos** es **reducir** el efecto de **overfitting (sobreajuste)** del modelo.

- En el área de Load data y seleccione las opciones ***Training y worksp***.

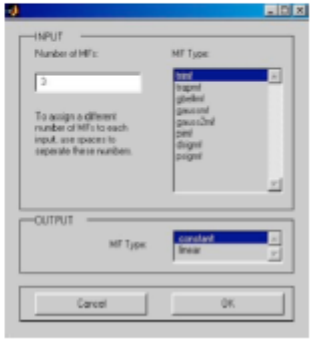
- A continuación pulse el botón ***Load Data...*** Para introducir la entrada de datos para el entrenamiento



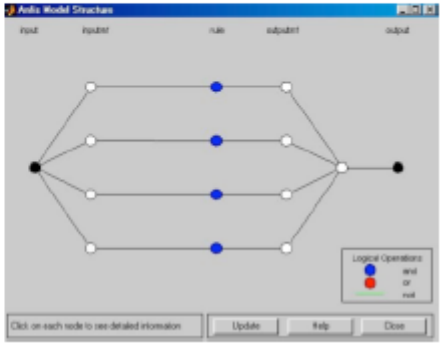
- Escriba el nombre *fuzex1trndata* y presione OK para cargar los datos.
- En el área ***Load data*** y seleccione las opciones ***Cheking y worksp.***
- A continuación presione el botón ***Load Data...*** para introducir la entrada de datos para verificar el entrenamiento.
- Escriba *fuzex1chkdata* y presione OK para cargar los datos Estos datos servirán para ajustar las Funciones de pertenencia.



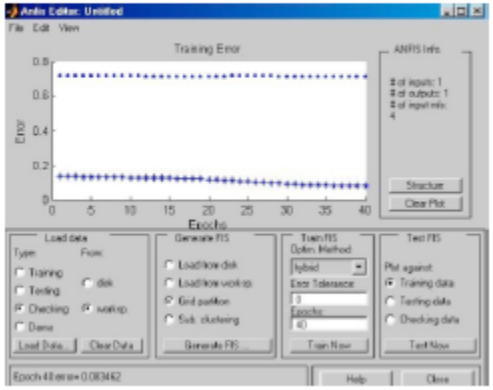
- A continuación se puede generar el sistema difuso o dejar que el toolbox cree uno.
- Para cargar un sistema ya hecho, basta con que se seleccione ***Generate FIS*** y ***cargar*** un sistema ya hecho ya sea desde ***disco o desde el espacio de trabajo***.
- Pero nosotros vamos a crear el sistema por lo que escogeremos ***Generate FIS, Grid partition*** y pulsar el botón ***Generate FIS...*** Sólo podemos modificar el número de funciones de pertenencia, el tipo de función y el tipo de salida, la cual sólo puede ser constante o lineal debido a que solo se puede usar un sistema del tipo Sugeno.



- A continuación seleccionamos.
  (INPUT) Number of MF: 4 y Type of MF: Gbellmf
  (OUTPUT) MF Type: Linear
- Una vez generado el sistema difuso podemos ver la estructura, seleccionando el botón ***structure***.



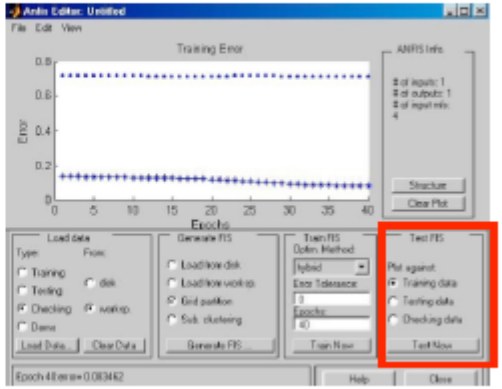
- Ahora solo queda entrenar el sistema. Esto se hace seleccionando método de optimización de la red, el error, y el número de iteraciones (***epochs***) que se necesitan.
  Optim Meted: Hibrid
  Error Tolerance: 0
  Epochs: 40
- A continuación pulse el botón ***Train now***, el sistema mostrará una gráfica con el error de los datos de entrenamiento (training) y el error de los datos de verificación (checking). La gráfica de la parte superior indica el error de entrenamiento y la inferior el error de verificación.



- Ya está creado el sistema difuso a partir de datos de entrada , para guardarlo, solo es necesario ir a la opción ***File, export, to disk*** , escribir el nombre del sistema y luego pulsar el botón guardar.
- Si queremos ver en la interfaz gráfica de usuario el sistema almacenado, basta con poner en la línea de comandos de Matlab ***>>fuzzy nombre***

- Esto nos permite editar las funciones de pertenencia para adaptarlas a un caso particular.

- Si queremos verificar algunas gráficas como por ejemplo los datos de entrenamiento y de verificación, basta seleccionar dichas opciones en



- y luego presionar el botón ***Test now***