ESCUELA SUPERIOR POLITECNICA DEL LITORAL

Facultad de Ingeniería Eléctrica y Computación

"APLICACIÓN DEL MODELO ANFIS A LA SINTETIZACION DE NOTAS MUSICALES Y SEÑALES DE VOZ"

INFORME DE PROYECTO DE GRADUACION

Previa a la obtención del título de:

INGENIERO EN COMPUTACION
ESPECIALIZACION SISTEMAS TECNOLÓGICOS

Presentada por:

CARLOS STALIN ALVARADO SANCHEZ

Guayaquil – Ecuador

AÑO

2010

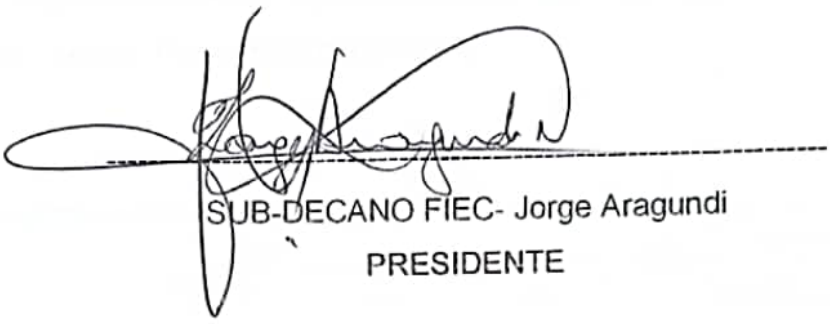
Agradecimientos

A todos mis ilustres profesores de la universidad que han aportado en mí conocimientos, los cuales me han dado la oportunidad de desempeñarme en mi profesión.

Dedicatoria

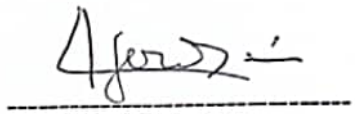
A las personas que nacen bajo condiciones adversas, que luchan y se esfuerzan por ser alguien en esta vida; el poder de superación, fe y perseverancia pueden lograr metas inalcanzables.

TRIBUNAL DE SUSTENTACION



SUB-DECANO FIEC- Jorge Aragundi

PRESIDENTE



Carlos Jordán Villamar

DIRECTOR

Xavier Antonio Ochoa Chehab

VOCAL PRINCIPAL

DECLARACION EXPRESA

"La responsabilidad del contenido de este trabajo de graduación, me corresponde exclusivamente; y el patrimonio intelectual de la misma a la Escuela Superior Politécnica del Litoral".

(Reglamento de Graduación de la ESPOL)

Carlos Stalin Alvarado Sánchez

RESUMEN

En este trabajo se presenta la construcción de una herramienta para la sintetización de notas musicales y de voz utilizando un sistema de inferencia borrosa con base en una red adaptable (ANFIS).

Con el propósito de entender mejor el método utilizado en la construcción de este sintetizador, se presentará a continuación una explicación un tanto somera de lo que son los sistemas difusos, las redes neuronales, y, más detallada en el caso de las redes ANFIS.

La herramienta de software que se construyó en este proyecto, a la que hemos llamado CACIQUE, por ser esta un ave de hermoso canto que habita en nuestra región, se implementó aplicando la metodología de desarrollo conocida como MSF (Microsoft Solutions Framework), que consiste en un modelo cíclico donde las principales fases de cada ciclo son: envisionamiento, planificación, desarrollo, estabilización e implementación.

Se utilizará esta herramienta para sintetizar notas musicales y señales de voz, finalmente se comparará los resultados obtenidos mediante este modelo con aquellos obtenidos por el método de Red Neuronal usado por Axel Röbel para re sintetizar señales de voz [1].

Por otro lado la herramienta se la utilizará para sintetizar algunas notas musicales y señales de voz. Se compararán entonces las reproducciones sintetizadas con las señales originales para determinar la bondad del sintetizador.

INDICE GENERAL

Agradecimientos 2
Dedicatoria 3
TRIBUNAL DE GRADUACION 4
DECLARACION EXPRESA 4
RESUMEN 6
INDICE GENERAL 7
ABREVIATURAS 9
INTRODUCCION 10
CAPITULO 1 1
El problema de la sintetización de música y de voz 1
1.1 El estado de arte en la sintetización de música y voz 2
1.2 Especificación de los Requerimientos de CACIQUE 10
CAPITULO 2 13
La Red ANFIS: Sistema de Inferencia Borrosa con Base en Redes Adaptivas 13
2.1. Introducción a la Redes Neuronales 14
2.2 Introducción a los Sistemas borrosos 20
2.3 Sistemas Neuro Difuso 25
2.4 Descripción de la RED ANFIS 37
CAPITULO 3 46
Diseño de CACIQUE 46
3.2 Diseño Arquitectónico de CACIQUE 49
3.3 Vista Panorámica del Diseño Detallado de CACIQUE 54
CAPITULO 4 55
Implementación de CACIQUE 55
4.1 El entorno de desarrollo de CACIQUE 56
4.2 Obtención de las señales de música y de voz a sintetizar 56
4.3 Problemas encontrados en la implementación de CACIQUE 57
4.4 Vista panorámica de la codificación de las clases de CACIQUE 58
CAPITULO 5 59
Pruebas y Resultados 59
5.1 El entorno de pruebas de CACIQUE 61
5.2 Pruebas de CACIQUE con notas musicales 62
5.3 Pruebas de CACIQUE con señales de voz 71

5.4 Comparación de CACIQUE con otros sintetizadores 80
5.5 Calculo de la métrica para medir la eficiencia del sintetizador 82
Conclusiones 84
Recomendaciones 86
Anexos 88
Anexo 1: Requerimientos del Sistema 88
Anexo 2: Especificación de Casos de Usos 91
Anexo 3: Detalle de Clases 96
Anexo 4: Diseño Detallado 110
Anexo 5: Vista Preliminar de clases codificadas en java 127
Anexo 6: Manual del Usuario 137
Bibliografía 145

ABREVIATURAS

ANFIS	Adaptive Network Fuzzy Inference System
MSF	Microsoft Solutions Framework
FL	Fuzzy Logic
ANN	Artificial Neural Networks
RNA	Redes Neuronales Artificiales
TTS	Text To Speech
UML	Unified Model Language
ECM	Error Cuadrático Medio

INTRODUCCION

Siendo visionarios vemos que las interfaces con el usuario en lenguaje natural serán las interfaces del futuro, porque permitirán al usuario comunicarse con el computador de una manera más “natural”, como lo hacemos normalmente con nuestros congéneres, utilizando la facultad del habla. Al diseñar estas interfaces, dos funciones esenciales deben implementarse: el reconocimiento y la sintetización de la voz. En este trabajo queremos aplicar la red ANFIS (Sistemas de Inferencias Borrosas Basados en Redes Adaptables) al problema de sintetizar señales de voz y notas musicales, debido a que combina técnicas de redes neuronales y sistemas difusos; me hace prever que se podría obtener mejores resultados que los otros algoritmos de redes neuronales, como en caso de Axel Röbel en su artículo propuesto en [1].

Por otro lado existen algunos problemas, para los que la inteligencia humana es mucho más rápida y eficiente que el procesamiento de la mejor computadora actual; por ejemplo en un juego de ajedrez a pesar que hay reglas como jugarlo, cada partida es única y por ende hay una infinidad de casos que hace imposible indicarle a la computadora las instrucciones a seguir para cubrir cada posibilidad. Otro ejemplo es manejar un automóvil son infinitas las situaciones que imposibilitan programar un sistema que siga una serie de instrucciones predefinidas. Tratando de atacar estas deficiencias en la forma de resolver problemas tradicionales surgieron áreas como Lógica Difusa (Fuzzy Logic, FL), Redes Neuronales Artificiales (Artificial Neural Networks, ANN) y otras herramientas que se suelen agrupar en el concepto de Inteligencia Computacional.

A medida que esta disciplina fue creciendo, se fueron identificando mejor las propiedades de cada herramienta y su área de aplicación. Posteriormente se desarrollaron arquitecturas combinadas o híbridas usando Lógica Difusa y Redes Neuronales Artificiales para ampliar la clase de problemas que cada una puede tratar por sí misma y mejorar la solución global encontrada.

La fortaleza de los Sistemas Neuro-Difuso reside en dos principios contradictorios del modelado difuso: interpretabilidad frente a precisión. En la práctica, una de las dos propiedades prevalece sobre la otra según Takagi en [2].

Los Sistemas Neuro-difuso combinan las ventajas de las redes neuronales con las de los Sistemas Difusos, como ya se comentó anteriormente. Éstos últimos proporcionan una explicación y comprensión al conocimiento explícito, mientras que las Redes Neuronales Artificiales tratan con el conocimiento implícito que se puede adquirir por medio del aprendizaje.

El aprendizaje de Redes Neuronales proporciona una buena forma de ajustar el conocimiento experto y generar automáticamente reglas difusas y funciones de pertenencia o idoneidad para ajustarse a ciertas especificaciones y reducir el tiempo de diseño y los costos. Por otro lado, la Lógica Difusa aumenta la capacidad de generalización de las Redes Neuronales proporcionando salidas más confiables cuando se necesita extrapolar más allá de los datos de entrenamiento según Jang en [3].

CAPITULO 1

El problema de la sintetización de música y de voz

En la disciplina de Ingeniería de Software para poder desarrollar un producto y posteriormente implementarlo, de debe conocer previamente el ¿qué?, ¿por qué?, y ¿para quién? Se va a desarrollar un producto o aplicación de software.

En este caso el ¿para quién? Vendría a ser la comunidad de investigadores que buscan nuevas tecnologías para poder lograr mejoras a la calidad de los sonidos reproducidos artificialmente.

Ante el problema que posteriormente puntualizamos en este capítulo, obtendremos una lista de requerimientos de los cuales CACIQUE fue desarrollado.

1.1 El estado de arte en la sintetización de música y voz

En la actualidad existen instrumentos electrónicos diseñados para producir sonido generado artificialmente, usando técnicas como síntesis aditiva, substractiva, de modulación de frecuencia, de modelado físico o modulación de fase, para crear sonidos, según la fuente web propuesta en [4].

Los más conocidos son los órganos musicales que pueden producir sonidos de cualquier instrumento musical; la gran mayoría utiliza el estándar MIDI que se trata de un protocolo industrial estándar que permite a las computadoras, sintetizadores, secuenciadores, controladores y otros dispositivos musicales electrónicos comunicarse y compartir información para la reproducción de sonidos.

Por otro lado están los sintetizadores por software, también conocidos como softsynth (instrumento virtual) son programas de computadora que se utilizan para la creación de audio digital. El hecho que las computadoras puedan crear música o sonidos no es nuevo, sin embargo, los avances en la capacidad de procesamiento permitieron que los sintetizadores por software puedan realizar las mismas tareas que el hardware creado para los mismos fines. Una ventaja de los sintetizadores por software es que pueden interactuar con otras aplicaciones, por ejemplo, secuenciadores, según la fuente web propuesta en [5].

Existen dos tipos de sintetizadores por software: Los emuladores, los cuales crean sonidos a base de algoritmos; y los basados en muestras, estos como su nombre lo indican utilizan muestras de audio para generar sonido. Algunas de estas muestras están específicamente diseñadas para imitar el sonido de instrumentos reales como pianos y teclados, según la fuente web propuesta en [5].

En lo que compete a la Sintetización de señales de voz, tenemos que expresar que la voz sintética es una voz artificial (no pregrabada), generada mediante un proceso de sintetización del habla.

La síntesis de habla es la producción artificial de habla humana. Un sistema usado con este propósito recibe el nombre de sintetizador de habla y puede llevarse a cabo en software o en hardware. La síntesis de voz se llama a menudo en inglés text-to-speech (TTS), en referencia a su capacidad de convertir texto en habla. Sin embargo, hay sistemas que en lugar de producir voz a partir de texto lo hacen a partir de representación lingüística simbólica en habla.

La calidad de una voz sintética vendrá dada, según la fuente web propuesta en [5] por:

Su inteligibilidad: ¿con qué facilidad/dificultad es entendida?

Su naturalidad: ¿en qué medida se asemeja a la voz real de un humano?

Panorama de la tecnología de síntesis de voz

Un sistema texto a voz se compone de dos partes: un front-end y un back-end. A grandes rasgos, el front-end toma como entrada texto y produce una representación lingüística fonética. El back-end toma como entrada la representación lingüística simbólica y produce una forma de onda sintetizada, según la fuente web propuesta en [5].

El front-end desempeña dos tareas principales. Primero, toma el texto y convierte partes problemáticas como números y abreviaturas en palabras equivalentes. Este proceso se llama a menudo normalización de texto o pre procesado. Entonces asigna una transcripción fonética a cada palabra, y divide y marca el texto en varias unidades prosódicas, como frases y oraciones.

El proceso de asignar transcripciones fonéticas a las palabras recibe el nombre de conversión texto a fonema (TTP en inglés) o grafema a fonema (GTP en inglés). La combinación de transcripciones fonéticas e información prosódica constituye la representación lingüística fonética, según la fuente web propuesta en [5].

La otra parte, el back-end, toma la representación lingüística simbólica y la convierte en sonido. El back-end se llama a menudo sintetizador.

Tecnologías de Síntesis

En la actualidad estas tecnologías han alcanzado un importante desarrollo, tanto en el ámbito de la síntesis como en el del reconocimiento. Este progreso está muy relacionado con el conocimiento de la fisiología de la voz y de la audición, a los que la tecnología trata de imitar.

Hay dos tecnologías usadas para generar habla sintética, según la fuente web propuesta en [5]:

- Síntesis Concatenativa
- Síntesis de Formantes

La síntesis Concatenativa se basa en la concatenación de segmentos de voz grabados. Por lo general estas producen los resultados más naturales, hay tres tipos de síntesis concatenativa: síntesis por selección de unidades, síntesis de difonos, síntesis específica para un dominio.

La síntesis por selección de unidades utiliza una base de datos de voz grabada, en el cual el habla se segmenta en fonemas, silabas, palabras, frases y oraciones; en tiempo de ejecución, el objetivo deseado se crea determinando la mejor cadena de candidatos de la base de datos. Este proceso se logra típicamente usando un árbol de decisión especialmente ponderado.

La síntesis de difonos utiliza una base de datos mínima conteniendo todos los difonos que pueden aparecer en un lenguaje dado; en tiempo de ejecución la prosodia de una oración se sobre impone a estas unidades mínimas mediante procesamiento digital de señales como codificación lineal predictiva. La calidad del habla es peor que la obtenida mediante selección de unidades pero más natural que las obtenidas mediante sintetización de formantes.

La síntesis específica para un dominio concatena palabras y frases grabadas para crear salidas completas. Se usa en aplicaciones donde la

variedad de textos que el sistema puede producir está limitada a un dominio particular, como anuncios de salidas de trenes o información meteorológica.

Esta tecnología es muy sencilla de implementar, y se ha utilizado comercialmente durante largo tiempo: es la tecnología usada por aparatos como relojes y calculadoras parlantes. La naturalidad de estos sistemas puede ser muy grande, porque la variedad de oraciones está limitada y corresponde a la entonación y a la prosodia de las grabaciones originales. Sin embargo, al estar limitados a ciertas frases y palabras de la base de datos, no son de propósito general y sólo pueden sintetizar la combinación de palabras y frases para los que fueron diseñados.

La síntesis de formantes no usa muestras de habla humana al tiempo de ejecución. En su lugar, la salida se crea mediante un modelo acústico. Parámetros como la frecuencia fundamental y los niveles de ruido se varían en el tiempo para crear una forma de onda o habla artificial. Este método se conoce también como ***síntesis basada en reglas.***

Muchos sistemas basados en síntesis de formantes generan habla robótica y de apariencia artificial, y la salida nunca se podría confundir con la voz humana. Sin embargo, la naturalidad máxima no es siempre la meta de un sintetizador de voz, y estos sistemas tienen algunas ventajas sobre los sistemas concatenativos.

La síntesis de formantes puede ser muy inteligible, incluso a altas velocidades, evitando los defectos acústicos que pueden aparecer con frecuencia en los sistemas concatenativos. La síntesis de voz de alta velocidad

es a menudo usada por los discapacitados visuales para utilizar computadores con fluidez. Por otra parte, los sintetizadores de formantes son a menudo programas más pequeños que los sistemas concatenativos porque no necesitan una base de datos de muestras de voz grabada. De esta forma, pueden usarse en sistemas empotrados, donde la memoria y la capacidad de procesamiento son a menudo exiguas. Por último, dado que los sistemas basados en formantes tienen un control total sobre todos los aspectos del habla producida, pueden incorporar una amplia variedad de tipos de entonaciones, que no sólo comprendan preguntas y enunciaciones.

Problemas de la voz sintética

- Rechazo por parte de los usuarios que no perdonan su falta de naturalidad y su timbre robótico.
- Los CTV (Convertidores de Texto-Voz) producen voz, generalmente, voz de hombre. Hay varias razones que pueden explicar este hecho:
  - Una explicación sociológica obvia es que, hasta hace relativamente poco, las personas que trabajaban en los laboratorios eran hombres y éstos empleaban su propia voz durante los experimentos.
  - La voz masculina ofrece mejor calidad sonora que la femenina. Esto se debe a que la frecuencia fundamental (primer armónico) de la mujer es bastante más alta que la de hombre.
  - La forma de onda en la voz de mujer tiene un componente de oscilación no periódica, que viene dado por una mayor frecuencia en la aspiración, que resulta más notable que la del hombre. Este

componente de la excitación global es difícil de modelar adecuadamente.

En los últimos tiempos han aparecido sintetizadores que utilizan voz de mujer de calidad aceptable, sin embargo, siguen sin alcanzar la calidad ofrecida por un sintetizador de similares características que emplee voz masculina, según la fuente web propuesta en [5].

El Sintetizador CACIQUE

CACIQUE es un sintetizador tanto musical como de voz, que puede ser clasificado como un emulador en el área musical, y como un sintetizador formante en el área del habla debido a que es un sintetizador por software que utiliza un algoritmo para poder reproducir sonido.

Debido a la naturalidad como la inteligibilidad de CACIQUE podrá ser usado como base para construir sintetizadores más potentes en el futuro.

A continuación la figura 1 detallará brevemente como fue desarrollado CACIQUE a nivel superior de la arquitectura del producto.

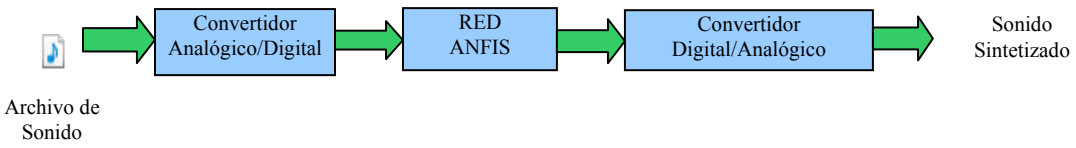


Figura 1: Arquitectura general de CACIQUE

La figura 1 muestra cómo trabaja CACIQUE; lee un archivo de sonido pasa por un convertidor analógico digital para digitalizar la señal, luego pasa a la etapa de aprendizaje utilizando la Red ANFIS, finalmente pasa al convertidor digital analógico para posteriormente reproducir el sonido sintetizado.

Este mismo producto puede ser utilizado como base para construir otros sintetizadores, un ejemplo de ello es que CACIQUE podría llegar a ser el primer sintetizador formante concatenativo, esto es además de utilizar la tecnología formante, podría utilizar la tecnología concatenativa, para esto se deberá implementar una base de conocimiento que guarde las funciones de las ondas sonoras; esta base servirá para concatenar palabras o frases de voz o para poder obtener notas musicales; en la figura 2 esta detallado lo expresado; cabe puntualizar que CACIQUE al utilizar la tecnología concatenativa se diferenciará en que la base no es de datos sino de conocimiento porque graba funciones y no archivos acústicos.

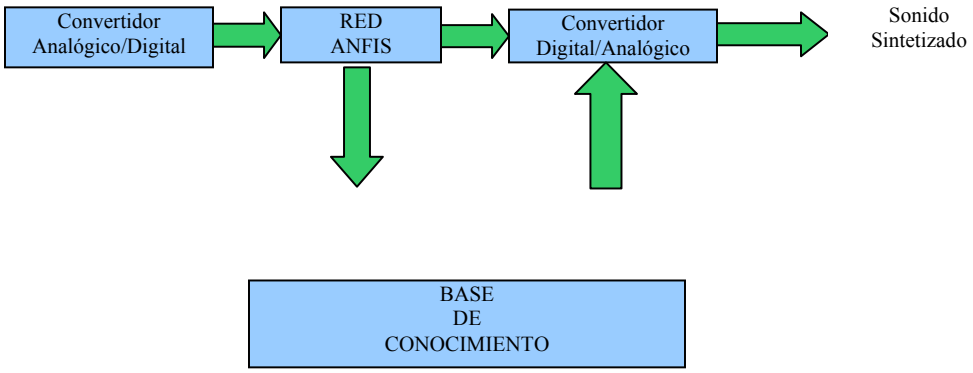


Figura 2: Arquitectura de CACIQUE como un sintetizador formante concatenativo

1.2 Especificación de los Requerimientos de CACIQUE

Los requerimientos para un sistema son la descripción de los servicios proporcionados por el sistema y sus restricciones operativas. Estos requerimientos reflejan las necesidades de los clientes de un sistema que ayude a resolver algún problema como el control de un dispositivo. Hacer un pedido o encontrar información. El proceso de descubrir, analizar, documentar y verificar estos servicios y restricciones se denomina ingeniería de requerimientos.

Requerimientos del Usuario

Los requerimientos del usuario son declaraciones, en lenguaje natural y en diagramas, de los servicios que se espera que el sistema proporcione y de las restricciones bajo las cuales debe funcionar.

Como usuario se tomó al desarrollador, debido a su experiencia adquirida en las diferentes empresas que ha trabajado y con los clientes que ha tratado, para el desarrollo de software específico en diferentes dominios; esto le da un "plus", para obtener requerimientos necesarios para CACIQUE, así como una buena interfaz amigable y con un nivel de usabilidad alto.

Requerimientos Funcionales

- El proceso se inicia seleccionando el archivo de audio de formato WAV en el cual tiene la señal de voz o de una nota musical en particular

- Luego de esto el sistema tendrá que digitalizar la señal y escribir los datos de la amplitud en un archivo Excel
- El sistema tendrá que dar la opción para escoger y seleccionar el tipo de función de membrecía a usar y también el número de funciones para posteriormente sintetizar los datos a través de la red ANFIS
- Una vez concluido el proceso de aprendizaje, el software escribirá en el archivo Excel los datos sintetizados
- Una vez concluido el proceso de aprendizaje, el software mostrará la señal original y sintetizada con diferentes colores, además de visualizar cual ha sido el error cuadrático medio (ECM)
- El software debe reproducir sonido original y el sintetizado.

Casos de Usos

- Abrir Archivo
- Buscar Archivo
- Digitalizar Señal
- Escribir en Archivo
- Aprendizaje
- Algoritmo ANFIS
- Reproducir Audio
- Grafica de Señales
- Cancelar

Diagrama de Casos de Usos

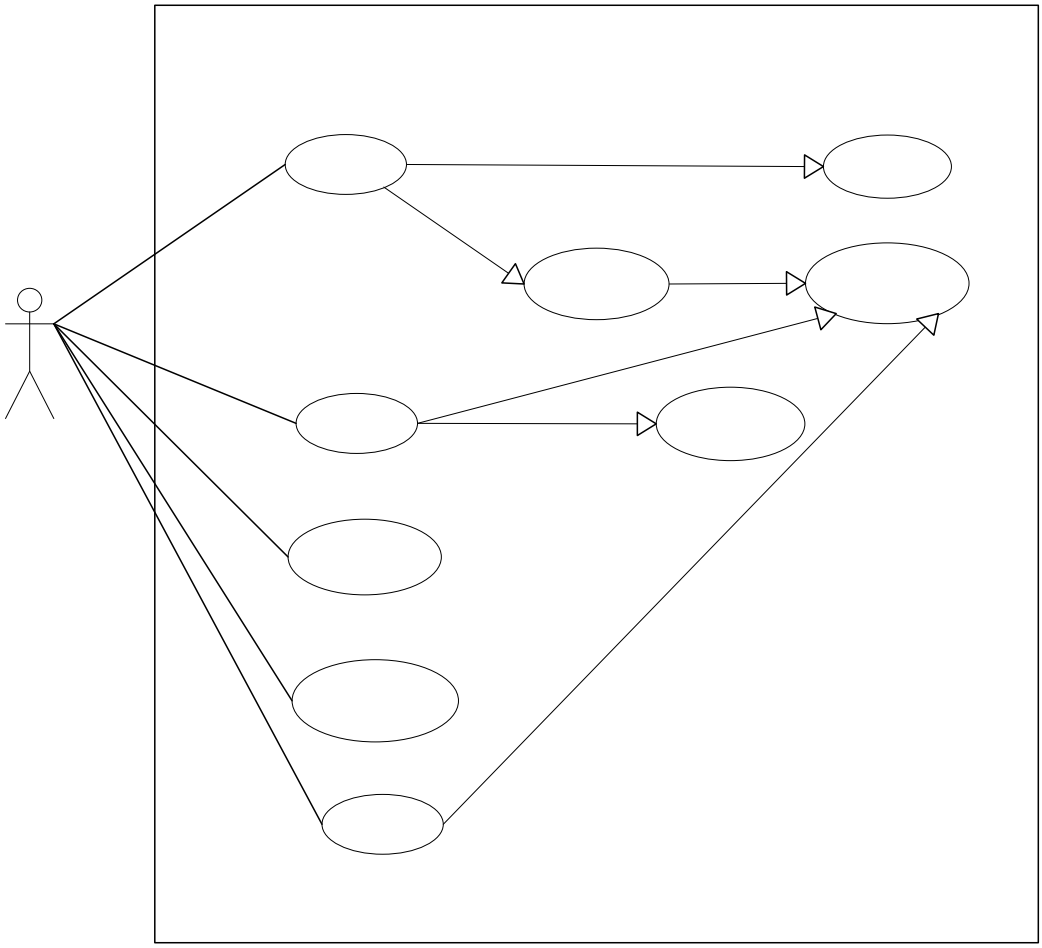


Figura 3: Diagrama de Casos de Usos

Mas detalles de los requerimientos del sistema funcionales y no funcionales están en el anexo A.

Usuarios

Como es de esperar el número de usuarios de la aplicación a desarrollar será uno, ya que este tipo de aplicaciones son claramente mono usuario, debido a la funcionalidad que tienen.

CAPITULO 2

La Red ANFIS: Sistema de Inferencia Borrosa con Base en Redes Adaptivas

En este capítulo se introducirá leves conceptos, principios y ejemplos sencillos para la comprensión de la Red ANFIS, debido a que la red ANFIS es una combinación de Redes Neuronales con Sistemas Difusos es necesario entender en qué consisten estas dos disciplinas, para posteriormente detallar todo lo relacionado a la red ANFIS.

2.1. Introducción a la Redes Neuronales

Resulta irónico pensar que máquinas de cómputo capaces de realizar millones de operaciones por segundo, no sean capaces de entender el significado de las formas visuales o de distinguir entre distintas clases de objetos. Los sistemas secuenciales, son exitosos en la resolución de problemas matemáticos o científicos, en la creación, manipulación y mantenimiento de bases de datos, en el procesamiento de textos, gráficos y auto edición, incluso en funciones de control de electrodomésticos, haciéndolos más eficientes y fáciles de usar, pero definitivamente tienen una gran incapacidad para interpretar el mundo.

Esta dificultad de los sistemas de computo que trabajan bajo la filosofía de los sistemas secuénciales, desarrollados por Von Neuman, ha hecho que un gran número de investigadores centre su atención en el desarrollo de nuevos sistemas de tratamiento de la información, que permitan solucionar problemas cotidianos, tal como lo hace el cerebro humano; este órgano biológico cuenta con varias características deseables para cualquier sistema de procesamiento digital, según la fuente web propuesta en [6], tales como:

- Es robusto y tolerante a fallas, diariamente mueren neuronas sin afectar su desempeño.
- Es flexible, se ajusta a nuevos ambientes por aprendizaje, no hay que programarlo.
- Puede manejar información difusa, con ruido o inconsistente.
- Es altamente paralelo
- Es pequeño, compacto y consume poca energía.

Basados en la eficiencia de los procesos llevados a cabo por el cerebro, e inspirados en su funcionamiento, varios investigadores han desarrollado desde hace más de 30 años la teoría de las Redes Neuronales Artificiales (RNA), las cuales emulan las redes neuronales biológicas, y que se han utilizado para aprender estrategias de solución basadas en ejemplos de comportamiento típico de patrones; estos sistemas no requieren que la tarea a ejecutar se programe, ellos generalizan y aprenden de la experiencia.

La teoría de las RNA ha brindado una alternativa a la computación clásica, para aquellos problemas, en los cuales los métodos tradicionales no han entregado resultados muy convincentes, o poco convenientes. Las aplicaciones más exitosas de las RNA son según la fuente web propuesta en [6]:

- Procesamiento de imágenes y de voz
- Reconocimiento de patrones
- Planeamiento
- Interfaces adaptivas para sistemas Hombre/máquina
- Predicción
- Control y optimización
- Filtrado de señales

Los sistemas de computo tradicional procesan la información en forma secuencial; un computador serial consiste por lo general de un solo procesador que puede manipular instrucciones y datos que se localizan en la memoria, el procesador lee, y ejecuta una a una las instrucciones en la memoria; este sistema serial es secuencial, todo sucede en una sola secuencia determinística

de operaciones. Las RNA no ejecutan instrucciones, responden en paralelo a las entradas que se les presenta. El resultado no se almacena en una posición de memoria, este es el estado de la red para el cual se logra equilibrio. El conocimiento de una red neuronal no se almacena en instrucciones, el poder de la red está en su topología y en los valores de las conexiones (pesos) entre neuronas, según la fuente web propuesta en [6].

Las RNA son una teoría que aún está en proceso de desarrollo, su verdadera potencialidad no se ha alcanzado todavía; aunque los investigadores han desarrollado potentes algoritmos de aprendizaje de gran valor práctico, las representaciones y procedimientos de que se sirve el cerebro, son aún desconocidas. Tarde o temprano los estudios computacionales del aprendizaje con RNA acabarán por converger a los métodos descubiertos por evolución, cuando eso suceda, muchos datos empíricos concernientes al cerebro comenzarán súbitamente a adquirir sentido y se tornarán factibles muchas aplicaciones desconocidas de las redes neuronales, según la fuente web propuesta en [6].

CARACTERISTICAS DE UNA RED NEURONAL ARTIFICIAL

El modelo de una neurona artificial es una imitación del proceso de una neurona biológica, puede también asemejarse a un sumador hecho con un amplificador operacional tal como se ve en la figura 3.

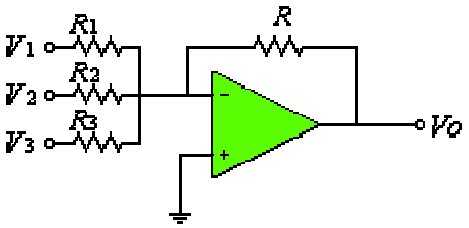


Figura 3: Neurona Artificial

Existen varias formas de nombrar una neurona artificial, es conocida como nodo, neuronodo, celda, unidad o elemento de procesamiento (PE); En la figura 4 se observa un PE en forma general y su similitud con una neurona biológica

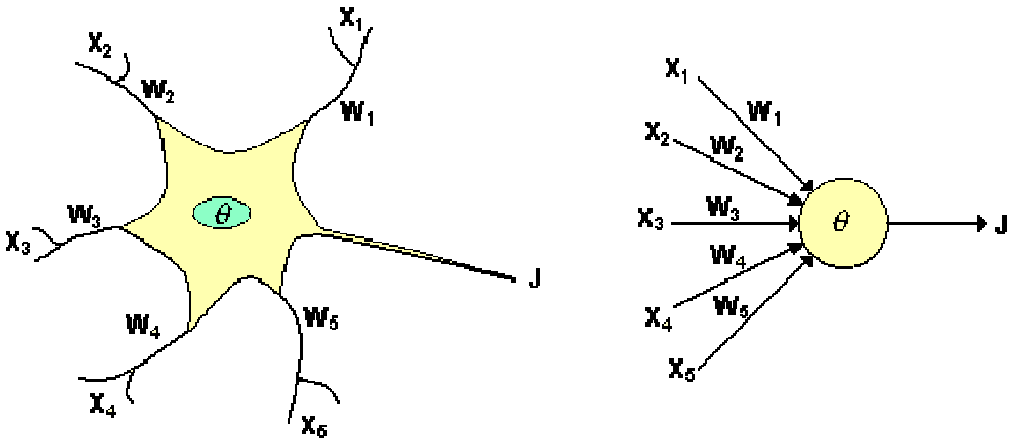


Figura 4: De la neurona biológica a la neurona artificial

De la observación detallada del proceso biológico se han hallado los siguientes análogos con el sistema artificial [6]:

- Las entradas X_i representan las señales que provienen de otras neuronas y que son capturadas por las dendritas.
- Los pesos W_i son la intensidad de la sinapsis que conecta dos neuronas; tanto X_i como W_i son valores reales.
- θ es la función umbral que la neurona debe sobrepasar para activarse; este proceso ocurre biológicamente en el cuerpo de la célula.

Las señales de entrada a una neurona artificial X_1, X_2,.., X_n son variables continuas en lugar de pulsos discretos, como se presentan en una neurona biológica. Cada señal de entrada pasa a través de una ganancia o peso, llamado peso sináptico o fortaleza de la conexión cuya función es análoga a la de la función sináptica de la neurona biológica. Los pesos pueden ser positivos (excitatorios), o negativos (inhibitorios), el nodo sumatorio acumula todas las señales de entradas multiplicadas por los pesos o ponderadas y las pasa a la salida a través de una función umbral o función de transferencia. La entrada neta a cada unidad puede escribirse de la siguiente manera, según la fuente web propuesta en [6]

$$neta_i = \sum_{i=1}^{n} W_i X_i = \vec{X}\,\vec{W}$$

Una idea clara de este proceso se muestra en la figura 5, en donde puede observarse el recorrido de un conjunto de señales que entran a la red.

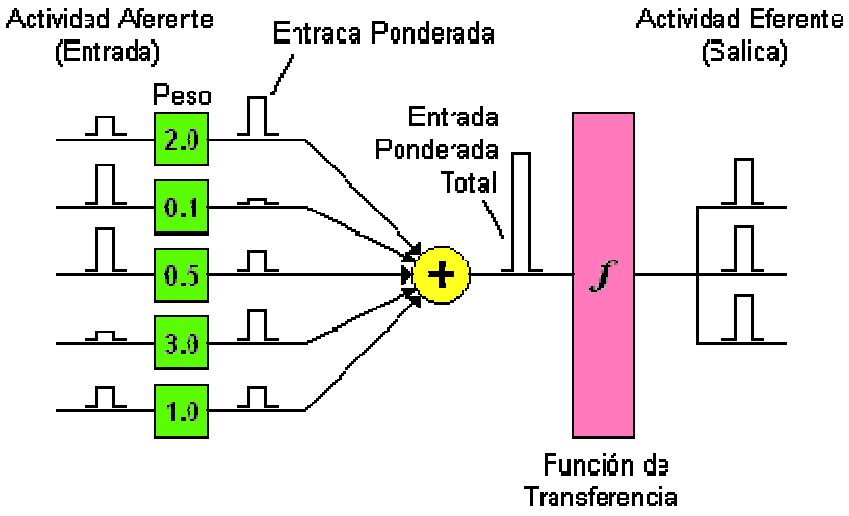


Figura 5: Proceso de una red neuronal

Una vez que se ha calculado la activación del nodo, el valor de salida equivale a

$$x_i = f_i(neta_i)$$

Donde f_i representa la función de activación para esa unidad, que corresponde a la función escogida para transformar la entrada neta en el valor de salida x_i y que depende de las características específicas de cada red.

Funciones de Transferencia: Un modelo más académico que facilita el estudio de una neurona, puede visualizarse en la figura 6

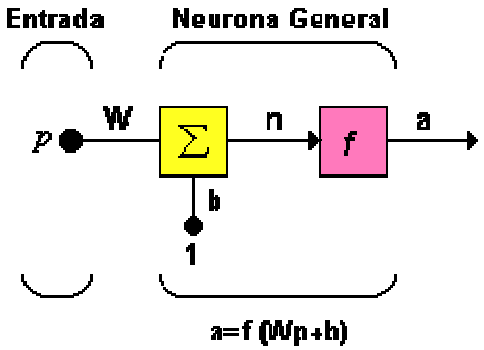


Figura 6: Neurona de una sola entrada

Las entradas a la red serán ahora presentadas en el vector p, que para el caso de una sola neurona contiene solo un elemento, w sigue representando los pesos y la nueva entrada b es una ganancia que refuerza la salida del sumador n, la cual es la salida neta de la red; la salida total está determinada por la función de transferencia , la cual puede ser una función lineal o no lineal de n, y que es escogida dependiendo de las especificaciones del problema que la neurona tenga que resolver; aunque las RNA se inspiren en modelos biológicos no existe ninguna limitación para realizar modificaciones en las funciones de salida, así que se encontrarán modelos artificiales que nada tienen que ver con las características del sistema biológico.

2.2 Introducción a los Sistemas borrosos

En los años 60’s fue el comienzo de la teoría de la lógica difusa, esto fue debido por Lotfi A. Zadeh en 1965 con su publicación de ”conjuntos difusos”, en los 70’s la teoría continuo creciendo, surgieron las aplicaciones reales, es justo decir que la teoría difuso establecida como el campo independiente el gran parte debido a la dedicación y notablemente al trabajo de Zadeh, lo más fundamental

de la teoría de lógica difusa fue propuesto por Zadeh en los 60's y cerca de los 70's, después el propuso los algoritmos difusos en 1968.

La lógica difusa se utiliza en un amplio rango de aplicaciones como:

Controladores de lavadoras, sistemas de aire acondicionado, video cámaras, mecanismos de control, sistemas de control de mini submarinos, controladores del metro así como horno de microondas, según Choque Aspiaxu en [7].

La lógica difusa permite tratar información imprecisa, en términos de conjuntos difusos veremos que estos conjuntos se combinan en reglas para definir acciones, por ejemplo, si la temperatura es alta entonces enfría mucho.

De esta manera, los sistemas de control basados en lógica difusa combinan una variable de entrada (definidos en términos de conjuntos difusos), por grupos que producen uno o varios valores de salida. Hablando ya en términos más rigurosos, la teoría de lógica difusa parte de la teoría clásica de conjuntos, añadiendo una función de pertenencia al conjunto, definida ésta como un número real entre 0 y 1, así se introduce el concepto de lógica difusa determinado a un valor lingüístico. Para cada conjunto o subconjunto difuso se define una función de pertenencia o inclusión $\mu A(t)$, que indica el grado en el cual la variable t está incluida en el concepto que está representado por la etiqueta A

Conjuntos Difusos

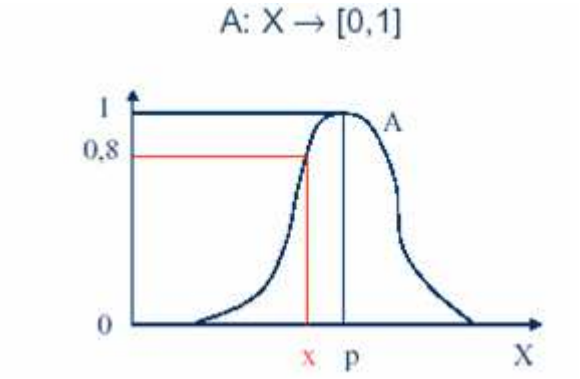


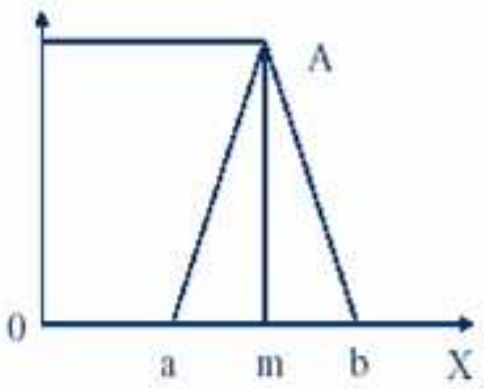
Figura 7: Función de pertenencia

Una situación puede describirse asignando un 1 a todos los elementos incluidos en el conjunto y un 0 a los no incluidos. A la función que asigna estos valores se llama función de inclusión o pertenencia.

Formas de Conjuntos Difusos

- Triangulares

$$A(x) = \begin{cases} 0, & \text{se } x \le a \\ (x-a)/(m-a), & \text{se } x \in [a,m] \\ (b-x)/(b-m), & \text{se } x \in [m,b] \\ 0, & \text{se } x \ge b \end{cases}$$



$$A(x; a, m, b) = \max\{\min[(x-a)/(m-a),(b-x)/(b-m)],0\}$$

Figura 8 Función Triangular

- Función Γ

$$A(x) = \begin{cases} 0, & \text{se } x \le a \\ 1-e^{-k(x-a)^2} & \text{se } x > a \end{cases}$$

$$A(x) = \begin{cases} 0, & \text{se } x \le a \\ \dfrac{k(x-a)^2}{1+k(x-a)^2} & \text{se } x > a \end{cases}$$

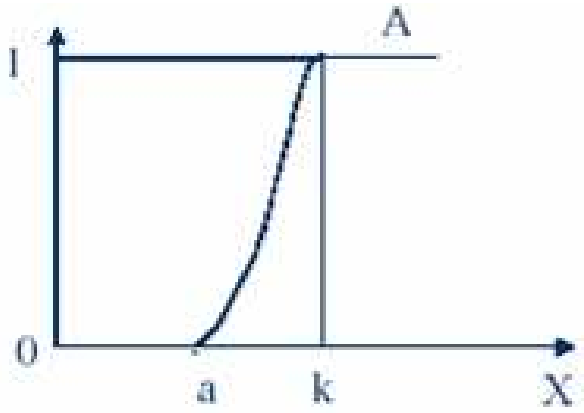


Figura 9: Función Γ

- Función S

$$A(x) = \begin{cases} 0, & \text{se } x \le a \\ 2((x-a)/(b-a))^2, & \text{se } x \in [a,m] \\ 1-2((x-b)/(b-a))^2, & \text{se } x \in [m,b] \\ 1, & \text{se } x > b \end{cases}$$

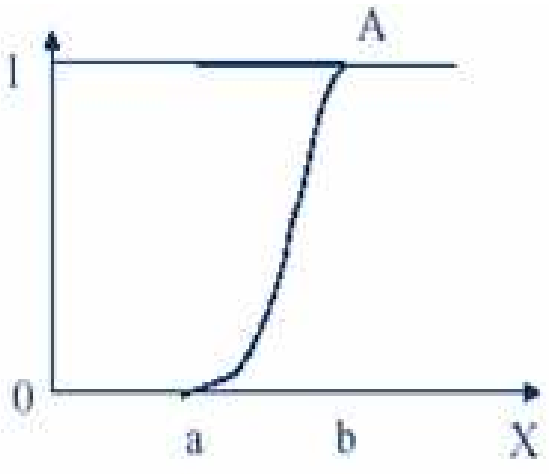
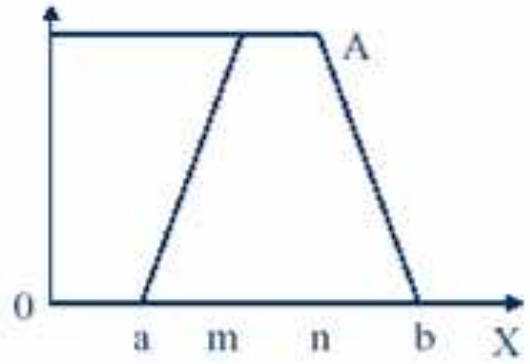


Figura 10: Función S

- Trapezoidal

$$A(x) = \begin{cases} 0, & \text{se } x \le a \\ (x-a)/(m-a), & \text{se } x \in [a,m] \\ 1, & \text{se } x \in [m,n] \\ (b-x)/(b-n), & \text{se } x \in [n,b] \\ 0, & \text{se } x > b \end{cases}$$



$$A(x; a, m, n, b) = \max\{\min[(x-a)/(m-a), 1, (b-x)/(b-m)], 0\}$$

Figura 11: Función Trapezoidal

- Gaussiana

$$A(x) = e^{-k(x-m)^2}$$

onde $k > 0$

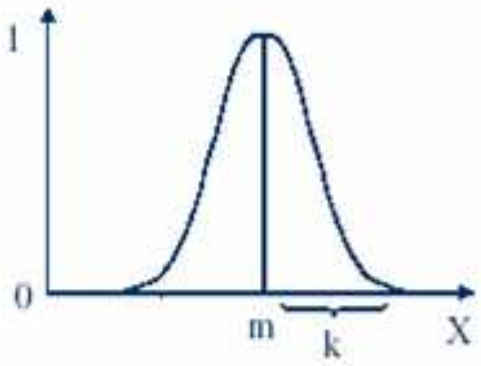


Figura 12: Función Gaussiana

2.3 Sistemas Neuro Difuso

Los sistemas difusos son aquellos que usan lógica difusa que se comprende de juegos difusos y las reglas difusas que es combinando los datos numéricos y lingüísticos. La lógica difusa usa las condiciones del idioma naturales como: frío, caluroso, caliente, pequeño, medio, grande.

La diferencia primaria entre los sistemas difusos y neuro difuso, es que los sistemas difusos son especialistas en la construcción y los sistemas neuro difusos son entrenados a partir de datos, según Jang en [8].

Por otro lado los sistemas híbridos que combinan lógica difusa, redes neuronales, algoritmos genéticos y sistemas expertos proporcionan los métodos más eficientes para resolver una gran variedad de problemas. Cada una de esas técnicas tiene propiedades computacionales particulares (por ejemplo: habilidad de aprender) que las hace óptimas para resolver ciertos problemas. Uno de estos sistemas híbridos corresponde a los sistemas Neuro-Difuso, que combinan las técnicas de redes neuronales artificiales y las técnicas de inferencia difusa, según Jang en [8].

La lógica difusa proporciona un mecanismo de inferencia sobre la incertidumbre y las redes neuronales ofrecen grandes ventajas computacionales, tales como el aprendizaje, adaptación, tolerancia a fallas, el paralelismo y la generalización. Las redes neuronales son usadas para representar los sistemas de inferencia difusa, los mismos que son empleados como sistemas de toma de decisiones. A pesar de que la lógica difusa puede codificar el conocimiento a través de etiquetas lingüísticas, usualmente toma

mucho tiempo definir y ajustar las funciones de pertenencia. Las técnicas de aprendizaje de las redes neuronales pueden automatizar este proceso y reducir sustancialmente el tiempo y el costo de desarrollo al mejorar el desempeño del modelo, según Jang en [8].

Teóricamente las redes neuronales y los sistemas difusos son equivalentes, pero en la práctica cada uno tiene sus propias ventajas y desventajas. En las redes neuronales, el conocimiento se adquiere automáticamente por el algoritmo de backpropagation, pero el proceso de aprendizaje es relativamente lento (gran cantidad de épocas de entrenamiento) y el análisis de la red entrenada es difícil (modelo de caja negra). No es posible extraer el conocimiento estructural (reglas) de la red neuronal ni puede éste integrarse a la información especial sobre el problema en la red neuronal con el fin de simplificar el procedimiento de aprendizaje. Los sistemas difusos son más favorables porque su comportamiento puede ser explicado con base en reglas difusas y, de esta forma, su desempeño puede ser ajustado modificando estas reglas. Sin embargo, la adquisición del conocimiento es difícil, y, además, el universo de discurso de cada variable necesita ser dividido en intervalos, por lo que las aplicaciones de los sistemas difusos se restringen a problemas en los cuales el conocimiento está disponible en un número de variables de entrada pequeño. Para superar el problema de la adquisición del conocimiento, las redes neuronales son extendidas para extraer automáticamente las reglas difusas de los datos numéricos, según Jang en [8].

Reglas Difusas Si- Entonces

Las reglas difusas si-entonces son expresiones de la forma Si A Entonces B, donde A y B son etiquetas de conjuntos difusos caracterizadas por las funciones de pertenencia; además A se la llama antecedente y B consecuente.

Las reglas difusas si-entonces son empleadas con frecuencia para tomar decisiones en un ambiente de incertidumbre e imprecisión. Por ejemplo:

- Si presión es alta, entonces volumen es pequeño

Donde presión y volumen son variables lingüísticas, alta y pequeña son valores lingüísticos o etiquetas que son caracterizadas por las funciones de pertenencia.

Otra forma de las reglas difusas si-entonces, tiene conjuntos difusos involucrados solo en la primera parte. Por ejemplo podemos describir la fuerza resistente de un objeto en movimiento así:

- Si velocidad es alta, entonces Fuerza = k * (velocidad)2

Donde en la primera parte alta es una etiqueta lingüística caracterizada por una función de pertenencia y la parte consecuente esta descrita por una ecuación no difusa de la variable de entrada (velocidad), según Takagi y Sugeno en [2].

Sistemas de inferencias Difusas

Los sistemas de inferencias difusas también llamados sistemas basados en reglas difusas, modelos difusos, memoria asociativa difusa (FAM), o controladores difusos, están compuesto por cinco bloques según Jang en [8] (figura 13).

- **Regla Base (rule base)** contiene un numero de reglas difusas si-entonces
- **Base datos (database)** que define las funciones de pertenencias de los conjuntos difusos usados en las reglas difusas
- **Unidad de toma de decisión (decisión-making unit)** que genera las operaciones de inferencias sobre las reglas
- **Interface de Difusificación (fuzzification interface)** que transforma las entradas a valores lingüístico
- **Interface de Difusificación (Defuzzification interface)** que transforma los resultados de inferencia difusa a salidas

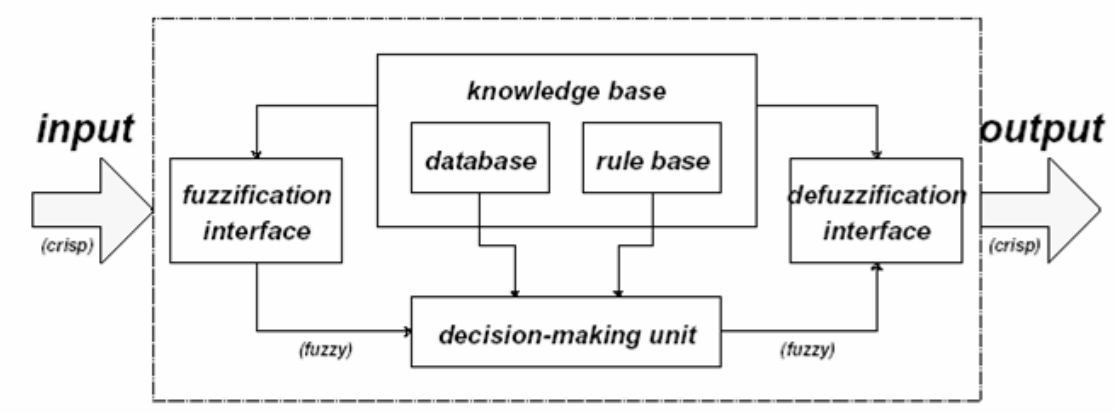


Figura 13 Sistema de Inferencia Difusa

Usualmente la regla base y la base de datos conforma un bloque llamado base de conocimiento.

Los pasos del razonamiento difuso para sistemas de inferencia difusa son:

- Compare las variables de entradas con las funciones de pertenencias en la primera parte para obtener los valores de pertenencias (este paso se llama Difusificación)
- Combine (a través de un operador T-norma, usualmente una multiplicación o mínimo) el valor de pertenencia en la primera parte para obtener el peso de cada regla
- Genere la parte consecuente de cada regla
- Sume toda las parte consecuente de cada regla para producir la salida con sus pesos

Existen tres tipos de razonamiento difusos que son según Jang en [8]:

- Tipo 1: La salida total es el promedio de los pesos de cada regla de salida que son inducidas por los pesos y las funciones de pertenencias de salida.
- Tipo 2: La salida total difusa es derivada de aplicar la operación máxima de calificar las salidas difusas.
- Tipo 3: denominadas Reglas difusas si-entonces de Takagi-Sugeno donde la salida de cada regla es la combinación lineal de las variables de

entradas más un término constante y la salida final es el peso promedio de cada regla de salida.

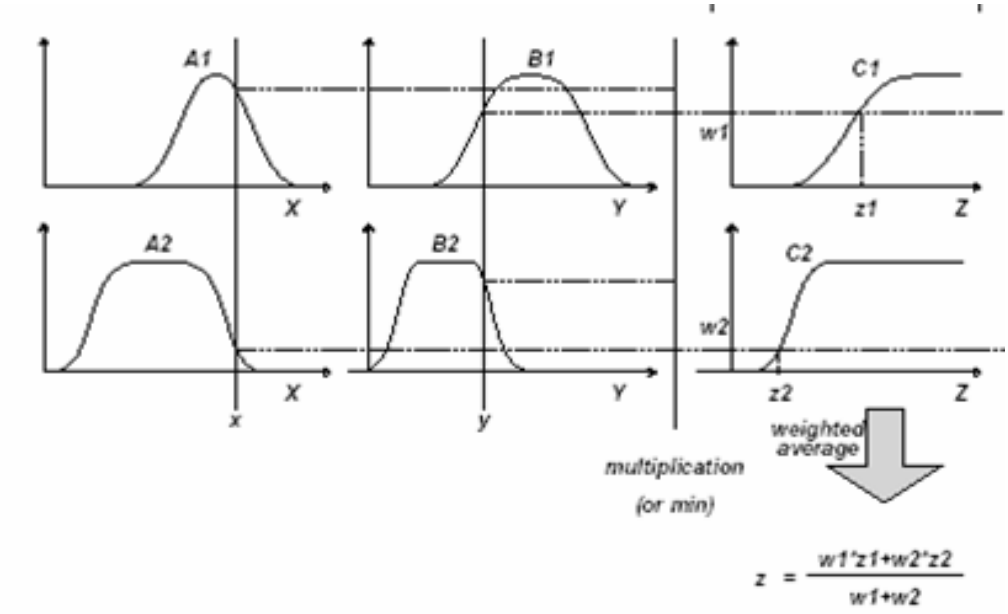


Figura 14 Razonamiento difuso tipo 1

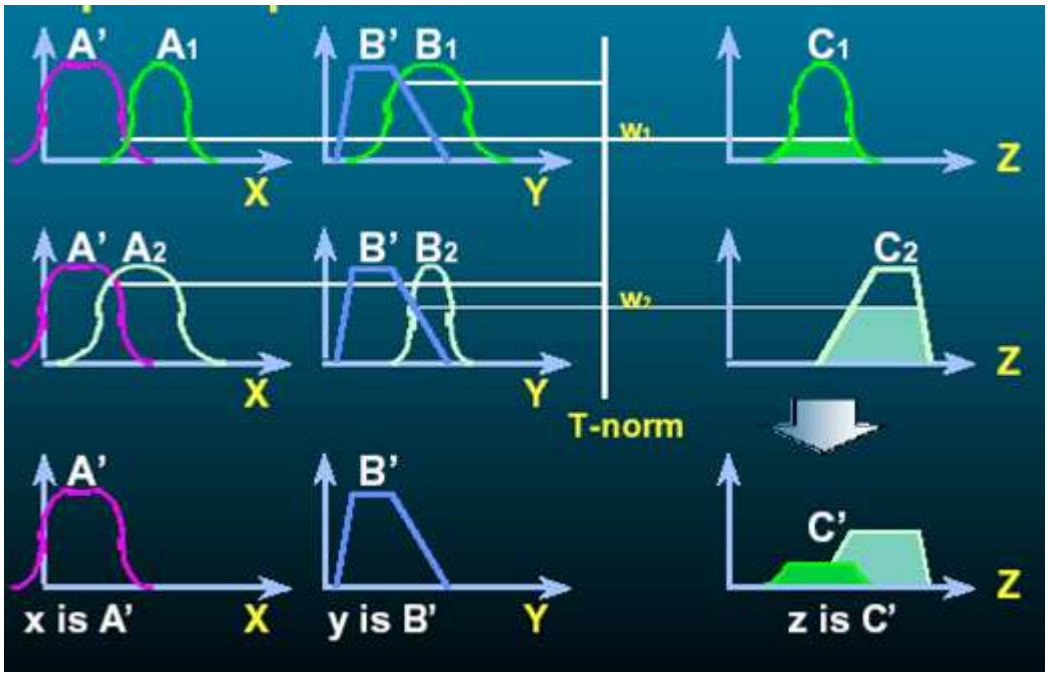


Figura 15 Razonamiento difuso Tipo 2

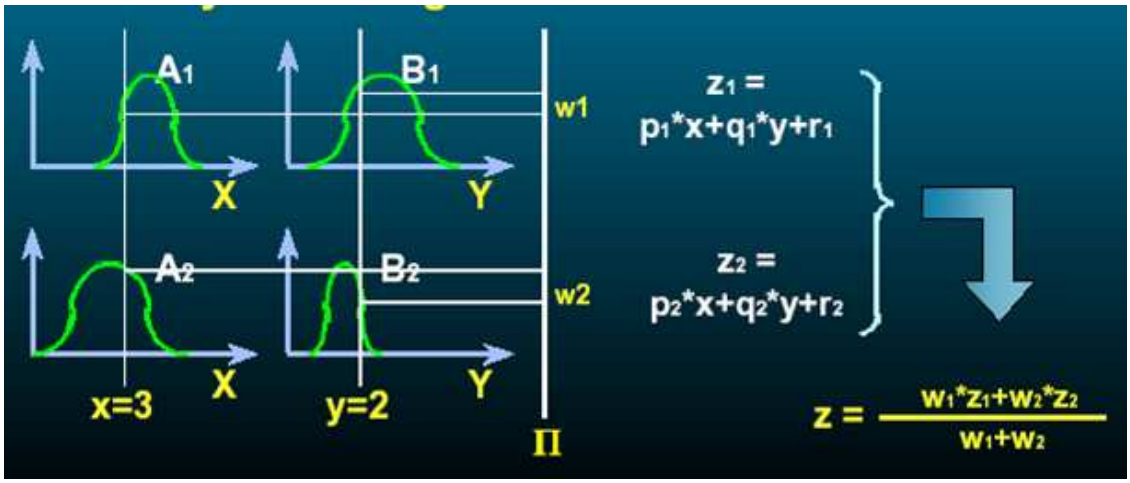


Figura 16 Razonamiento difuso Tipo 3

Redes Adaptivas

Una red adaptiva es una estructura de nodos y enlaces direccionales a través de los cuales los nodos están conectados: Además parte o todos los nodos son adaptivos esto significa que cada salida de estos nodos dependen de los parámetros que pertenecen a estos nodos y a la regla de aprendizaje que especifican cómo estos parámetros deberían ser cambiados para minimizar la medida del error.

La regla básica de aprendizaje de redes adaptivas está basado en el vector gradiente y la regla de la cadena el cual fue propuesto por Werbos en los setenta según Jang en [8].

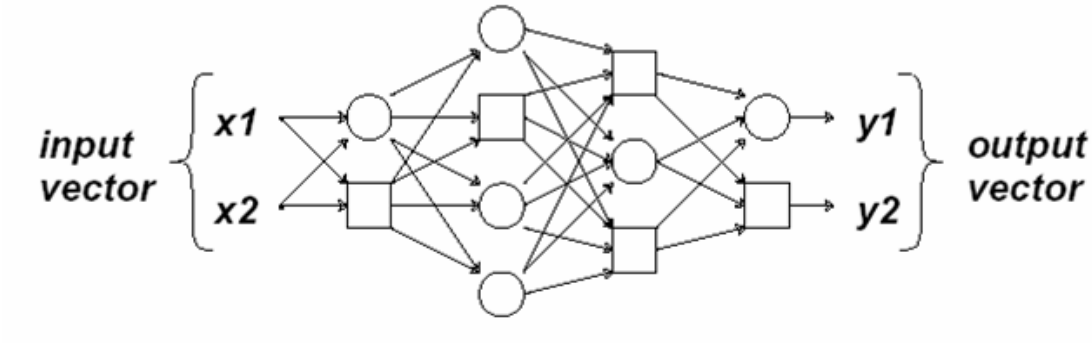


Figura 17 Una Red Adaptiva

Como vemos en la figura la red tiene dos vectores de salida (y1, y2) pero podría tener tres o más incluso una así como las entradas. En nuestro caso usaremos una entrada y una salida.

La regla básica para el aprendizaje de redes adaptivas es muy general y complicado pero en nuestro caso se va a simplificar, como se explicará más adelante.

Arquitectura y Regla de Aprendizaje Básica

Una red adaptiva es una red de multicapas con alimentación hacia adelante en el que cada nodo realiza una función particular en las señales de llegada, usando un conjunto de parámetros relacionados con este nodo.

La naturaleza de las funciones de nodo puede variar de nodo a nodo, y la elección de cada uno de los nodos sobretodo de cual función la red adaptativa está diseñada para implementar.

Hay que tener en cuenta que los vínculos de una red de adaptación sólo indican la dirección del flujo de la señal entre nodos, y no están asociadas con el peso de los vínculos, según Jang en [8].

Para reflejar diferentes capacidades adaptativas, se usan nodos circulares y cuadrados en una red adaptativa. Un nodo cuadrado (nodo adaptativo) tiene parámetros modificables mientras que un nodo circular (nodo fijo) tiene ninguno. El conjunto de parámetros de una red adaptativa es la unión del conjunto de parámetros de cada nodo adaptativo. En orden para obtener un mapeo de entrada - salida deseado, estos parámetros son actualizados de acuerdo a los datos de entrenamiento dados y un procedimiento de actualización basado en gradiente descrito por Jang en [8] a continuación

Supongamos que una red adaptiva tiene L capas y el k-ésimo capa tiene k nodos. Para indicar el nodo en la i-ésima posición de la k-ésima capa la denotamos como (k, i), y su función de nodo de Oi-k, en donde:

$$O_i^k = O_i^k(O_1^{k-1}, \ldots O_{\#(k-1)}^{k-1}, a, b, c, \ldots), \quad \{1\}$$

En donde a, b y c son los parámetros que pertenecen al nodo.

Suponiendo que el conjunto de datos de entrenamiento tiene P entradas, podemos definir la medida del error para la p-ésima entrada ($1 \leq p \leq P$) del conjunto de datos de entrenamiento como la suma de errores cuadráticos

$$E_p = \sum_{m=1}^{\#(L)} (T_{m,p} - O_{m,p}^L)^2 \quad \{2\}$$

Donde $T_{m,p}$ es el m-ésimo componente de p-ésimo vector de salida deseado, y $O^L_{m,p}$ es el m-ésimo componente del vector de salida actual producido por la presentación del p-ésimo vector de entrada, de aquí la medida total del error es

$$E = \sum_{p=1}^{P} E_p$$

{3}

Con el fin de desarrollar un procedimiento de aprendizaje que implemente el gradiente de descenso en E sobre el conjunto de parámetros, primero tenemos que calcular la tasa de error para la de p-ésima entrada y para cada nodo de salida O. La tasa del error para la salida del nodo (L,i) puede ser calculada a partir de la ecuación {2}

$$\frac{\partial E_p}{\partial O_{i,p}^L} = -2(T_{i,p} - O_{i,p}^L)$$

{4}

Para el nodo interno (k,i), la tasa del error puede ser derivada mediante la regla de la cadena:

$$\frac{\partial E_p}{\partial O_{i,p}^k} = \sum_{m=1}^{\#(k+1)} \frac{\partial E_p}{\partial O_{m,p}^{k+1}} \frac{\partial O_{m,p}^{k+1}}{\partial O_{i,p}^k}$$

{5}

Donde 1 ≤ k ≤ L-1; esto es la tasa de error de un nodo interno puede ser expresada como una combinación lineal de las tasas de error de los nodos en la próxima capa.

Ahora si α es un parámetro de una red adaptiva, entonces

$$\frac{\partial E_p}{\partial \alpha} = \sum_{O^* \in S} \frac{\partial E_p}{\partial O^*} \frac{\partial O^*}{\partial \alpha}$$

{6}

Donde S es el conjunto de nodos cuyas salidas dependen de α, entonces la derivada de E con respecto a α es:

$$\frac{\partial E}{\partial \alpha} = \sum_{p=1}^{P} \frac{\partial E_p}{\partial \alpha}$$

{7}

En consecuencia, la fórmula para la actualización los parámetros α es:

$$\triangle \alpha = -\eta \frac{\partial E}{\partial \alpha}$$

{8}

Donde η es la tasa de aprendizaje, que puede ser expresada como:

$$\eta = \frac{k}{\sqrt{\sum_{\alpha}\left(\frac{\partial E}{\partial \alpha}\right)^2}}$$

{9}

Donde k es el tamaño del paso, usualmente se puede variar k para mejorar la rapidez de convergencia y se usan métodos heurísticos

Existen dos paradigmas para aplicar este procedimiento, según Jang en [8]:

- Batch Learning (off-line learning): que actualiza el parámetro α apoyándose en la ecuación (7) y esta actualización la realiza después de presentar todos los ejemplos de entrenamiento (después de cada época).
- Pattern learning (on-line learning): actualiza los parámetros justo después de presentar cada par de entrenamiento entrada/salida. En este caso, la fórmula para actualizar los parámetros se basa en la ecuación (6).

2.4 Descripción de la RED ANFIS

ANFIS (Sistemas de Inferencia difusos basados en redes adaptivas) es una clase de redes adaptivas el cual es funcionalmente equivalente a sistemas de inferencia difuso.

Una red ANFIS es un modelo híbrido donde las reglas se aplican siguiendo una estructura de red tipo neuronal que puede ser interpretado como

una red neuronal con parámetros difusos o como un sistema difuso con parámetros o funcionamiento distribuido, según Jang en [8].

Las capacidades adaptivas de las redes ANFIS las hacen directamente aplicables a una gran cantidad de áreas como la sintonización automatizada de los controladores difusos, en el modelamiento donde se necesita explicar datos pasados y predecir datos futuros, en control adaptativo, en procesamiento y filtrado de señales, en clasificación de datos y extracción de características a partir de ejemplos, entre otros.

Un sistema ANFIS engloba las mejores características de los sistemas difusos y de las redes neuronales. De los primeros utiliza la representación del conocimiento previo en un conjunto de restricciones (que se representan en la topología de la red) para reducir el espacio de búsqueda de optimización, mientras que de las redes neuronales emplean la adaptación de propagación inversa a la red estructurada para automatizar el ajuste de los parámetros, según Jang en [8].

La parte de la premisa de una regla define un subespacio difuso, mientras que el consecuente especifica la salida dentro de ese subespacio.

La estructura de los sistemas ANFIS permite utilizar métodos cualitativos y cuantitativos en la construcción de modelos. Además permite integrar, a la información incluida dentro de un conjunto de datos, el conocimiento de expertos expresados en forma lingüística y a través de la teoría de conjuntos difusos, expresados con base

La arquitectura ANFIS es la siguiente:

Este sistema híbrido neuro-difuso es funcionalmente equivalente al mecanismo de inferencia Takagi-Sugeno (T-S) de primer orden, según Jang en [8].

Regla 1: Si x es A1 and y es B1, entonces f1 = p1 x + q1 y + r1
Regla 2: Si x es A2 and y es B2, entonces f2 = p2 x + q2 y + r2

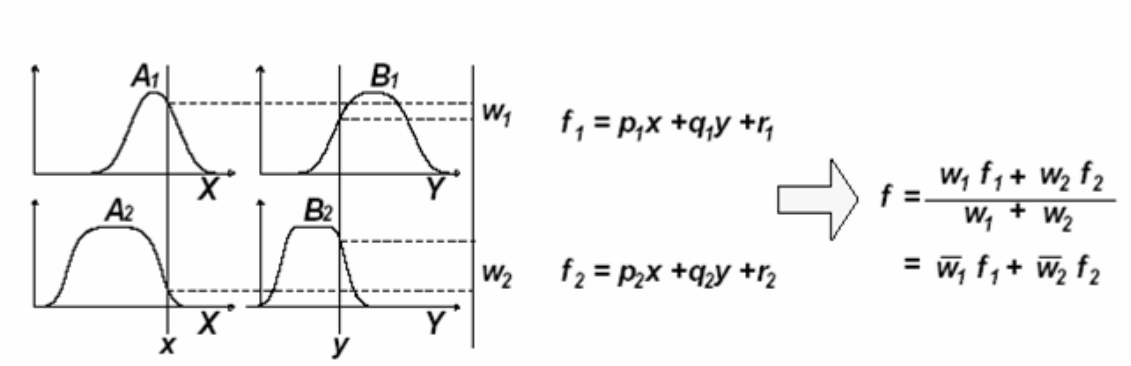


Figura 18 Razonamiento ANFIS

Donde A1, A2, B1, B2 son funciones de pertenencias (Conjuntos difusos)

Los niveles de activación de las reglas se calculan como wi = Ai(x). Bi(y) , i=1,2.., donde el operador lógico and puede ser modelado por una t-norma continua (producto). Las salidas individuales de cada regla son obtenidas como una combinación lineal entre los parámetros del antecedente de cada regla: fi = pi x+qi y+ri, i=1,2..,,. La salida de control del modelo se obtiene por la

normalización de los grados de activación de las reglas por la salida individual de cada regla:

W1 y w2 son los valores normalizados de w1 y w2 y con respecto a la suma w1+w2 La red neuronal híbrida que representa este tipo de inferencia es una red adaptable con 5 capas, donde cada capa representa una operación del mecanismo de inferencia difuso. Esta red se muestra en la figura 19.

En esta arquitectura, todos los nodos de una misma capa tienen la misma función (los nodos representados con cuadros son nodos adaptables, es decir, sus parámetros son ajustables). La estructura de la red ANFIS consiste de cinco capas.

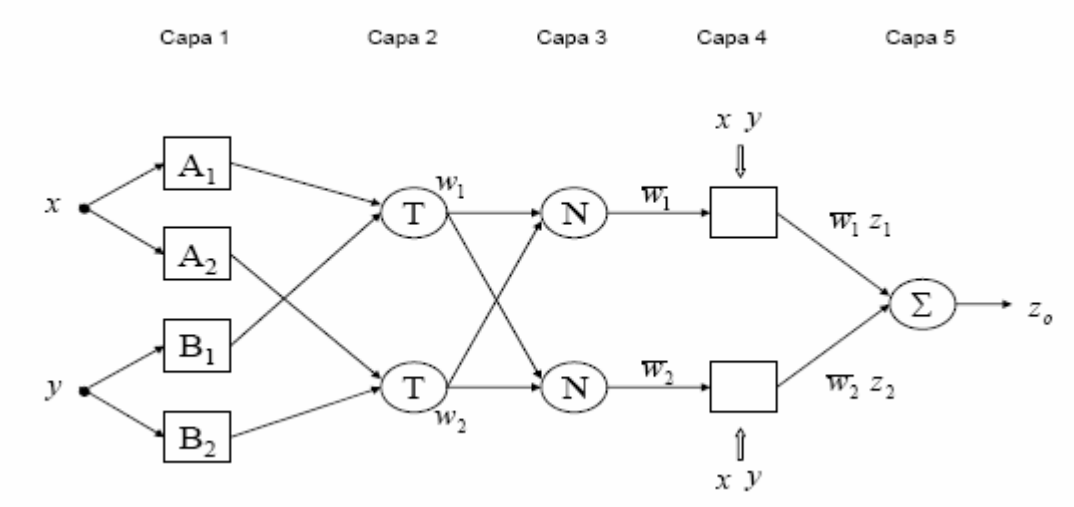


Figura 19 Red adaptiva equivalente ANFIS

Capa 1: Las entradas en esta capa corresponden a las entradas *x* y *y*, y la salida del nodo es el grado de pertenencia para el cual la variable de entrada satisface el término lingüístico asociado a este nodo.

$$O_i^1 = A_i(x)$$

Capa 2: Cada nodo calcula el grado de activación de la regla asociada a dicho nodo. Ambos nodos están representados con una T en figura 2, por el hecho de que ellos pueden representar cualquier t-norma para modelar la operación lógica and. Los nodos de esta capa son conocidos como nodos de reglas.

$$O_i^2 = w_i = A_i(x) \cdot B_i(y), i = 1,2..$$

Capa 3: Cada nodo en esta capa está representado por una N en la figura 2, para indicar la normalización de los grados de activación. La salida del nodo es el grado de activación normalizado (con respecto a la suma de los grados de activación) de la regla

$$O_i^3 = \overline{w_i} = \frac{w_i}{w_1 + w_2}, i = 1,2.$$

Capa 4: La salida de los nodos corresponde al producto entre el grado de activación normalizado por la salida individual de cada regla.

$$O_i^4 = \overline{w}_i f_i = \overline{w}_i (p_i x + q_i y + r_i)$$

pi, qi, ,ri y forman el conjunto de parámetros. Los parámetros de esta capa se conocen como parámetros del consecuente.

Esos parámetros, como se puede ver, son los coeficientes de las funciones lineales que forman el consecuente de las reglas. Son parámetros ajustables, como los de la capa 1.

Capa 5: El único nodo de esta capa calcula la salida total del sistema (agregación) como la suma de todas las entradas individuales de este nodo.

$$O_1^5 = \sum_i \overline{w}_i f_i = \frac{\sum_i w_i f_i}{\sum_i w_i}$$

En resumen, cada una de las capas tiene una misión concreta dentro del sistema:

- La primera capa representa la capa de pertenencia.
- La segunda capa se usa para generar el grado de disparo de la regla (T-norma)
- La tercera capa actúa de normalizador.
- La cuarta capa calcula la salida
- La última capa combina todas las salidas en una en su único nodo.

El modelo ANFIS tiene dos conjuntos de parámetros que deben ser entrenados: los parámetros del antecedente (constantes que caracterizan las funciones de pertenencia) y los parámetros del consecuente (parámetros lineales de la salida del modelo de inferencia).

El paradigma de aprendizaje del modelo ANFIS emplea algoritmos de gradiente descendiente para optimizar los parámetros del antecedente y el algoritmo de mínimos cuadrados para determinar los parámetros lineales del consecuente. Debido a esta combinación se lo conoce como regla de aprendizaje híbrido, según Jang en [8].

Para aplicar el aprendizaje híbrido en grupo, en cada época de entrenamiento debe ejecutarse un paso forward y un paso backward. En el paso forward, los parámetros de las funciones de pertenencia son inicializados y se presenta un vector de entrada-salida, se calculan las salidas del nodo para cada capa de la red y entonces los parámetros del consecuente son calculados usando el método de mínimos cuadrados, según Jang en [8].

Una vez identificados los parámetros del consecuente, el error es calculado como la diferencia entre la salida de la red y la salida deseada presentada en los pares de entrenamiento. Una de las medidas más usadas para el error de entrenamiento es la suma de errores cuadráticos, definida como:

$$ECM = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Los Yi corresponden a los patrones de entrenamiento proporcionados (salidas deseadas) y Yi sombrero es la correspondiente salida de la red. En el paso backward, las señales de error son propagadas desde la salida, en dirección de las entradas; el vector gradiente es acumulado para cada dato de entrenamiento. Al final del paso backward para todos los datos de entrenamiento, los parámetros en la capa 1 (parámetros de las funciones de pertenencia) son actualizados por el método descendente

Esto constituye la forma más corriente de entrenar un sistema ANFIS, pero de hecho, hay 4 métodos de actualización de parámetros que según sus complejidades son según Jang en [8]:

- Sólo gradiente descendente: Todos los parámetros se actualizan por esta técnica.
- Gradiente descendente y un paso de mínimos cuadrados: Sólo se aplican una vez al principio los mínimos cuadrados, para obtener los valores iniciales de los parámetros del consecuente y luego se utiliza el gradiente descendente para actualizar todos los parámetros.
- Gradiente descendente y mínimos cuadrados: Es el entrenamiento híbrido que se ha descrito.
- Mínimos cuadrados sólo: Lineariza el ANFIS. Utiliza los parámetros de las premisas y el algoritmo del filtro de Kalman extendido para actualizar los parámetros.

Se debe elegir el método más adecuado en función de la complejidad de computación y de los resultados obtenidos. En general, el método de mínimos cuadrados suele llevar una mayor computación que el gradiente descendente.

CAPITULO 3

Diseño de CACIQUE

La esencia del diseño del software es la toma de decisiones sobre la organización lógica del software, el siguiente capítulo presentará como aplicar la Red ANFIS al problema planteado para luego a través de la arquitectura escogida se diseñen los modelos que describen la naturaleza del mismo usando la metodología del diseño orientado a objetos utilizando el UML.

3.1Aplicación de la Red ANFIS a la sintetización de música y de voz

Para señales de voz se usará como ejemplo la palabra “MANNA”, la palabra “HOLA”, y la palabra “POLITECNICA”; la primera es para comparar con los resultados obtenidos por otros investigadores del tema; se escogió la palabra “HOLA”, por su simplicidad, ya que es una palabra que tiene dos silabas y no tiene acento prosódico; y para observar el comportamiento de la red con una palabra polisílaba y con acento prosódico se escogió “POLITECNICA” y para las notas musicales, la nota LA, SOL. SOL# de una guitarra eléctrica.

Para identificar la mejor configuración del modelo ANFIS para pronosticar las señales se han tomado dos consideraciones:

a) Se construirán un modelo ANFIS por cada cincuenta pares cardinales, determinándose en total n/50 modelos por cada señal; donde n es el número de pares cardinales de los datos muestreados; el dividendo de 50 corresponde a un valor experimental encontrado, para que el algoritmo logre terminar, esto se debe a que el algoritmo no converge con más de cincuenta pares cardinales debido a la cantidad de datos de entrada.

b) Se considerará que el orden autorregresivo del modelo es 1 es decir, la red adaptable tendrá solo una entrada correspondiente al tiempo de la señal acústica y una salida correspondiente al valor de amplitud que se desea sintetizar.

La frecuencia de muestreo para nuestro estudio será de n/t KHz; donde t es el tiempo de duración de la señal y n el número de pares cardinales de datos.

La cantidad de funciones de pertenencia *nFP*, empleadas para *fuzzyficar* la entrada al modelo dependerá de la cantidad de información (datos) disponible, ya que debe guardar relación con el número de parámetros no lineales que deberán ser calculados para la salida del modelo.

Cada función de pertenencia presente en la entrada del modelo está relacionada con el número de parámetros a calcularse en la capa 4 de la red de la figura 23, por tanto, *nFP* deberá variar desde 2 hasta *nFPmax* según Jang en [8].

Se tendrá un máximo de 30 funciones de pertenencia para la entrada del modelo ANFIS, esto se hace para establecer un límite, pero se puede extender, si algún otro investigador lo desee hacer.

Se consideraron además 2 tipos de funciones de pertenencia, *tFP*, para representar la entrada al modelo; las funciones de pertenencia empleadas son, en orden de utilización, las funciones tipo triangular (*trimf*), gaussiana (*gaussmf*). Se escogieron estas dos debido a que han sido exitosas en múltiples aplicaciones en el área de genética, incertidumbre, sistemas predictivos, etc.

De acuerdo a la metodología heurística para identificación de modelos ANFIS propuesta por Jang en [8] y aplicada por Riyanto en [9].

Se realizaron 30 x 2 simulaciones, lo que determinó 60 posibles modelos ANFIS.

Para determinar la mejor combinación de *nFP* x *tFP* se utilizó como criterio de selección el mínimo Error Cuadrático Medio determinado en cada una de las simulaciones (entrenamientos de la red adaptable) con el mínimo de funciones de pertenencias. Cada entrenamiento de la red adaptable fue realizado considerando un número máximo de 300 épocas.

3.2 Diseño Arquitectónico de CACIQUE

El diseño arquitectónico es a primera etapa en el proceso de diseño y representa un enlace crítico entre los procesos de ingeniería de diseño y de requerimientos. El proceso de diseño arquitectónico está relacionado con el establecimiento de un marco estructural básico que identifica los principales componentes de un sistema y las comunicaciones entre estos componentes.

Estructuración del Sistema

El sistema trabajará bajo una estructura de tres capas utilizando el modelo de maquina abstracta tal y como se ilustra en la figura 20; los motivos por la cual se escogió está estructura son los siguientes:

- Soporte del desarrollo incremental; a medida que se desarrolla una capa, algunos de los servicios proporcionados por esa capa pueden estar disponibles para los usuarios
- Portabilidad y soporte a cambios

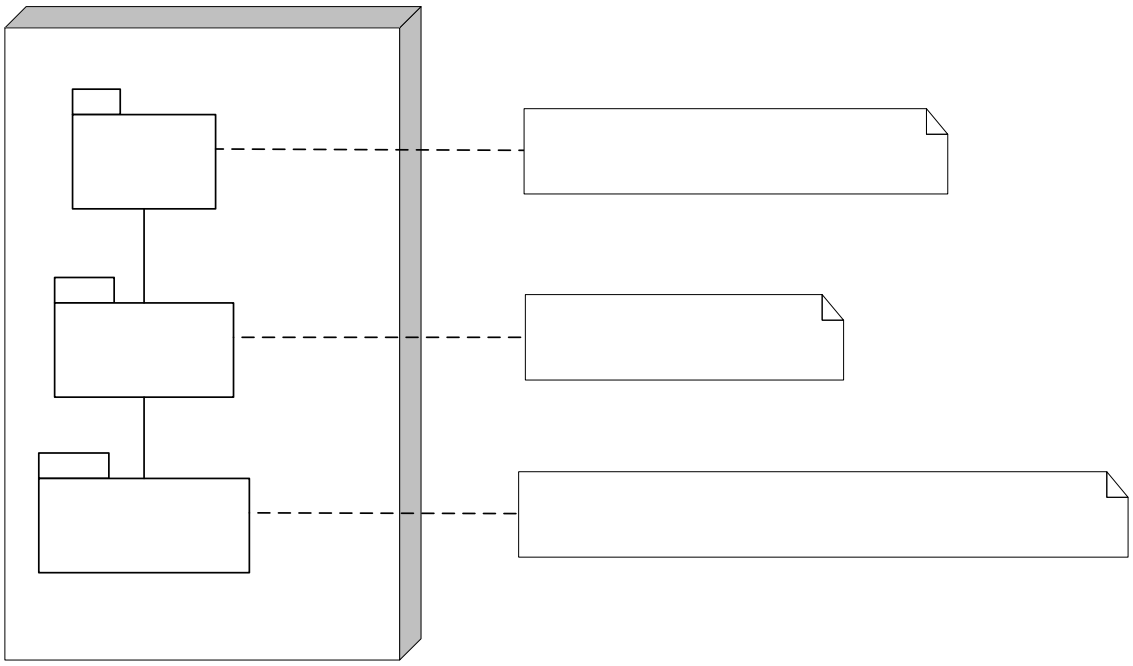


Figura 20: Estructura de CACIQUE

CACIQUE

«subsistema»
Interfase

«subsistema»
Control de Datos

El detalle de cada capa se ilustra a continuación:

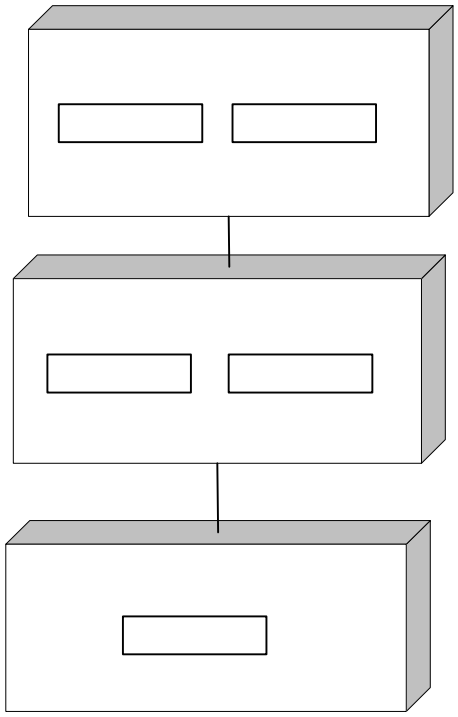


Figura 21: Detalle de la estructura de CACIQUE

Modelado de Control

Para el modelado control se ha establecido un control centralizado usando el modelo de administrador que es el que controla la parada, arranque y coordinación de otros procesos del sistema; para el subsistema Interface se utilizará una clase CACIQUE que administrará el formulario principal y el formulario de graficas, mientras en el subsistema Control de Datos se utilizará

una clase Aprendizaje que administrará la clase ANFIS y la clase Archivo, la clase Inicio que administrará a la clase Convertidor y la clase Archivo.

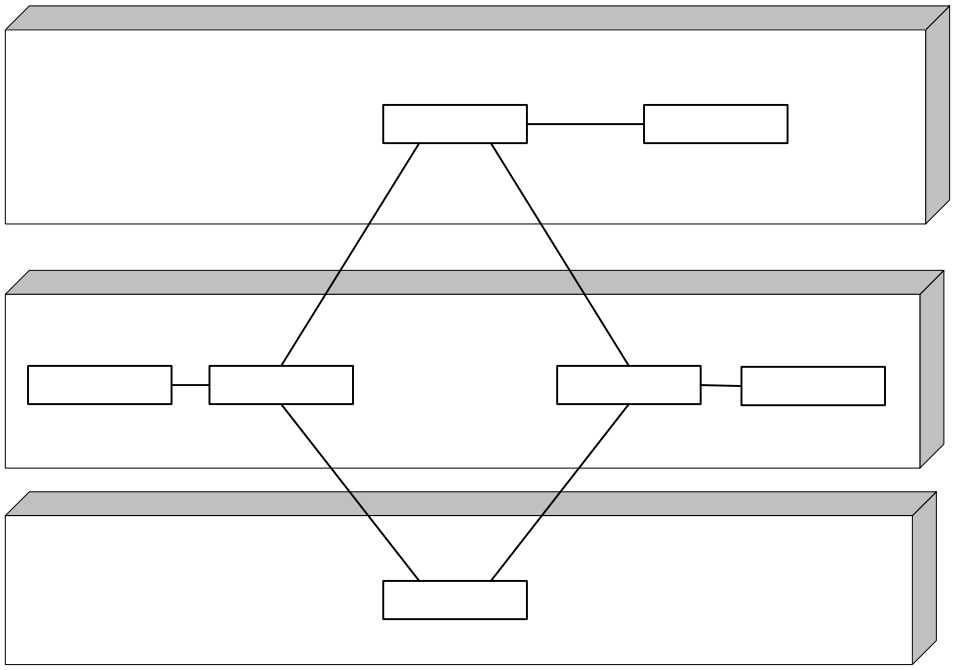
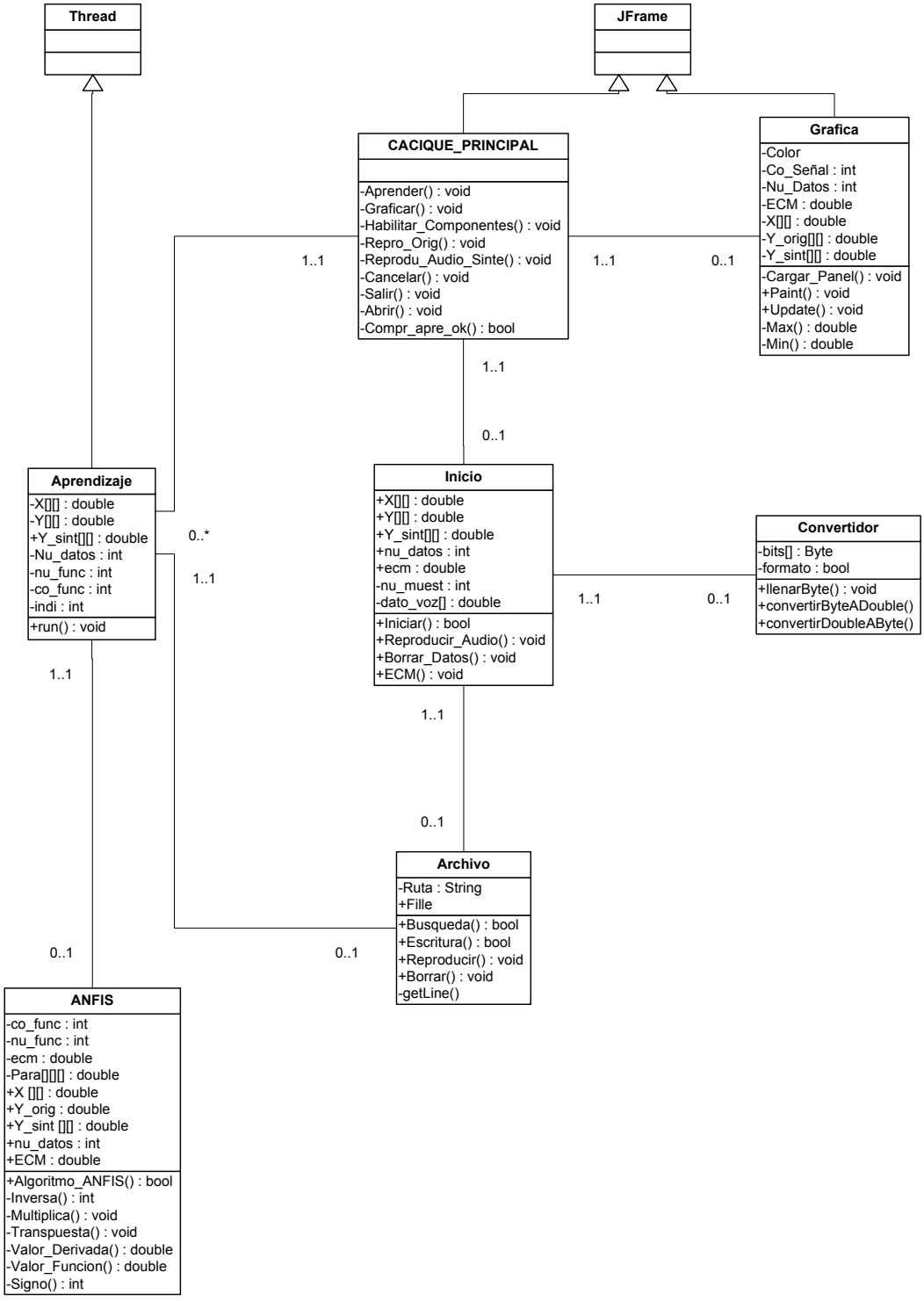


Figura 22: Modelado de control de CACIQUE

Relaciones entre Clases


Thread
JFrame
CACIQUE_PRINCIPAL
-Aprender() : void
-Graficar() : void
-Habilitar_Componentes() : void
-Repro_Orig() : void
-Reprodu_Audio_Sinte() : void
-Cancelar() : void
-Salir() : void
-Abrir() : void
-Compr_apre_ok() : bool
Grafica
-Color
-Co_Señal : int
-Nu_Datos : int
-ECM : double
-X[][] : double
-Y_orig[][] : double
-Y_sint[][] : double
-Cargar_Panel() : void
+Paint() : void
+Update() : void
-Max() : double
-Min() : double
1..1
1..1
0..1
1..1
0..1
Aprendizaje
-X[][] : double
-Y[][] : double
+Y_sint[][] : double
-Nu_datos : int
-nu_func : int
-co_func : int
-indi : int
+run() : void
Inicio
+X[][] : double
+Y[][] : double
+Y_sint[][] : double
+nu_datos : int
+ecm : double
-nu_muest : int
-dato_voz[] : double
+Iniciar() : bool
+Reproducir_Audio() : void
+Borrar_Datos() : void
+ECM() : void
Convertidor
-bits[] : Byte
-formato : bool
+llenarByte() : void
+convertirByteADouble()
+convertirDoubleAByte()
0..*
1..1
1..1
0..1
1..1
1..1
0..1
0..1
0..1
Archivo
-Ruta : String
+Fille
+Busqueda() : bool
+Escritura() : bool
+Reproducir() : void
+Borrar() : void
-getLine()
ANFIS
-co_func : int
-nu_func : int
-ecm : double
-Para[][][] : double
+X [][] : double
+Y_orig : double
+Y_sint [][] : double
+nu_datos : int
+ECM : double
+Algoritmo_ANFIS() : bool
-Inversa() : int
-Multiplica() : void
-Transpuesta() : void
-Valor_Derivada() : double
-Valor_Funcion() : double
-Signo() : int

3.3 Vista Panorámica del Diseño Detallado de CACIQUE

Las clases detalladas tienen la siguiente estructura:

Nombre	
Declaración	
Descripción	
Atributos	
Nombre	
Tipo	
Declaración	
Descripción	
Métodos	
Nombre	
Tipo	
Declaración	
Descripción	
Entrada	
Salida	

En el Anexo 3 están las clases detalladas del sistema.

CAPITULO 4

Implementación de CACIQUE

En este capítulo se presentará como fue implementado CACIQUE, desde la elección del lenguaje de programación, los problemas encontrados y una vista panorámica de las clases codificadas en lenguaje Java.

4.1 El entorno de desarrollo de CACIQUE

Como lenguaje de programación se ha usado Java, el cual es un lenguaje orientado a objetos con cualidades como simple, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico.

La versión del kit de desarrollo java usado es jdk1.6.0_16, que es un conjunto de programas y librerías que permiten desarrollar, compilar y ejecutar.

Como entorno de desarrollo integrado IDEs se ha utilizado el NetBeans 5.0, esta herramienta permite codificar, compilar y ejecutar código java sin necesidad de cambiar de aplicación.

El lenguaje Java es muy versátil, pudiendo ser utilizados para varios tipos de aplicaciones, para diferentes medios y funcionalidades, pero el usado en este trabajo es el tipo Stand-alone que es una aplicación que tiene acceso total a los recursos del sistema, memoria, disco, dispositivos, etc.

4.2 Obtención de las señales de música y de voz a sintetizar

Cuando se entona un instrumento musical o al hablar se genera ondas sonoras en el ambiente, dichas ondas tienen forma sinusoidal a lo largo del tiempo en amplitud.

Los Archivos de audio a usar son de extensión WAV; para los archivos de voz hemos obtenidos palabras en nuestro léxico a través de una grabadora, y para los archivos de notas musicales la hemos obtenido del Internet.

4.3 Problemas encontrados en la implementación de CACIQUE

Durante la implementación solo se detectaron inconvenientes en la codificación del algoritmo ANFIS para su convergencia, ya que en algunas ocasiones se pasaban de las 300 épocas de aprendizajes y el programa se quedaba en un lazo infinito; esto se debe a que:

- La tasa de aprendizaje no debe ser seteada al azar, se hizo un análisis de convergencia para nuestro caso y llegamos a la conclusión de que debe ser seteada con 45 para los parámetros a de las funciones de membrecías y 0.05 para los parámetros b de las funciones de membrecías
- De igual manera al setear los parámetros a y b de las funciones de membrecías se analizo y se concluyo que para su convergencia se debe usar para la función triangular :

  - Parámetro a: Valor inicial X +14 + contador
  - Parámetro b: 80
- Donde Valor inicial X es el primer valor de entrada de señal X del tiempo y el contador es un incrementador que varía de acuerdo al número de funciones de membrecías a usar.
- Y para la función Gaussiana:

- Parámetro a: 5
- Parámetro b: Valor inicial X +2 * contador

- Al obtener los datos muestreados que son alrededor de más 33000 datos por archivo, se trato de realizar el aprendizaje y se observo un disparo de memoria física e inconvergencia del algoritmo; posteriormente se concluyo que eran demasiado datos para el algoritmo lo que llevo a que se trabaje por segmentos de 50 pares cardinales cada uno, esto conlleva a realizar más de 660 aprendizajes promedio por archivo, lo cual es muy engorroso y molestoso para el usuario hacer más 660 clics; por ende se tomaron solo 500 pares cardinales de las muestras originales para poder sintetizar estos datos.
- Por otro lado el método heurístico usado para la convergencia es el siguiente:

  - Si el error cuadrático medio se decrementa cuatro veces consecutivas el valor de la tasa de aprendizaje debe incrementarse en un 10%
  - Si el error cuadrático medio incrementa y se decrementa dos veces consecutivas el valor de la tasa de aprendizaje debe decrementarse en un 10%

4.4 Vista panorámica de la codificación de las clases de CACIQUE

En el Anexo 5 se presentan una vista general de cada una de las clases del producto CACIQUE.

CAPITULO 5

Pruebas y Resultados

En este capítulo presentaremos los resultados a las pruebas hechas a CACIQUE, tanto como para notas musicales como para señales de voz.

En las pruebas con notas musicales se escogió las notas LA, SOL, SOL# de una guitarra eléctrica, estos archivos .WAV se consiguieron de la Internet

En las pruebas con señales de voz se escogió las palabras Hola, Politécnica y Manna, las dos primeras fueron grabadas en archivos .WAV de la

propia voz del autor de la tesis, y la ultima para poder realizar comparaciones con los resultados de Axel Röben en su trabajo con la misma palabra en [1].

Cabe puntualizar que CACIQUE divide la señal acústica en 10 segmentos de pares cardinales x-y debido a que una red ANFIS no puede converger una extensa cantidad de datos; esto quiere decir que por cada prueba habrá 10 redes ANFIS las cuales deberán aprender.

5.1 El entorno de pruebas de CACIQUE

El presente plan de pruebas contiene la descripción de los casos de prueba definidos con el fin de validar y verificar que el sistema puede sintetizar ondas sonoras.

Para esto se probará:

- Para el aprendizaje no debe pasar las 30 funciones de pertenencias como un máximo establecido
- No pasarse de las 300 épocas como máximo para el aprendizaje
- Que converja con los tipos de funciones de membrecías

Debido a la gran cantidad de datos que poseen las señales se ha segmentado en pares de 50 entradas-salidas

Características Excluidas de las Pruebas

El presente plan de pruebas no cubre y, por lo tanto no incluye los siguientes aspectos:

- Pruebas físicas o de elementos hardware
- Pruebas de carga y/ o rendimiento.
- Pruebas de precisión, resolución o funcionamiento del equipo en cuanto a la veracidad o no de las posiciones enviadas.

Criterios de Validación

En la descripción de cada uno de los casos de prueba contenidos en el presente documento se describen los resultados esperados del caso de prueba. Se considerará que una prueba ha pasado con éxito cuando los resultados esperados coincidan con los descritos en el caso de prueba.

En caso de no coincidencia, la persona encargada de la prueba (o sea yo) determinará si la discordancia supone un fallo en la validación del sistema y si debe continuarse con los restantes casos de prueba o bien dar por finalizada la validación del sistema.

5.2 Pruebas de CACIQUE con notas musicales

Por cada segmento de nota musical se puede obtener 58 posibles modelos ANFIS (resultado de multiplicar 2 tipos funciones de pertenencias con 29 números funciones de pertenencias), de los cuales algunos convergen y otros no; los modelos que convergen son separados para escoger el que obtuvo el mínimo error cuadrático medio, en cual resultaron los siguientes:

Prueba de la nota musical LA de una guitarra eléctrica

La grafica de la señal se ilustra en la figura 23:

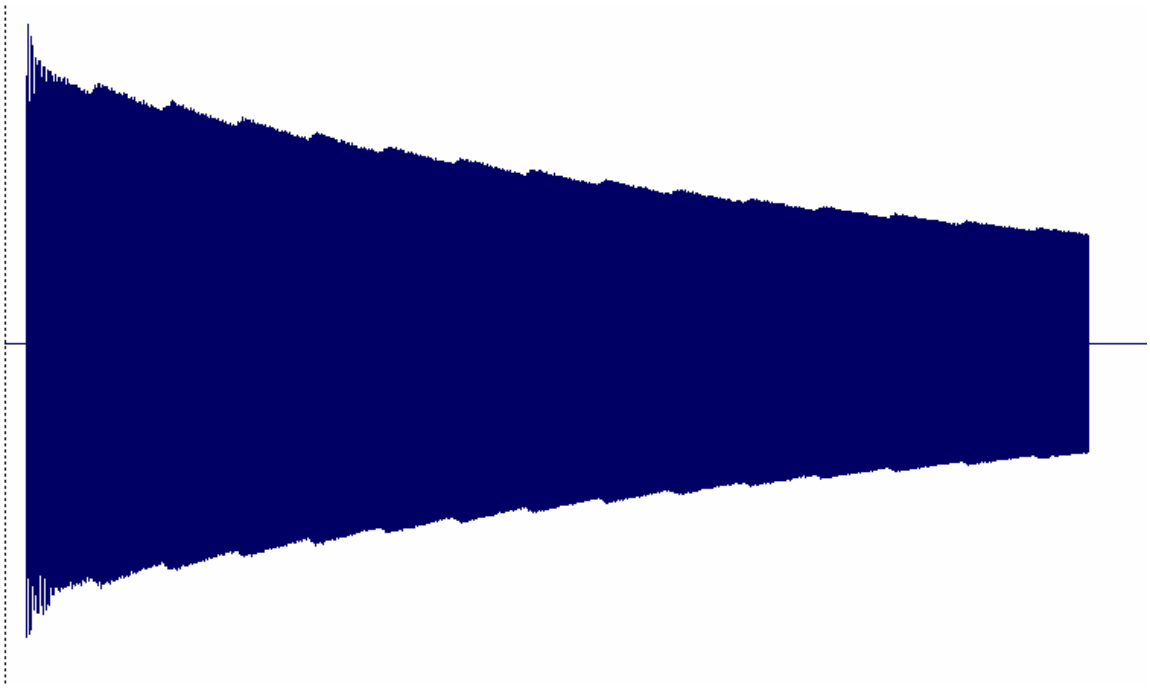


Figura 23: Señal de la nota musical LA de una guitarra eléctrica

Se obtuvo los siguientes resultados mostrados en la tabla 1; observamos que por cada segmento de la función el mínimo error cuadrático fue utilizando 25 funciones Gaussiana, finalmente la figura 24 muestra las señales original y sintetizada entrelazadas utilizando CACIQUE, se nota que la sintetizada está montada en la original.

Segmento	Tipo de función	Número de funciones de pertenencia	Error Cuadrático Medio (ECM)
1	Gaussiana	25	2.05 x E-13
2	Gaussiana	25	2.37 x E-11
3	Gaussiana	25	3.37 x E-10
4	Gaussiana	25	9.70 x E-10
5	Gaussiana	25	3.07 x E-9
6	Gaussiana	25	6.68 x E-9
7	Gaussiana	25	8.05 x E-9
8	Gaussiana	25	1.56 x E-8
9	Gaussiana	25	1.91 x E-8
10	Gaussiana	25	2.22 x E-8
		ECM TOTAL	**7.62 x E-9**

Tabla 1: Resultados de la nota musical LA de una guitarra eléctrica

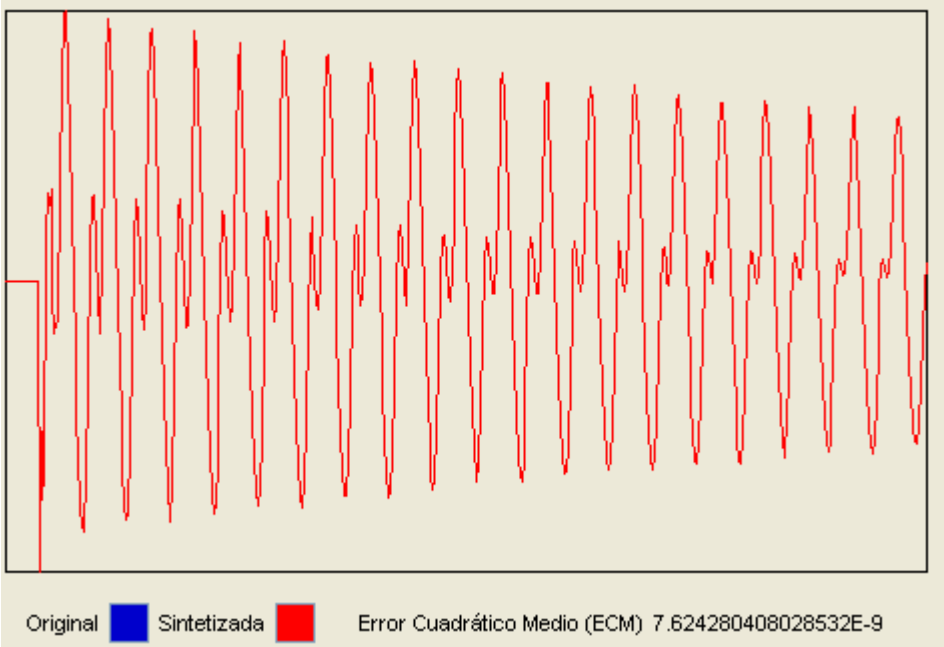


Figura 24: Señal original y sintetizada de la nota LA de una guitarra eléctrica

Prueba de la nota SOL de una guitarra eléctrica

La grafica de la señal se ilustra en la figura 25:

Figura 25: Señal de la nota musical SOL de una guitarra eléctrica

Se obtuvo los siguientes resultados mostrados en la tabla 2; observamos que por cada segmento de la función el mínimo error cuadrático fue utilizando 25 funciones Gaussiana, finalmente la figura 26 muestra las señales original y sintetizada entrelazadas utilizando CACIQUE, se nota que la sintetizada está montada en la original.

Segmento	Tipo de función	Número de funciones de pertenencia	Error Cuadrático Medio (ECM)
1	Gaussiana	25	1.41 x E-13
2	Triangular	25	3.69 x E-12
3	Gaussiana	25	1.59 x E-10
4	Gaussiana	25	6.87 x E-10
5	Gaussiana	25	2.75 x E-9
6	Gaussiana	25	3.85 x E-9
7	Gaussiana	25	5.78 x E-9
8	Gaussiana	25	1.05 x E-8
9	Gaussiana	25	1.94 x E-8
10	Gaussiana	25	2.96 x E-8
		ECM TOTAL	**7.29 x E-9**

Tabla 2: Resultados de la nota musical SOL de una guitarra eléctrica

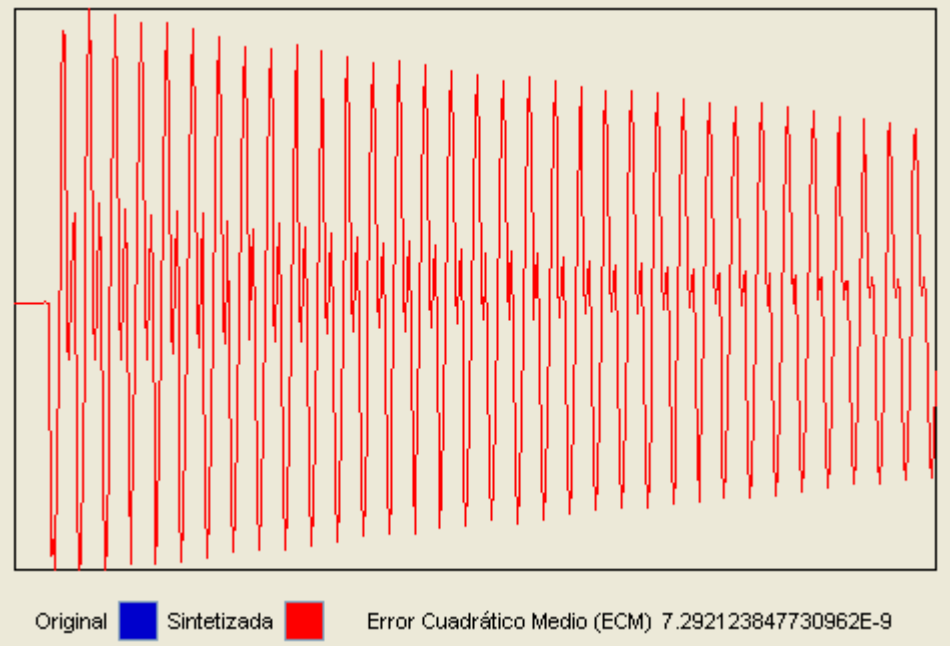


Figura 26: Señal original y sintetizada de la nota SOL de una guitarra eléctrica

Prueba de la nota SOL# de una guitarra eléctrica

La grafica de la señal se ilustra en la figura 27:

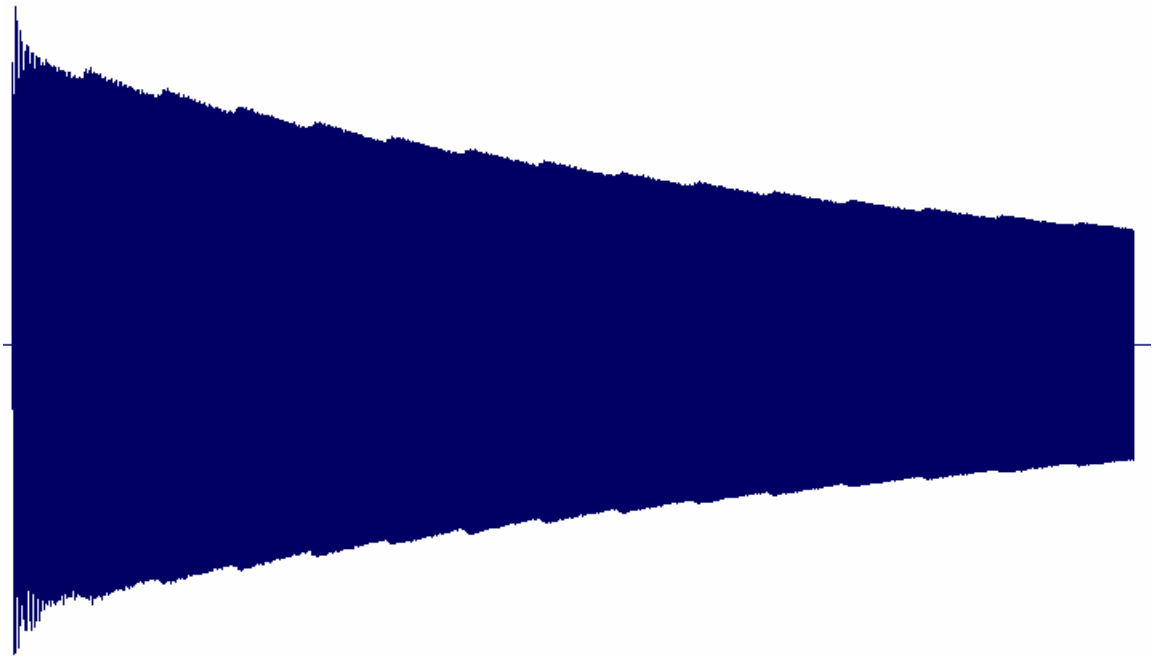


Figura 27: Señal de la nota musical SOL# de una guitarra eléctrica

Se obtuvo los siguientes resultados mostrados en la tabla 3; observamos que por cada segmento de la función el mínimo error cuadrático fue utilizando 25 funciones Gaussiana, finalmente la figura 28 muestra las señales original y sintetizada entrelazadas utilizando CACIQUE, se nota que la sintetizada está montada en la original.

Segmento	Tipo de función	Número de funciones de pertenencia	Error Cuadrático Medio (ECM)
1	Gaussiana	25	4.67 x E-14
2	Gaussiana	25	4.28 x E-11
3	Gaussiana	25	1.71 x E-10
4	Gaussiana	25	6.86 x E-10
5	Gaussiana	25	2.98 x E-9
6	Gaussiana	25	8.83 x E-9
7	Gaussiana	25	1.17 x E-8
8	Gaussiana	25	1.30 x E-8
9	Gaussiana	25	3.17 x E-8
10	Gaussiana	25	4.13 x E-8
		ECM TOTAL	**1.10 x E-8**

Tabla 3: Resultados de la nota musical SOL# de una guitarra eléctrica

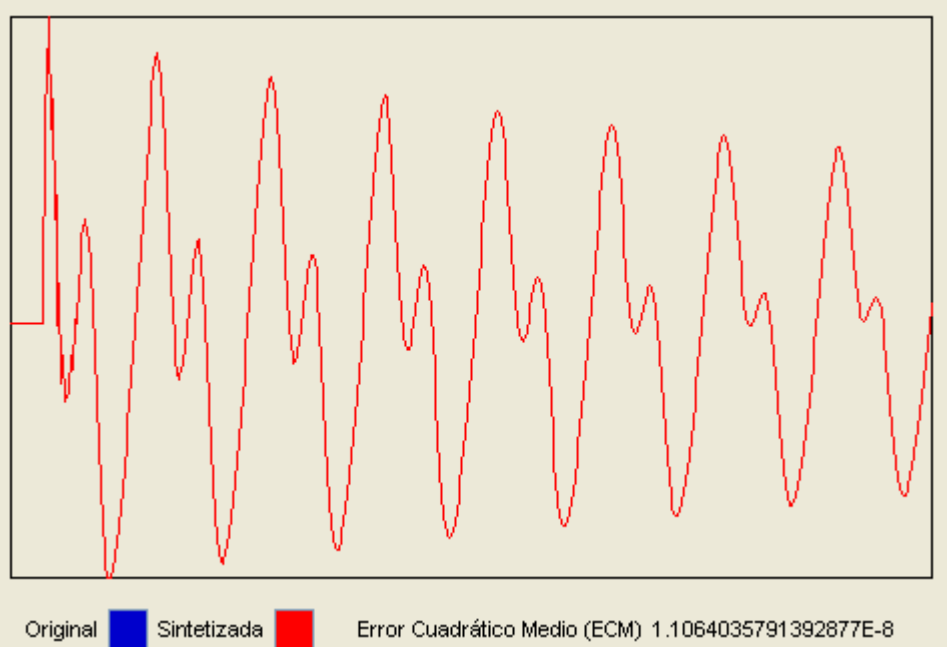


Figura 28: Señal original y sintetizada de la nota SOL# de una guitarra eléctrica

5.3 Pruebas de CACIQUE con señales de voz

Por cada segmento de señal de voz se puede obtener 58 posibles modelos ANFIS (resultado de multiplicar 2 tipos funciones de pertenencias con 29 números funciones de pertenencias), de los cuales algunos convergen y otros no; los modelos que convergen son separados para escoger el que obtuvo el mínimo error cuadrático medio, en cual resultaron los siguientes:

Prueba de la palabra Hola

La grafica de la señal se ilustra en la figura 29:

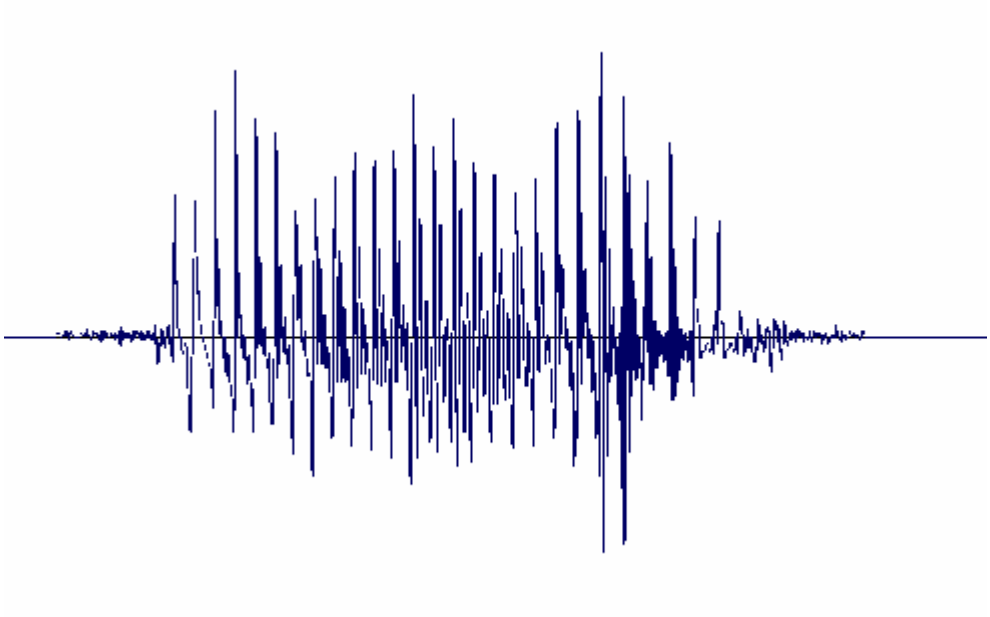


Figura 29: Señal de la palabra Hola

Se obtuvo los siguientes resultados mostrados en la tabla 4; observamos que por cada segmento de la función el mínimo error cuadrático fue utilizando 25 funciones Gaussiana, finalmente la figura 30 muestra las señales original y sintetizada entrelazadas utilizando CACIQUE, se nota que la sintetizada está montada en la original.

Segmento	Tipo de función	Número de funciones de pertenencia	Error Cuadrático Medio (ECM)
1	Gaussiana	25	2.29 x E-18
2	Gaussiana	25	6.27 x E-15
3	Gaussiana	25	3.16 x E-13
4	Gaussiana	25	8.47 x E-11
5	Gaussiana	25	2.52 x E-10
6	Gaussiana	25	5.09 x E-10
7	Gaussiana	25	2.04 x E-9
8	Gaussiana	25	1.71 x E-9
9	Gaussiana	25	2.70 x E-9
10	Gaussiana	25	2.67 x E-9
		ECM TOTAL	**9.99 x E-10**

Tabla 4: Resultados de la palabra Hola

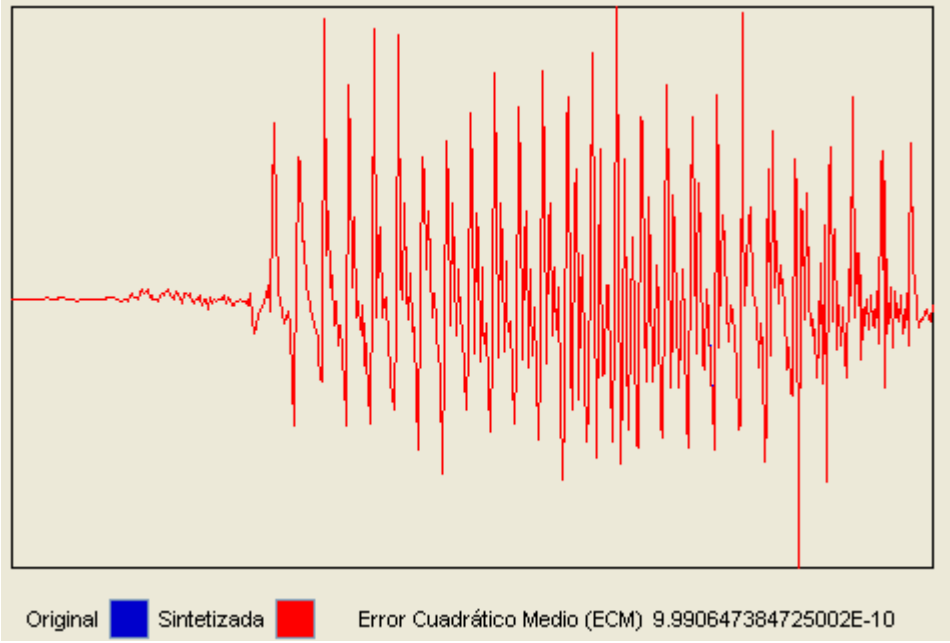


Figura 30: Señal original y sintetizada de la palabra Hola

Prueba de la palabra Manna

La grafica de la señal se ilustra en la figura 31:

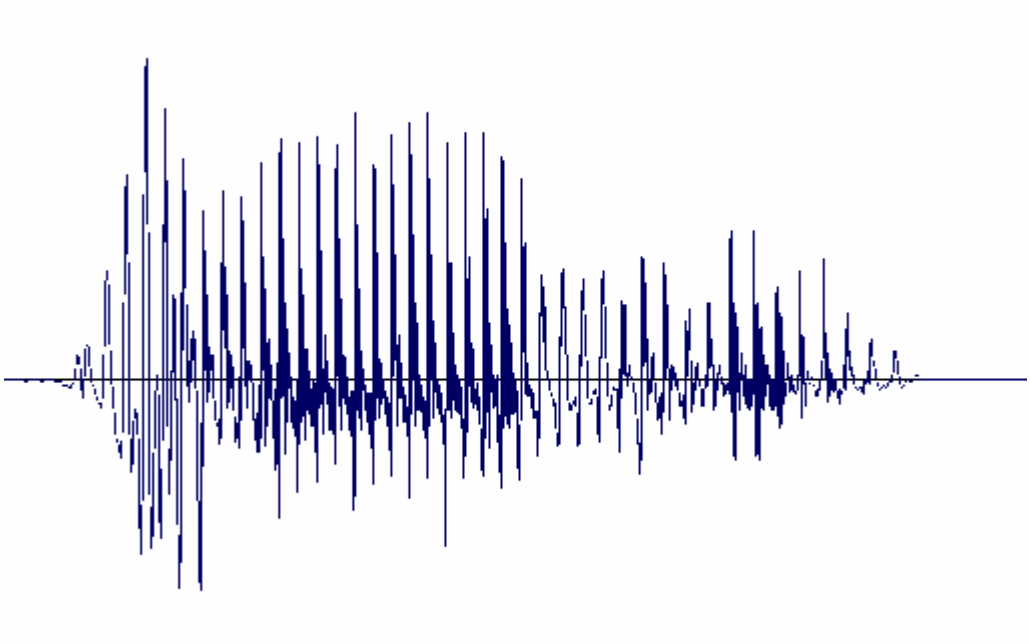


Figura 31: Señal de la palabra Manna

Se obtuvo los siguientes resultados mostrados en la tabla 5; observamos que por cada segmento de la función el mínimo error cuadrático fue utilizando 25 funciones Gaussiana, finalmente la figura 32 muestra las señales original y sintetizada entrelazadas utilizando CACIQUE, se nota que la sintetizada está montada en la original.

Segmento	Tipo de función	Número de funciones de pertenencia	Error Cuadrático Medio (ECM)
1	Gaussiana	25	2.03 x E-17
2	Gaussiana	25	3.53 x E-13
3	Gaussiana	25	9.80 x E-11
4	Gaussiana	25	9.13 x E-11
5	Gaussiana	25	1.01 x E-10
6	Gaussiana	25	4.18 x E-10
7	Gaussiana	25	7.06 x E-10
8	Gaussiana	25	3.85 x E-9
9	Gaussiana	25	1.40 x E-9
10	Gaussiana	25	2.33 x E-9
		ECM TOTAL	**9.01 x E-10**

Tabla 5: Resultados de la palabra Manna

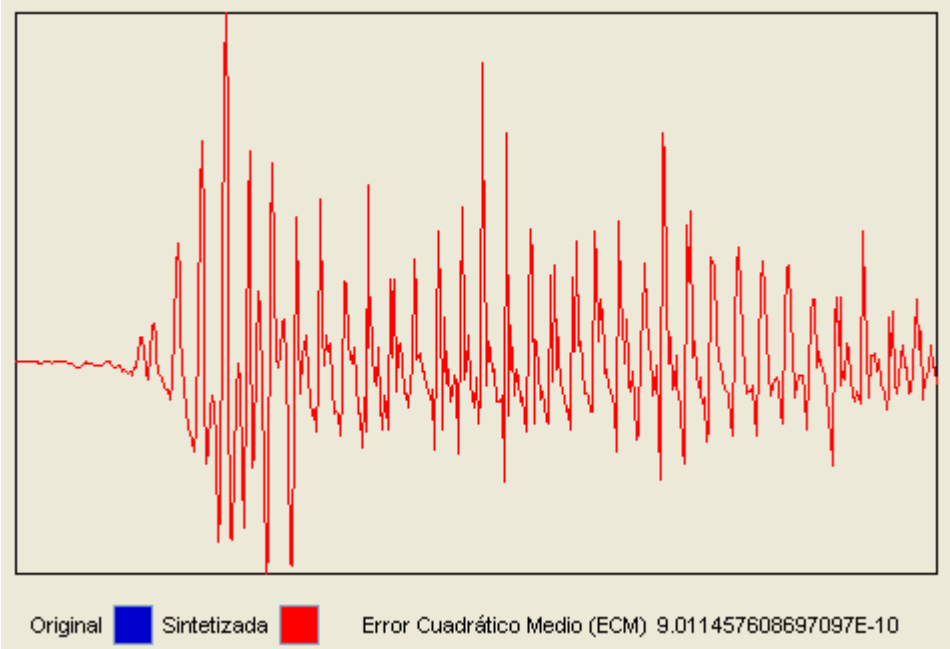


Figura 32: Señal original y sintetizada de la palabra Manna

Prueba de la palabra Politécnica

La grafica de la señal se ilustra en la figura 33:

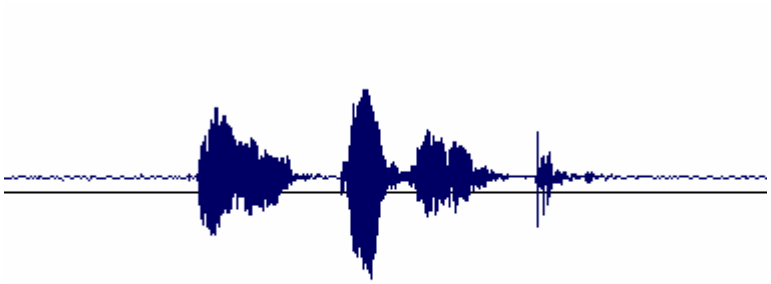


Figura 34: Señal de la palabra Politécnica

Se obtuvo los siguientes resultados mostrados en la tabla 6; observamos que por cada segmento de la función el mínimo error cuadrático fue utilizando 25 funciones Gaussiana, finalmente la figura 34 muestra las señales original y sintetizada entrelazadas utilizando CACIQUE, se nota que la sintetizada está montada en la original.

Segmento	**Tipo de función**	**Número de funciones de pertenencia**	**Error Cuadrático Medio (ECM)**
1	Gaussiana	25	1.70 x E-15
2	Gaussiana	25	2.23 x E-13
3	Gaussiana	25	1.92 x E-12
4	Gaussiana	25	8.80 x E-12
5	Gaussiana	25	4.15 x E-11
6	Gaussiana	25	9.21 x E-11
7	Gaussiana	25	1.40 x E-10
8	Gaussiana	25	2.01 x E-10

9	Gaussiana	25	4.27 x E-10
10	Gaussiana	25	5.68 x E-10
		ECM TOTAL	**1.48 x E-10**

Tabla 6: Resultados de la palabra Politécnica

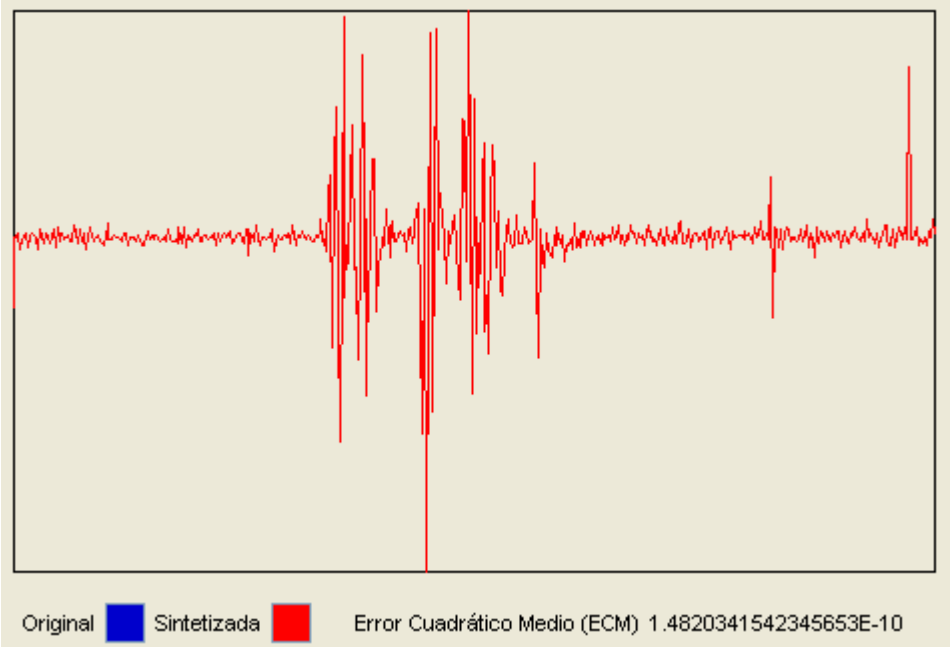


Figura 34: Señal original y sintetizada de la palabra Politécnica

5.4 Comparación de CACIQUE con otros sintetizadores

Para poder evaluar que tan bueno es nuestro sintetizador lo hemos comparado con el trabajo que realizo Axel Röbel publicado en Neural Network Modeling of Speech and Music Signals [1].

En este artículo el autor modela señales del habla con Redes Neurales para poder demostrar que la predicción de un sistema dinámico puede ser interpretada como un modelo de Red Neuronal.

Como ejemplo tomó la palabra Manna y obtuvo los siguientes resultados:

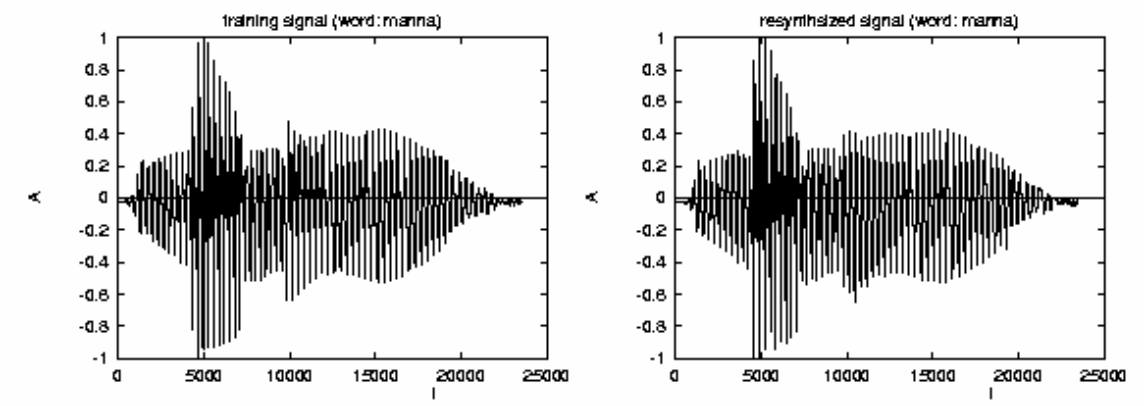


Figura 35: Señal original y sintetizada de la palabra Manna que obtuvo Axel Röbel

Como podemos observar en la figura 35 la señal es sintetizada es casi igual a la original pero difiere en algo a partir del punto x 10000; vemos que la amplitud de la original es mayor que la sintetizada.

Nuestro sintetizador obtuvo resultados más eficientes con la misma palabra Manna; podemos decir que la señal original es la misma que la sintetizada ya que su error cuadrático medio es demasiado insignificante, por eso es que en la figura 36 observamos que el color azul que representa la señal original es montada por la roja que es la sintetizada.

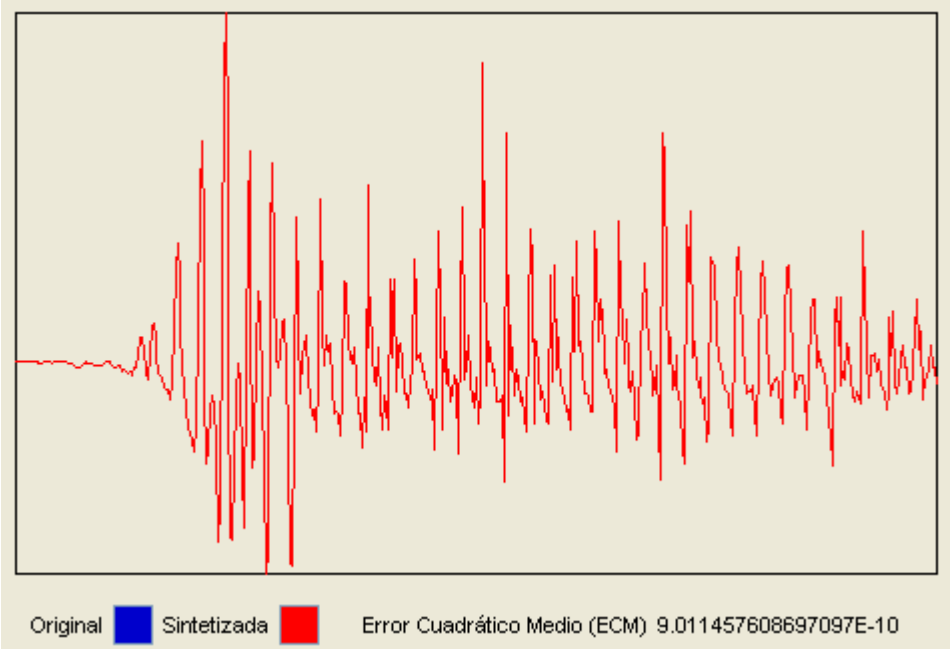


Figura 36: Señal original y sintetizada de la palabra Manna

5.5 Calculo de la métrica para medir la eficiencia del sintetizador

Para medir la eficiencia de nuestro sintetizador se utilizaron dos métricas o medidas: el error cuadrático medio y el error porcentual medio de cada señal probada

$$ECM = 1/n * \sum_{i=1}^{n} (Yi-yi)^2$$

$$EPM = 100/n * \sum_{i=1}^{n} (Yi-yi)^2/Yi$$

Donde n representa el número de muestras tomadas de las amplitudes de las señales, Yi representa la amplitud real y yi representa la amplitud sintetizada.

Señal	ECM	EPM %
Hola	9.99 x E-10	1.98 x E-11
Manna	9.01 x E-10	4.15 x E-13
Politécnica	1.48 x E-10	1.85 x E-14
LA (guitarra eléctrica)	7.62 x E-9	1.27 x E-10
SOL (guitarra eléctrica)	7.29 x E-9	5.76 x E-11
SOL# (guitarra eléctrica)	1.10 x E-8	7.17 x E-11

Tabla 7: Errores cuadráticos medios y porcentuales medios de cada una de las señales probadas

La tabla 7 nos muestra la señal junto con sus errores cuadráticos medio y porcentuales; notamos que los resultados son contundentes CACIQUE posee una calidad de sonido tanto en naturalidad como inteligilibilidad.

Conclusiones

1. CACIQUE presentó excelentes resultados, logrando naturalidad como intangibilidad, debido a la red ANFIS, la señal original es digitalizada y aprendida logrando obtener una función en el tiempo de dicha señal, por consecuente la reproducción del sonido sintetizado es casi igual a la señal original.

2. Al comparar con el método de Redes Neuronales aplicados a series de tiempos de Axel Röben [1] se observo que el ANFIS presenta un mejor desempeño en las señales de voz.

3. CACIQUE puede ser utilizado como base para crear un sintetizador formante - concatenativo de alta confiablidad, para ello se deberá implementar una base de conocimientos para guarda las funciones que permitirán concatenar sonidos

4. Se observo que siempre se necesita de 25 funciones de membrecías para lograr el aprendizaje con éxito, esto se debe a cada segmento de señal está conformado por 50 puntos cardinales, y es difícil que el algoritmo converja con pocas funciones de membrecía.

5. Una de las principales características del modelo ANFIS es la rapidez con que alcanza valores aceptables de error de entrenamiento debido al empleo del método de mínimos cuadrados para determinar los parámetros de la salida del modelo de inferencia en el paso forward del algoritmo de entrenamiento, lo que aumenta significativamente el tiempo de ejecución de este tipo de modelos. Modelos de Redes Neuronales Artificiales empleados para previsión de series de tiempo generalmente necesitan alrededor de 5000 épocas para completar su entrenamiento, debido a que solo emplean el algoritmo de *backpropagation* para actualizar los parámetros asociados a la red que almacenan el conocimiento adquirido por el modelo (pesos sinápticos).

Recomendaciones

1. No se recomienda usar el algoritmo con más de cincuenta pares cardinales ya que no podría converger debido a la cantidad de datos de entrada

2. Durante la implementación solo se detectaron inconvenientes en la codificación del algoritmo ANFIS para su convergencia, ya que en algunas ocasiones se pasaban de las 300 épocas de aprendizajes y el programa se quedaba en un lazo infinito; esto se debe a que la tasa de aprendizaje no debe ser seteada al azar, se hizo un análisis de convergencia para nuestro caso y llegamos a la conclusión de que debe ser seteada con 45 para los parámetros a de las funciones de

membrecías y 0.05 para los parámetros b de las funciones de membrecías

Anexos

Anexo 1: Requerimientos del Sistema

Los requerimientos del sistema establecen con detalle las funciones, servicios y restricciones operativas del sistema.

Requerimientos Funcionales

- Al abrir archivo de sonido; lo primero será buscar el archivo, aparecerá una ventana de búsqueda de archivo filtrado por extensiones WAV y cuya dirección inicial será "C:\"
- El nombre y ubicación del archivo Excel donde se escribirán los datos muestreados originales y sintetizados son C:\Datos.xls
- Al escribir los datos originales muestreados serán escritos en la columna A desde la fila 3 en adelante para los datos del tiempo y en la columna B desde la fila 3 en adelante para los datos de la amplitud
- Una vez concluido el proceso de aprendizaje se escribirán los datos sintetizados en la columna C desde la fila 3 en adelante.
- Si existiera un error en el proceso de aprendizaje si hubo un error indicar cuál fue, y si el proceso ha finalizado indicarlo.
- Una vez terminado el proceso de aprendizaje, si el usuario escoge la opción de graficar señales aparecerá una pantalla de gráficas y deshabilitará las opciones de la pantalla inicial a excepción del aprendizaje.

- Esta pantalla tendrá un combo indicando cual señal quiere que muestre (Ambas, Original o la Sintetizada), también tendrá dos botones uno para graficar al señal escogida en el combo y otro para volver a la pantalla inicial. Además tendrá una cuadricula que será el espacio XY para graficar las señales. La señal original será graficado de color azul y la sintetizada de rojo
- También se mostrara cual ha sido el error cuadrático medio.
- Para volver a sintetizar otros datos el sistema permitirá hacerlo a través de la opción cancelar
- Para reproducir el audio el usuario tendrá dos formas de hacerlo: En el menú y en dos botones en la pantalla de aprendizaje.
- Al cerrar el sistema o al cancelar los datos del archivo Excel deberán ser borrados.
- El sistema deberá muestrear los datos a frecuencia de 16KHz

Requerimientos No Funcionales

Los requerimientos no funcionales son restricciones de los servicios o funciones ofrecidos por el sistema. Incluyen restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requerimientos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema.

Requerimientos del Producto

- El sistema tendrá una facilidad de uso de 5 minutos de formación para aprender todas las funcionales del sistema

- El rendimiento del sistema debe ser tal que el tiempo de espera en el proceso de aprendizaje debe ser máximo de un minuto.
- La rapidez para ver los gráficos de las señales debe ser máximo de 2 seg.
- La velocidad para escuchar el audio no debe exceder los 10 seg.
- Al momento del aprendizaje se debe consumir hasta máximo la mitad de la capacidad del procesador
- La fiabilidad del sistema debe ser tal que la tasa de ocurrencia de fallos no exceda el 5%
- Cada ordenador debe de tener como mínimo 512 MB de Memoria RAM
- Cada ordenador debe tener un procesador Pentium IV como mínimo
- Cada ordenador debe tener un disco duro de 20GB mínimo
- Cada ordenador debe tener instalado Windows XP
- Tener instalado Microsoft Excel 2003 o 2007
- Tener instalado los drivers de Audio

Requerimientos de la Organización

- Para el modelo de proceso del software se usará la metodología MSF.
- Análisis, Diseño y Programación orienta a objetos.
- El lenguaje de implementación es Java
- El estándar de codificación será la establecida por el desarrollador y estará en el documento de estándar de programación Java

Anexo 2: Especificación de Casos de Usos

Nombre	1. Abrir Archivo
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Permite iniciar un nuevo proceso
Actores	Usuario del sistema
Precondiciones	El usuario debe haber iniciado el sistema.
Flujo Normal	
1. El actor selecciona Abrir Archivo	
2. Ir al caso de uso 2	
3. Ir al caso de uso 3	
4. El sistema Habilita Y Deshabilita los componentes	
Postcondiciones	
El sistema ha digitalizado los datos y escrito en el archivo Excel o enviado un mensaje de error	

Nombre	2. Buscar Archivo
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Permite buscar un archivo de sonido en el disco duro de la maquina.
Actores	Usuario del sistema
Precondiciones	Haber iniciado caso de uso 1
Flujo Normal	
1. El actor debe busca el archivo de sonido	
2. El actor selecciona el archivo	
Flujo Alternativo	

2. El actor cancela la búsqueda
Postcondiciones
El sistema ha detectado la ruta del archivo

Nombre	3. Digitalizar Señal
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Digitaliza la señal de audio del archivo de sonido y luego los escribirá en el archivo Excel
Actores	Ninguno
Precondiciones	Haber iniciado caso de uso 1 Y Terminado el caso de uso 2
Flujo Normal	
1. El sistema inicia la digitalización	
2. Ir al caso de uso 4	
Flujo Alternativo	
2. El sistema detecta un error en la digitalización de datos	
Postcondiciones	
El sistema ha conseguido digitalizar la señal	

Nombre	4. Escribir en Archivo
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Permite escribir datos de un archivo Excel
Actores	Ninguno
Precondiciones	Haber iniciado caso de uso 3,5 o el caso de uso 9
Flujo Normal	
1. El sistema escribe datos numéricos de la hoja Excel	
Flujo Alternativo	
2. El sistema detecta un error en la escritura de datos	

Postcondiciones
El sistema ha escrito datos en el Archivo

Nombre	5. Aprendizaje
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Permite iniciar un nuevo aprendizaje para sintetizar los datos digitalizados
Actores	Usuario del sistema
Precondiciones	Haber terminado el caso de uso 1
Flujo Normal	
1. El actor selecciona el tipo de función de pertenencia por cada segmento	
2. El actor selecciona cuantas funciones de pertenencias va a usar por cada segmento	
3. El actor debe seleccionar la opción de Aprendizaje por cada segmento	
4. Ir al caso de uso 6 por cada segmento	
5. El sistema verifica si todos los segmentos han terminado bien	
Flujo Alternativo	
6. El sistema Habilita Y Deshabilita los componentes	
Postcondiciones	
El sistema ha sintetizado los datos o enviado un mensajes de error	

Nombre	6. Algoritmo ANFIS
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Utiliza el modelo ANFIS para sintetizar los datos de una señal y=f(x)
Actores	Ninguno
Precondiciones	Haber iniciado caso de uso 5

Flujo Normal
1. El sistema inicia el algoritmo
Flujo Alternativo
2. El sistema detecta un error
Postcondiciones
El sistema ha conseguido sintetizar los datos

Nombre	7. Reproducir Audio
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Permite reproduce un archivo de sonido
Actores	Usuario del sistema
Precondiciones	Haber terminado el caso de uso 1
Flujo Normal	
1. El actor selecciona Reproducir Audio	
2. El sistema reproduce el audio	
Flujo Alternativo A	
2. El sistema convierte valores Double a Bytes	
3. Ir al paso 2 del flujo normal	
Flujo Alternativo B	
3. El sistema detecta un error	
Postcondiciones	
El sistema ha reproducido el audio.	

Nombre	8. Grafica de Señales
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09

Descripción	Permite ver gráficamente las señales original y sintetizada
Actores	Usuario
Precondiciones	Haber terminado el caso de uso 1
Flujo Normal	
1. El actor selecciona Grafica de Señales	
2. El sistema deshabilita los menús de la pantalla principal	
2. El actor selecciona cuales graficas desea ver	
3. El actor hace clic en ver	
Postcondiciones	
El sistema ha visualizado la grafica de las señales y el error cuadrático medio	

Nombre	9. Cancelar
Autor	Carlos Stalin Alvarado Sánchez
Fecha	02/10/09
Descripción	Permite cancelar la apertura del archivo de sonido y volver al estado inicial del sistema
Actores	Usuario
Precondiciones	Haber terminado el caso de uso 1
Flujo Normal	
1. El actor selecciona Cancelar	
2. El sistema setea valores de los componentes	
3. El sistema llama al caso de Uso 4 usando valores en blancos	
4. El sistema Habilita Y Deshabilita los componentes	
Postcondiciones	
El sistema ha vuelto a sus estado inicial	

Anexo 3: Detalle de Clases

Nombre	**CL_CACIQUE_PRINCIPAL**
Declaración	Pública
Descripción	Clase principal del software en donde está el método main; encargada de controlar las instancias de las otras clases.
Métodos	
Nombre	prHabilitar_Componentes
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que habilita y deshabilita los componentes de interfaces dado un evento
Entrada	Un char que indicará el evento realizado
Salida	Ninguna
Nombre	prAprender
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que iniciara el proceso de aprendizaje de un segmento a través de un thread
Entrada	Ninguna
Salida	Ninguna
Nombre	prGraficar
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que iniciara el proceso de graficas
Entrada	Ninguna
Salida	Ninguna
Nombre	prRepro_Orig
Tipo	Procedimiento
Declaración	Privada

Descripción	Procedimiento que iniciara el proceso de reproducción del sonido original
Entrada	Ninguna
Salida	Ninguna
Nombre	prReprodu_Audio_Sinte
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que iniciara el proceso de reproducción del sonido sintetizado
Entrada	Ninguna
Salida	Ninguna
Nombre	prCancelar
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que cerrará el archivo de audio abierto, borrara los datos del archivo Excel y dejará al sistema en su estado inicial
Entrada	Ninguna
Salida	Ninguna
Nombre	prSalir
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que cerrará el sistema
Entrada	Ninguna
Salida	Ninguna
Nombre	prAbrir
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que iniciara la apertura del archivo de audio y su muestreo
Entrada	Ninguna
Salida	Ninguna
Nombre	fbCompr_apre_ok
Tipo	Función
Declaración	Privada

Descripción	Función que indicará si todos los segmentos de aprendizaje fue completado su respectivo proceso de aprendizaje
Entrada	Ninguna
Salida	True si es verdad de lo contrario retornara false

Nombre	**CL_Aprendizaje**
Declaración	Pública
Descripción	Clase en donde controla el comportamiento del caso de uso Aprendizaje.
Atributos	
Nombre	X
Tipo	Matriz nu_datos x 1 double
Declaración	Privada
Descripción	Matriz en donde se guardaran los datos X
Nombre	Y
Tipo	Matriz nu_datos x 1 double
Declaración	Privada
Descripción	Matriz en donde se guardaran los datos Y originales
Nombre	Y_sint
Tipo	Matriz nu_datos x 1 double
Declaración	Pública
Descripción	Matriz en donde se guardaran los datos Y sintetizados
Nombre	co_func
Tipo	Entero
Declaración	Privada
Descripción	Indica el código de la función de pertenencia a usar en el aprendizaje
Nombre	nu_func
Tipo	Entero
Declaración	Privada
Descripción	Indica el número de funciones de pertenencias a usar
Nombre	nu_datos

Tipo	Entero
Declaración	Privada
Descripción	Indica el número de datos (X-Y) de entrada
Nombre	indi
Tipo	Entero
Declaración	Privada
Descripción	Indica el número de para X-Yz en donde se inicia el segmento de aprendizaje
Métodos	
Nombre	run
Tipo	Procedimiento
Declaración	Publica
Descripción	Procedimiento encargada de ejecutar el proceso de aprendizaje
Entrada	Ninguna
Salida	Ninguna

Nombre	**CL_Archivo**
Declaración	Pública
Descripción	Clase de lectura y escritura y búsqueda del archivo Excel dada la ruta del archivo
Atributos	
Nombre	ruta
Tipo	String
Declaración	Privada
Descripción	Indica la ruta del archivo Excel
Nombre	File
Tipo	File
Declaración	Publica
Descripción	Archivo de audio
Métodos	
Nombre	prReproducir

Tipo	Procedimiento
Declaración	Pública
Descripción	Procedimiento que Reproducirá el archivo de sonido original
Entrada	Ninguna
Salida	Ninguna
Nombre	prReproducir
Tipo	Procedimiento
Declaración	Pública
Descripción	Procedimiento que Reproducirá el sonido sintetizado
Entrada	Arreglo Bytes
Salida	Ninguna
Nombre	fbEscritura
Tipo	Función
Declaración	Pública
Descripción	Función que escribirá los valores en el archivo Excel dado un arreglo en una columna dada
Entrada	Matriz double n x 1,número de filas de la matriz(entero),número de columna
Salida	Retorna true se escribió correctamente y false en caso contrario
Nombre	fbBusqueda
Tipo	Función
Declaración	Publica
Descripción	Procedimiento que buscará el archivo de Sonido y la ruta del archivo la guardará en la variable ruta
Entrada	Ninguna
Salida	True si encontró, de lo contrario retornará false
Nombre	fbEscritura
Tipo	Función
Declaración	Pública
Descripción	Función que escribirá los valores en el archivo Excel dado un arreglo en una columna dada y desde una fila dada
Entrada	Matriz double n x 1,número de filas de la matriz(entero),número de

	columna,número de fila
Salida	Retorna true se escribió correctamente y false en caso contrario
Nombre	prBorrar
Tipo	Procedimiento
Declaración	Pública
Descripción	Procedimiento que borrará el archivo Excel
Entrada	Ninguna
Salida	Ninguna
Nombre	getLine
Tipo	Función
Declaración	Privada
Descripción	Función que retornará el contenido de Líneas del Archivo de Sonido
Entrada	AudioFormato del archivo del sonido
Salida	SourceDataLine

Nombre	**CL_Grafica**
Declaración	Pública
Descripción	Clase de Interfase para las grafica de la señal original y sintetizada
Atributos	
Nombre	Color
Tipo	Color
Declaración	Privada
Descripción	Indica el color el cual se van a trazar líneas
Nombre	co_senal
Tipo	Entero
Declaración	Privada
Descripción	Indica el tipo de señal a graficar

Nombre	nu_datos
Tipo	Entero
Declaración	Privada
Descripción	Indica el número de datos (X-Y) de entrada
Nombre	Ecm
Tipo	Double
Declaración	Privada
Descripción	Constante que indica el error cuadrático en la cual el algoritmo ANFIS se detuvo
Nombre	X
Tipo	Matriz nu_datos x 1 double
Declaración	Privada
Descripción	Matriz en donde se guardaran los datos X
Nombre	Y_orig
Tipo	Matriz nu_datos x 1 double
Declaración	Privada
Descripción	Matriz en donde se guardaran los datos Y originales
Nombre	Y_sint
Tipo	Matriz nu_datos x 1 double
Declaración	Pública
Descripción	Matriz en donde se guardaran los datos Y sintetizados
Métodos	
Nombre	prCarga_Panel
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que dibujará el eje bidimensional X-Y y sus escalas
Entrada	JmenuBar,JToolBar
Salida	Ninguno
Nombre	Paint
Tipo	Procedimiento
Declaración	Privada

Descripción	Función que dibujara las señales solicitadas
Entrada	Ninguno
Salida	Ninguno
Nombre	Update
Tipo	Procedimiento
Declaración	Privada
Descripción	Actualiza el grafico de la señal
Entrada	Ninguna
Salida	Ninguna
Nombre	fdMax
Tipo	Función
Declaración	Privada
Descripción	Obtiene el valor máximo de un arreglo de números reales
Entrada	Matriz double nx1, numero_datos(Entero)
Salida	Valor Máximo(Double)
Nombre	fdMin
Tipo	Función
Declaración	Privada
Descripción	Obtiene el valor mínimo de un arreglo de números reales
Entrada	Matriz double nx1, numero_datos(Entero)
Salida	Valor Mínimo(Double)

Nombre	**CL_ANFIS**
Declaración	Pública
Descripción	Clase que utilizará el modelo ANFIS para sintetizar los datos originales
Atributos	
Nombre	co_func_pert
Tipo	Entero
Declaración	Privada
Descripción	Indica el código de la función de pertenencia a usar en el aprendizaje

Nombre	nu_func_pert
Tipo	Entero
Declaración	Privada
Descripción	Indica el número de funciones de pertenencias a usar
Nombre	nu_datos
Tipo	Entero
Declaración	Publica
Descripción	Indica el número de datos (X-Y) de entrada
Nombre	ecm
Tipo	Double
Declaración	Privada
Descripción	Constante que indica el mínimo error cuadrático medio en cual el algoritmo de aprendizaje se detiene
Nombre	X
Tipo	Matriz nu_datos x 1 double
Declaración	Publica
Descripción	Matriz en donde se guardaran los datos X
Nombre	Y_orig
Tipo	Matriz nu_datos x 1 double
Declaración	Publica
Descripción	Matriz en donde se guardaran los datos Y originales
Nombre	Y_sint
Tipo	Matriz nu_datos x 1 double
Declaración	Pública
Descripción	Matriz en donde se guardaran los datos Y sintetizados
Nombre	ECM
Tipo	Double
Declaración	Pública
Descripción	Error Cuadrático Medio cuando el algoritmo termine
Nombre	Para
Tipo	Matriz nu_datos x nu_func x2 double

Declaración	Privada
Descripción	Matriz de parámetros de las funciones de membresías
Métodos	
Nombre	fdValor_Funcion
Tipo	Función
Declaración	Privada
Descripción	Función que devolverá el valor de la función de pertenencia escogida dado un valor X
Entrada	Valor X de entrada (double), Parámetro a (double), Parámetro b (double)
Salida	Valor de la función (double)
Nombre	fdValor_Derivada
Tipo	Función
Declaración	Privada
Descripción	Función que devolverá el valor de la derivada de la función de pertenencia escogida dado un valor X
Entrada	Valor X de entrada (double), Parámetro a (double), Parámetro b (double), Es_parametro_a (Boolean)
Salida	Valor de la derivada de la función de pertenencia (double)
Nombre	prTranspuesta
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que obtendrá la transpuesta de una matriz
Entrada	Matriz m x n (double)
Salida	Ninguna
Nombre	prMultiplica
Tipo	Procedimiento
Declaración	Privada
Descripción	Procedimiento que obtendrá la multiplicación de dos matrices
Entrada	MatrizA m x n (double), MatrizB n x p (double), Indicador (Char)
Salida	Ninguna
Nombre	fiInversa

Tipo	Función
Declaración	Privada
Descripción	Procedimiento que obtendrá la inversa de una matriz cuadrada
Entrada	MatrizA n x n (double)
Salida	Entero(0 -> Si la matriz se invirtió; 1-> Si hubo una indeterminación 0/0; 2-> Si hubo una división por 0)
Nombre	fbAlgoritmo_ANFIS
Tipo	Función
Declaración	Pública
Descripción	Función en la cual está el algoritmo ANFIS para el aprendizaje
Entrada	Ninguna
Salida	True si todo sale OK, false si algo fallo
Nombre	fiSigno
Tipo	Función
Declaración	Privada
Descripción	Función en la cual retornara el valor Sign(X)
Entrada	Valor X de entrada (double)
Salida	Entero

Nombre	**CL_Inicio**
Declaración	Pública
Descripción	Clase que manejará los casos de Usos: Abrir, Reproducción y Cancelar
Atributos	
Nombre	nu_datos
Tipo	Entero
Declaración	Publica
Descripción	Indica el número de datos (X-Y) de entrada
Nombre	ecm
Tipo	Double
Declaración	Publica

Descripción	Valor que indica el error cuadrático de todos los segmentos del aprendizaje
Nombre	X
Tipo	Matriz nu_datos x 1 double
Declaración	Publica
Descripción	Matriz en donde se guardaran los datos X
Nombre	Y_orig
Tipo	Matriz nu_datos x 1 double
Declaración	Publica
Descripción	Matriz en donde se guardaran los datos Y originales
Nombre	Y_sint
Tipo	Matriz nu_datos x 1 double
Declaración	Publica
Descripción	Matriz en donde se guardaran los datos Y sintetizados
Nombre	nu_muest
Tipo	Entero
Declaración	Privada
Descripción	Indica el número de muestras del archivo de sonido
Nombre	dato_voz
Tipo	Arreglo de Double
Declaración	Privada
Descripción	Arreglo que contiene las muestras del archivo de sonido
Métodos	
Nombre	fbIniciar
Tipo	Función
Declaración	Publica
Descripción	Función que realizará el caso de uso de Abrir
Entrada	Ninguno
Salida	True si no hubo error, de lo contrario retornará false
Nombre	prReproducir_Audio
Tipo	Procedimiento
Declaración	Publico

Descripción	Procedimiento que reproducirá sonido ya sea original o sintetizado
Entrada	es_orig:Indicador boolean
Salida	Ninguno
Nombre	prBorrar_Datos
Tipo	Procedimiento
Declaración	Publico
Descripción	Procedimiento que borrará los datos del Archivo Excel
Entrada	Ninguna
Salida	Ninguna
Nombre	prECM
Tipo	Procedimiento
Declaración	Publica
Descripción	Obtiene el error cuadrático medio de todo el aprendizaje
Entrada	Ninguna
Salida	Ninguna

Nombre	**CL_Convertidor**
Declaración	Pública
Descripción	Clase que encargará de convertir las muestras del sonido de Bytes a Doubles y viceversa
Atributos	
Nombre	bits
Tipo	Arreglo de Bytes
Declaración	Privada
Descripción	Arreglo que contiene las muestras del archivo de sonido
Nombre	formato
Tipo	Boolean
Declaración	Privada
Descripción	Formato del archivo si es BigEndian o littleEndian
Métodos	

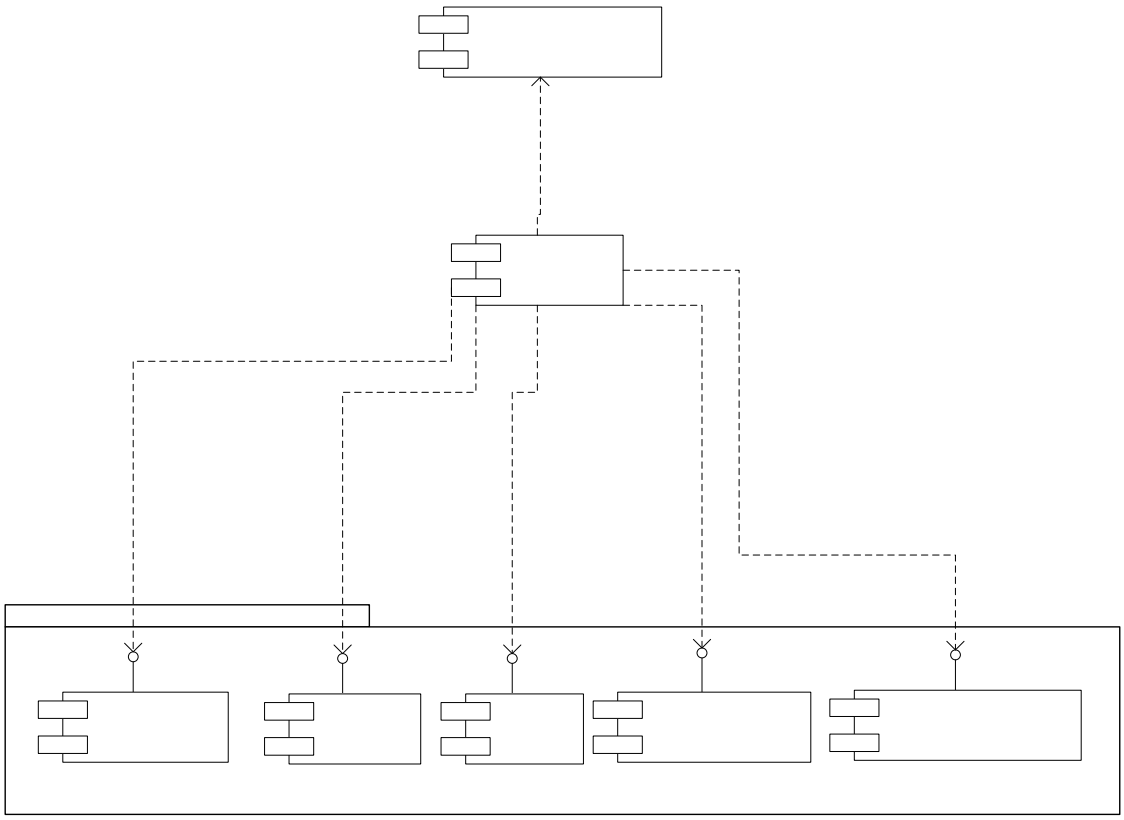
Nombre	prllenarByte
Tipo	Procedimiento
Declaración	Publica
Descripción	Procedimiento que llenara los Bytes al atributo bits
Entrada	Arreglo de Bytes
Salida	Ninguna
Nombre	fadconvertirByteADouble
Tipo	Función
Declaración	Publico
Descripción	Función que convertirá un arreglo de Bytes a Double
Entrada	Ninguna
Salida	Arreglo de Double
Nombre	fayconvertirDoubleAByte
Tipo	Función
Declaración	Publica
Descripción	Función que convertirá un arreglo de Doubles a Bytes
Entrada	Arreglo de Bytes
Salida	Arreglo de Double

Anexo 4: Diseño Detallado

Modelos de Objetos

Modelos Estáticos

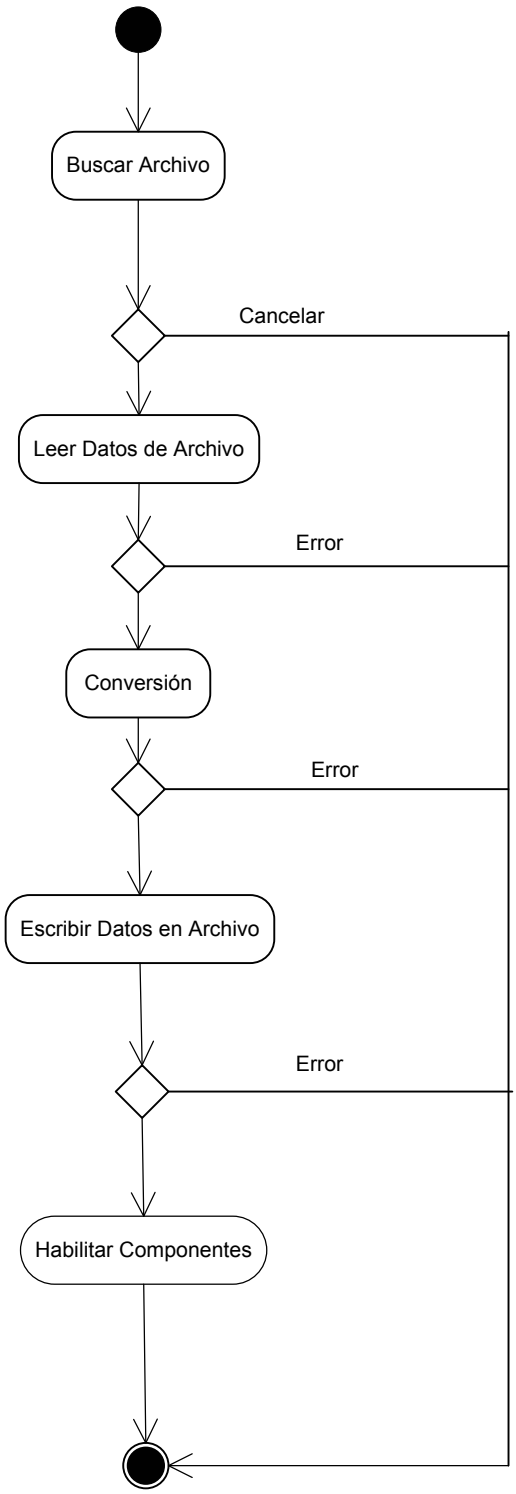
Diagrama de Componentes



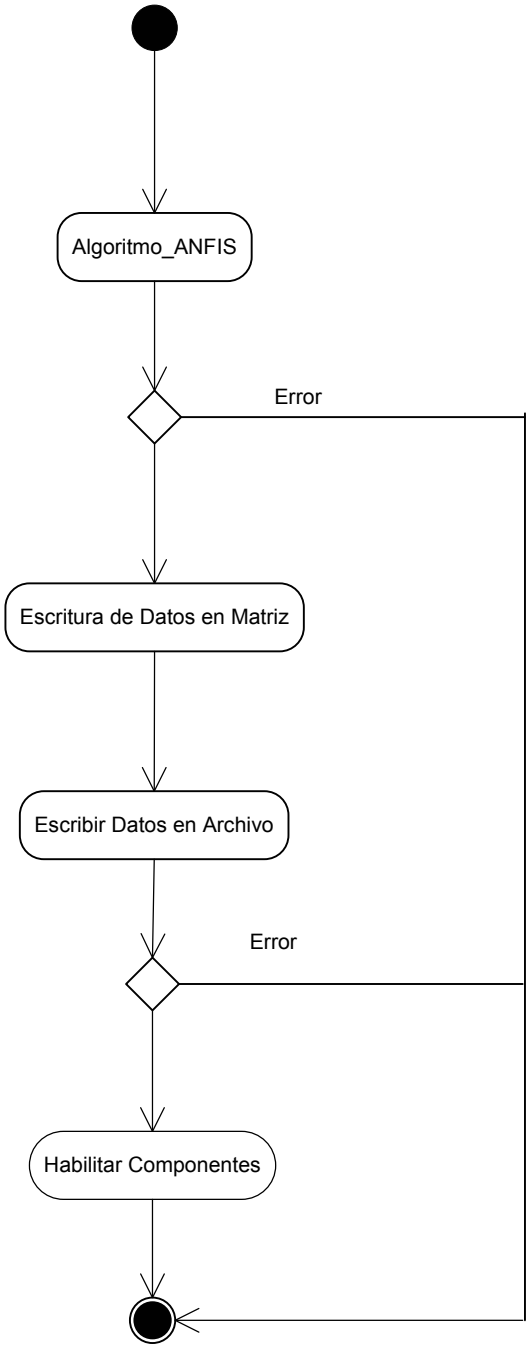
Modelos Dinámicos

Diagrama de Actividades

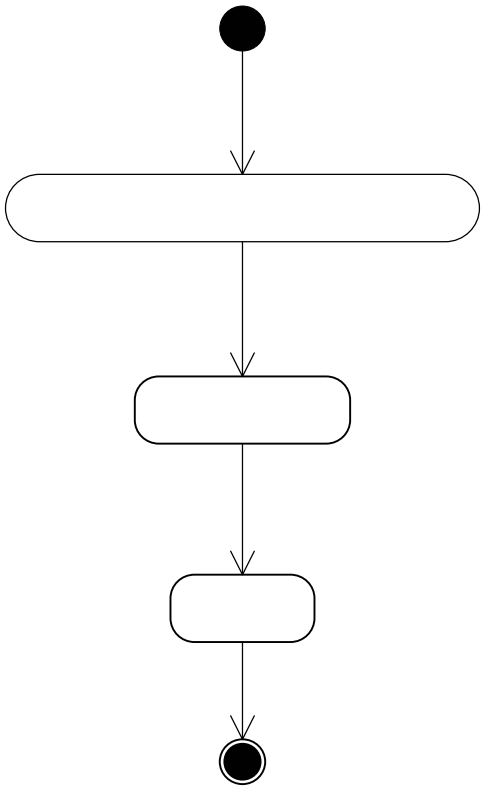
Abrir


Buscar Archivo
Cancelar
Leer Datos de Archivo
Error
Conversión
Error
Escribir Datos en Archivo
Error
Habilitar Componentes

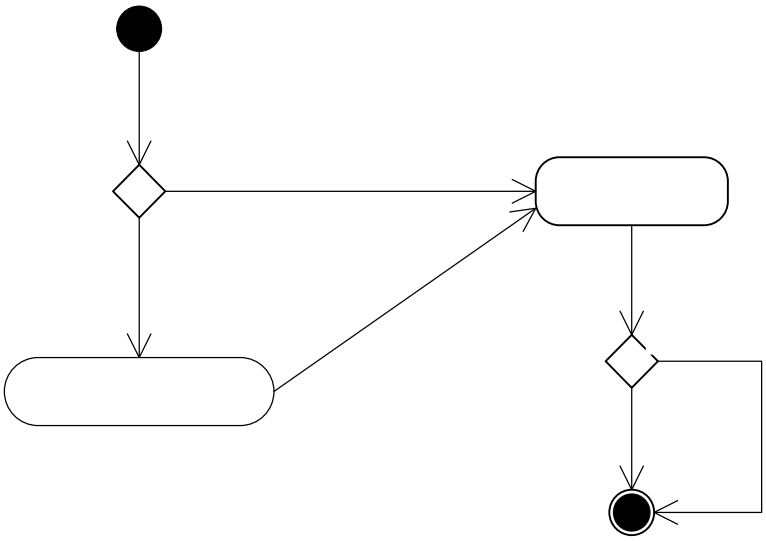
Aprendizaje



Graficas



Reproducción



Cancelar

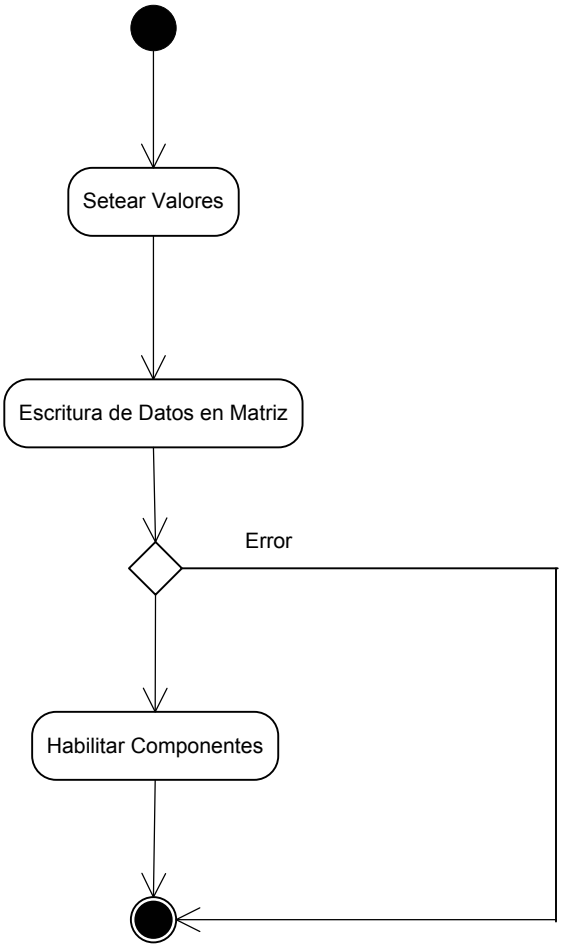
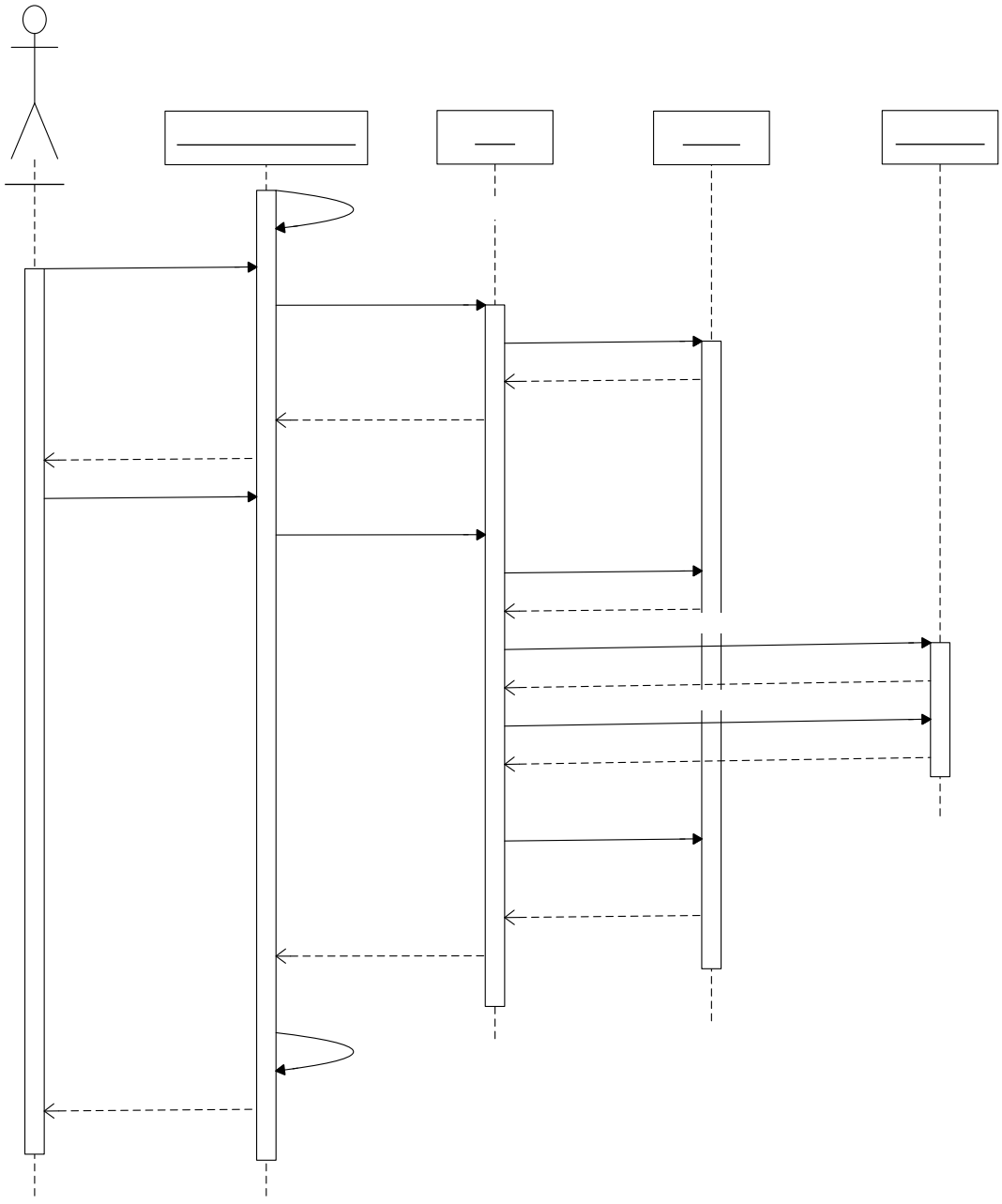


Diagrama de Secuencias

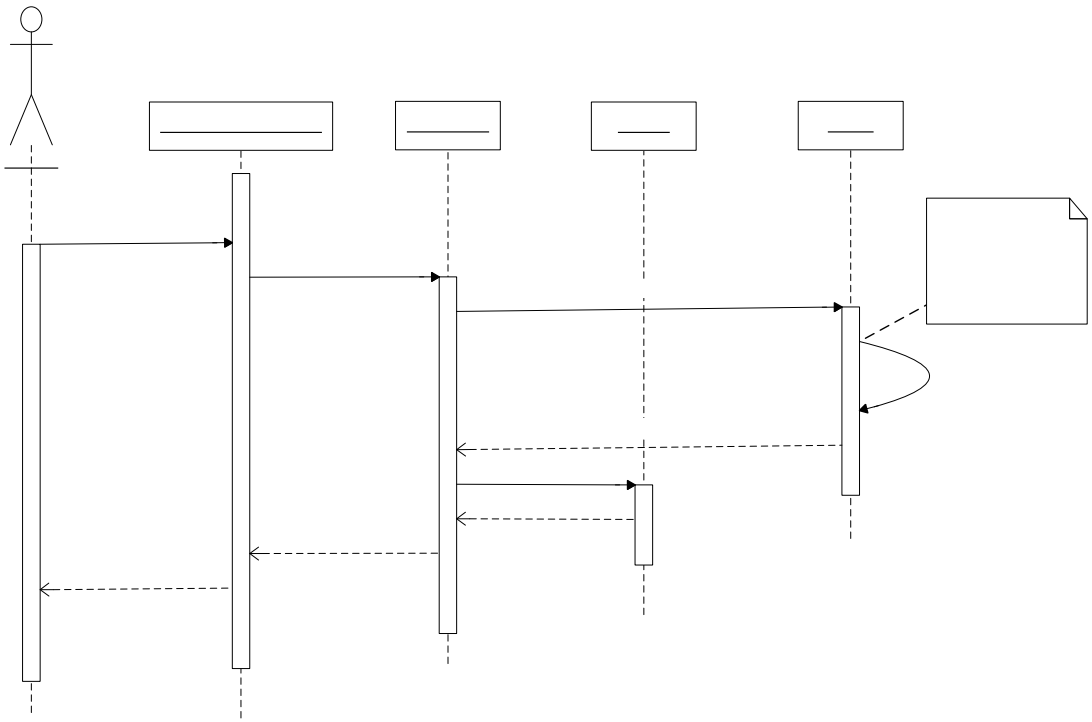
Abrir



Usuario

Abrir

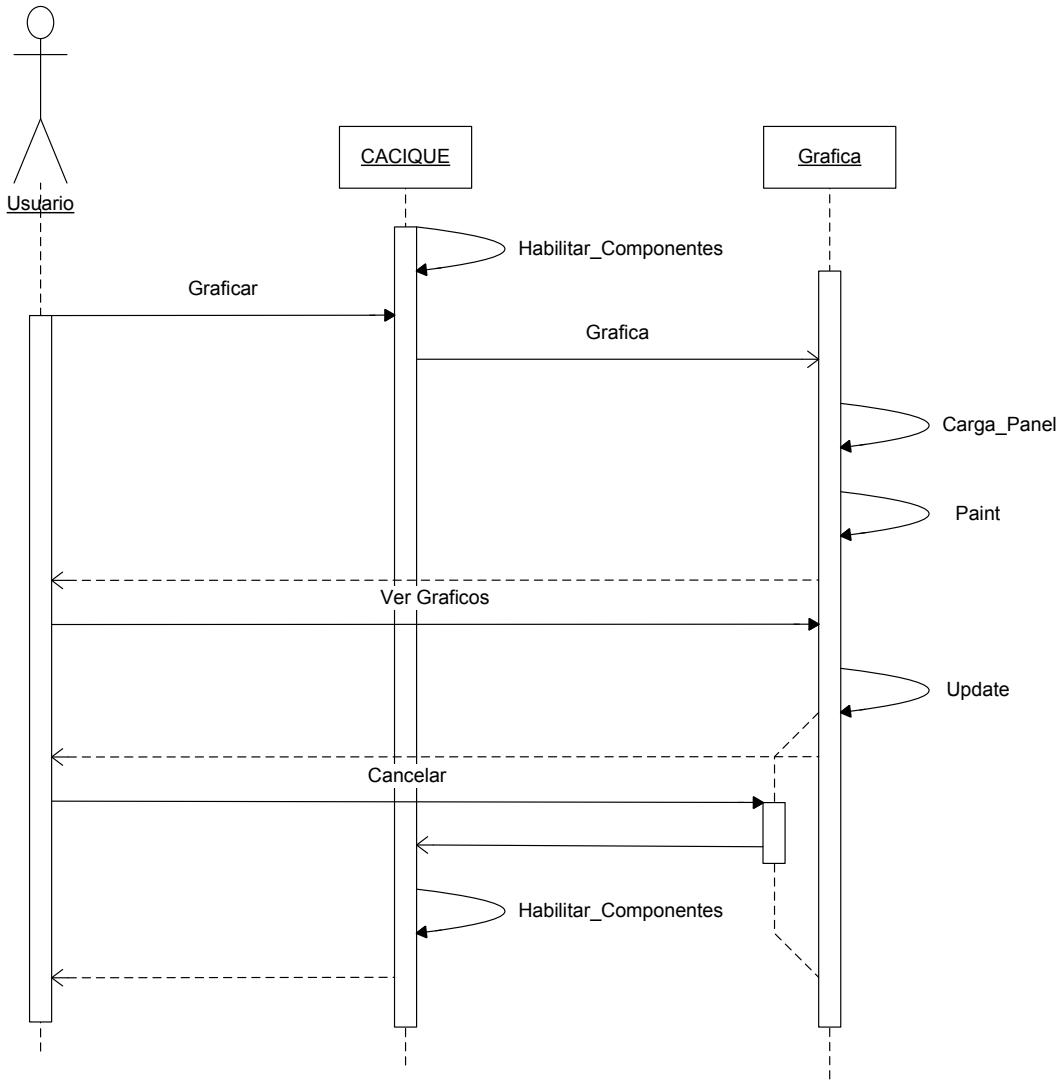
Aprendizaje



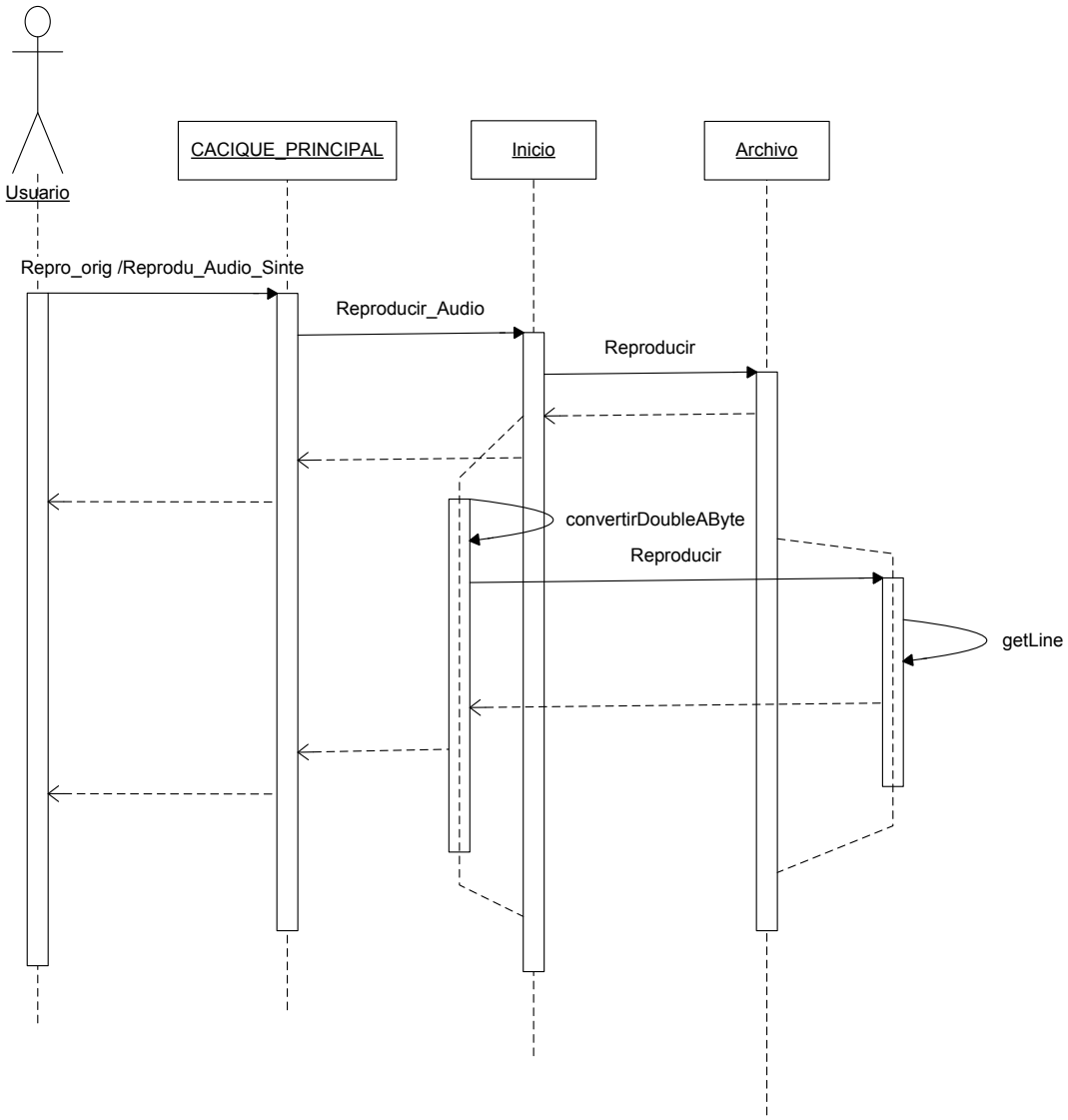
CAC

Usuario

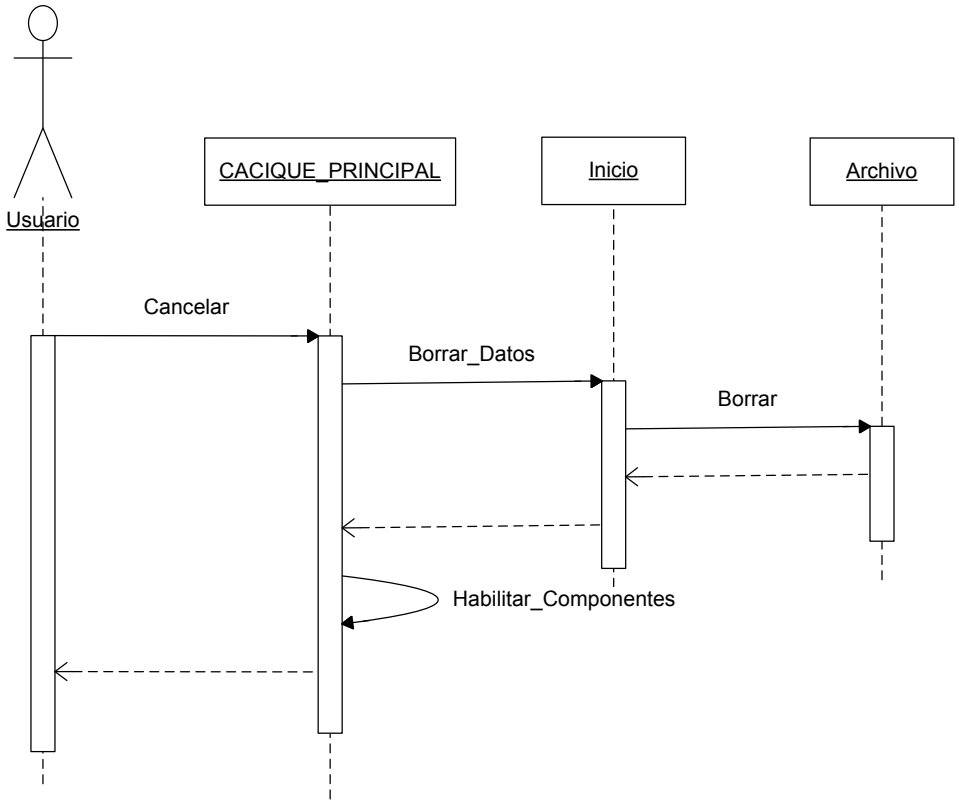
Graficas


Usuario
CACIQUE
Grafica
Habilitar_Componentes
Graficar
Grafica
Carga_Panel
Paint
Ver Graficos
Update
Cancelar
Habilitar_Componentes

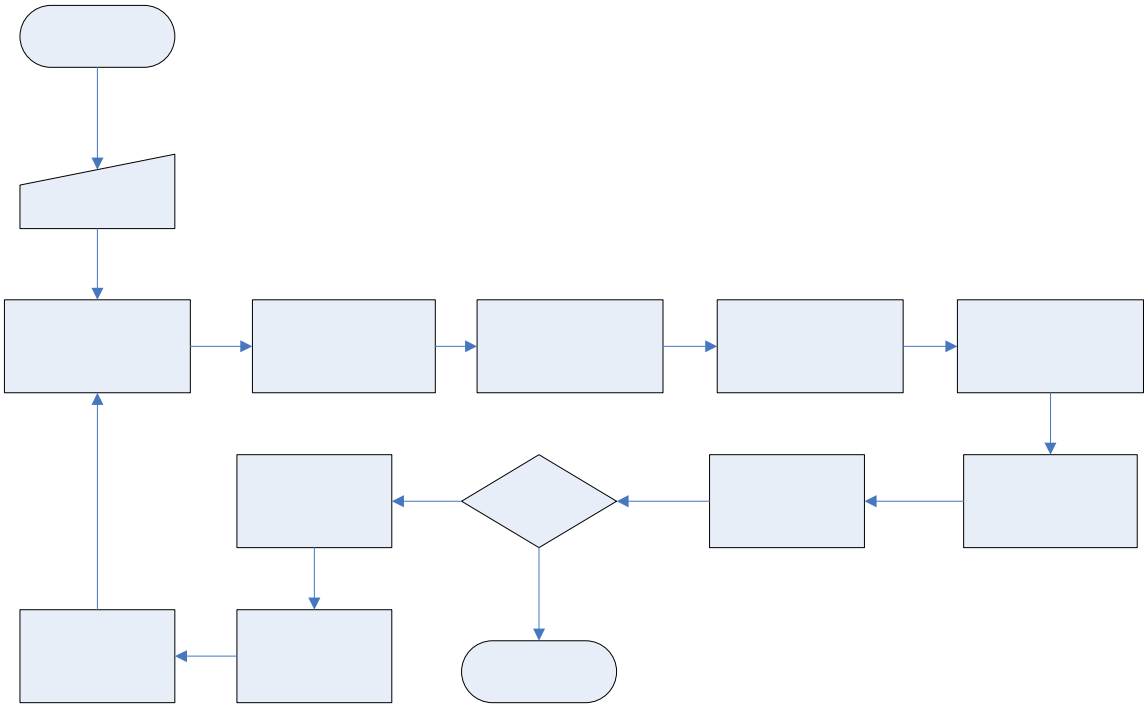
Reproducción



Cancelar


Usuario
CACIQUE_PRINCIPAL
Inicio
Archivo
Cancelar
Borrar_Datos
Borrar
Habilitar_Componentes

Detalle del Algoritmo ANFIS



Diseño de la Interfaz de Usuario

Inicio

Principios de Diseño usados

Principio	Se aplica a
Familiaridad del Usuario	Ventanas, combos box, menús, botones
Uniformidad	Ctrl+S para Salir

Interacción con el Usuario

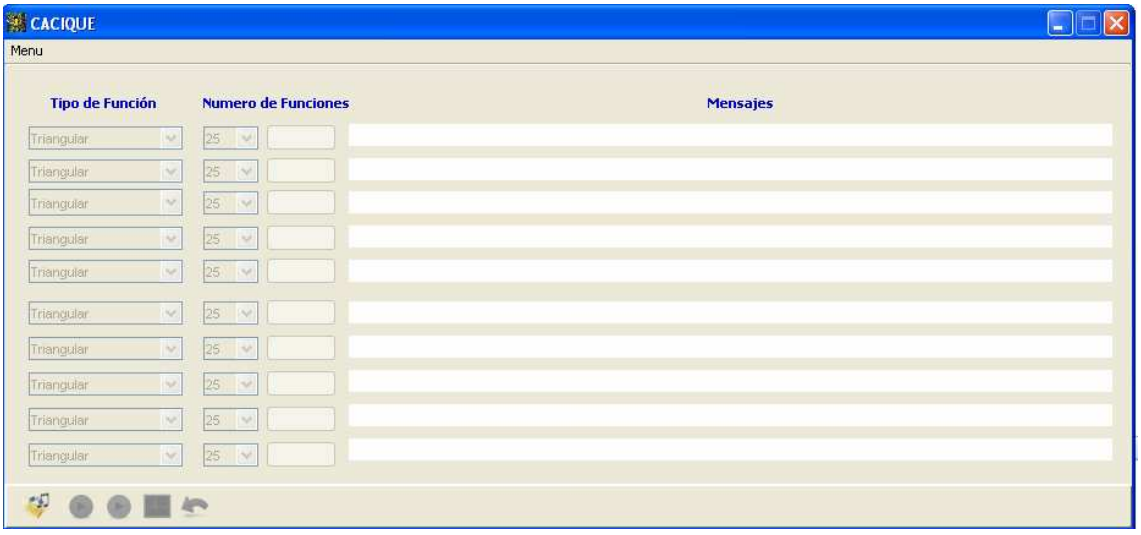
Estilo de interacción	Se aplica a

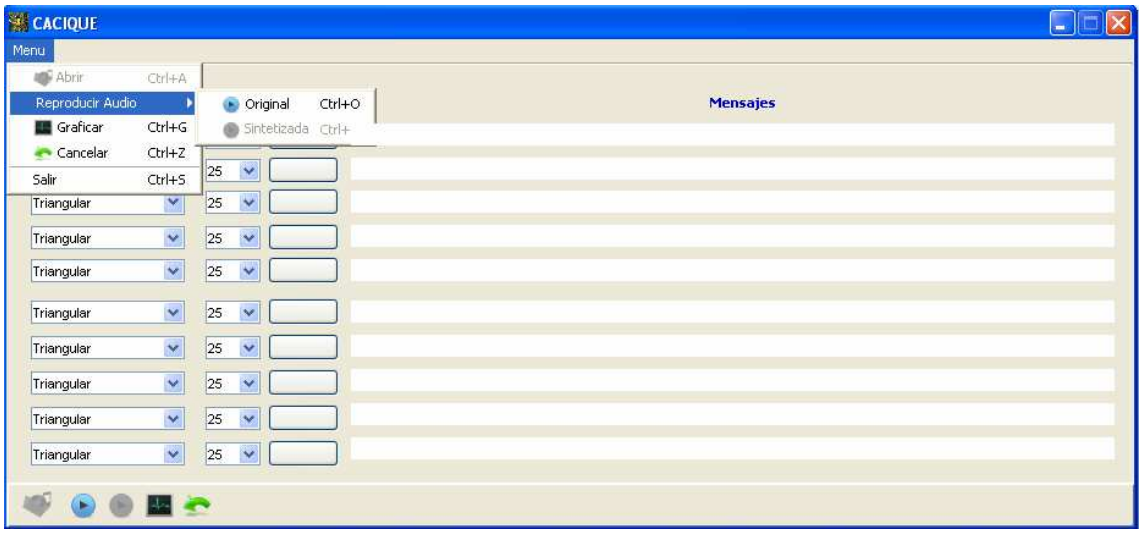
X(Frecuencia)
Y(Amplitud)

Manipulación Directa	Todas las pantallas usando el Mouse
Selección de Menús	Pantalla Principal
Lenguaje de Comandos	En la pantalla principal

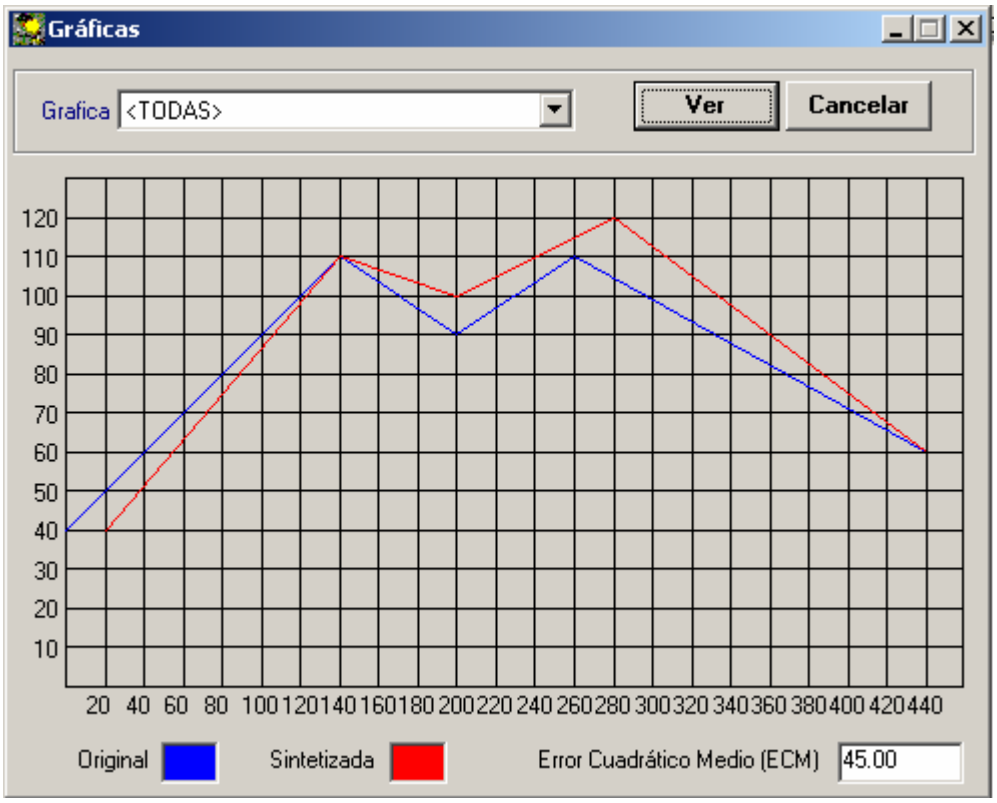
Interfaces

Pantalla Principal



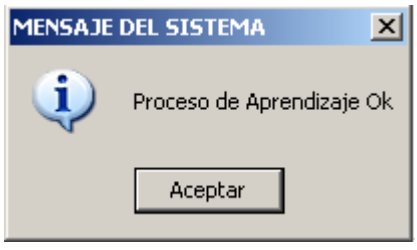


Pantalla de Graficas



Mensajes del Sistema

Mensajes de Información



Mensajes de Error

Anexo 5: Vista Preliminar de clases codificadas en java

Clase CL _ CACIQUE_PRINCIPAL

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

class CL_CACIQUE extends javax.swing.JFrame
{

   public CL_CACIQUE_PRINCIPAL()
   {……….}

   private void prSalir
   {……….}

   private void prCancelar
   {……….}

   private void prReprodu_Audio_Sinte
    {……….}

    private void prRepro_orig
   {……….}

   private void prAprender
```

```
    {……….}

    private void prGraficar
    {……….}

    private void prAbrir
     {……….}

    private boolean fbCompr_apre_ok()
     {……….}

    private void prHabilitar_Componentes(char lwcindi)
     {……….}

     public static void main(String args[])
      {……….}

}
```

Clase CL _ Archivo

```
import java.io.IOException;
import javax.sound.sampled.*;
import java.io.File;
import jxl.*;
import jxl.write.*;
import javax.swing.*;
```

```
public class CL_Archivo
{
public File pwfFile;
private String rwsruta;

public CL_Archivo()
{…….}

public void finalize() throws Throwable
{…….}

public boolean fbBusqueda ()
{……}

public boolean fbEscritura(double X[][],int lwimax,int lwicolu)
{……}

public boolean fbEscritura(double X[][],int lwimax,int lwicolu,int lwifila)
{……}

public void prBorrar()
{…….}

public void prReproducir()
{…….}
```

```
public void prReproducir(byte laydata[])
{……….}

private SourceDataLine getLine(AudioFormat lwoaf) throws
LineUnavailableException
{……….}

}
```

Clase CL _ Grafica

```
import java.awt.*;
import java.awt.event.*;
import javax.swing.*;


public class CL_Grafica extends JFrame
{
    private Color rwccolor;
    private int rwico_senal;
    private int rwinu_datos;
    private double rwdecm;
    private double rmdX[][];
    private double rmdY_orig[][];
    private double rmdY_sint[][];
```

```
public CL_Grafica(int lwinu_datos,double lmdX[][],double
lmdY_orig[][],double lmdY_sint[][],JMenuBar lwojmb,JToolBar lwojtb,double
lwdecm)
{……….}

public void finalize() throws Throwable
{……….}

private void prCarga_Panel(final JMenuBar lwojmb,final JToolBar lwojtb)
{……….}

public void update (Graphics g)
{……….}

public void paint (Graphics g)
{……….}

private double fdMax (double lmdX[][], int lwinu_datos)
{……….}

private double fdMin(double lmdX[][], int lwinu_datos)
{……….}

}
```

Clase CL _ Aprendizaje

```
import java.awt.Label;
import javax.swing.*;

public class CL_Aprendizaje extends Thread
{
public double [][]pmdY_sint;
private int rwiindi;
private int rwico_func;
private int rwinu_func;
private int rwinu_datos;
private double [][]rmdX;
private double [][]rmdY_orig;

public CL_Aprendizaje(int lwico_func,int lwinu_func,int lwinu_datos, double
[][]lmdX,double [][]lmdY_orig,int lwiindi,Label lwolabel,JButton lwoboto)
{……….}

public void finalize() throws Throwable
{……….}

public void run()
{……….}

}
```

Clase CL _ ANFIS

```
import java.awt.Label;

public class CL_ANFIS
{
    private int rwico_func;
    private int rwinu_func;
    private double rwdecm;
    private double rmdZT[][];
    private double rmdC[][];
    private double rmdD[][];
    private double rmdE[][];
    private double rmdF[][];
    private double rmdpara[][][];
    public int pwinu_datos;
    public double pwdecm;
    public double pmdX[][];
    public double pmdY_orig[][];
    public double pmdY_sint[][];

public CL_ANFIS(int lwico_func,int lwinu_func,int lwinu_datos,double
lmdX[][],double lmdY_orig[][],int lwiindi,Label lwolabel)
{……….}

public void finalize() throws Throwable
{……….}

public boolean fbAlgoritmo_ANFIS()
```

```
{……….}

private int fiInversa ()
{……….}

private void prMultiplica(double lmdX[][],double lmdY[][],char lwcind)
{……….}

private void prTranspuesta(double lmdX[][])
{……….}

private double fdValor_Derivada (double lwdvalor,double lwdpara_a,double
lwdpara_b,boolean lwbes_para_a)
{……….}

private double fdValor_Funcion(double lwdvalor,double lwdpara_a,double
lwdpara_b)
{……….}

private int fiSigno(double lwdvalor)
{……….}
}
```

Clase CL_Inicio

```
import javax.sound.sampled.*;
import javax.swing.*;
```

```
public class CL_Inicio
{
    public int pwinu_datos;
    public double pwdecm;
    public double [][] X;
    public double [][] Y_orig;
    public double [][] Y_sint;
    private int rwinu_muest;
    private double[] raddato_voz;

public CL_Inicio()
{……….}

public void finalize() throws Throwable
{……….}

public boolean fbIniciar()
{……….}

public void prReproducir_Audio(boolean lwbes_orig)
{……….}

public void prBorrar_Datos()
{……….}

public void prECM()
```

{…….}
}

Clase CL_Convertidor

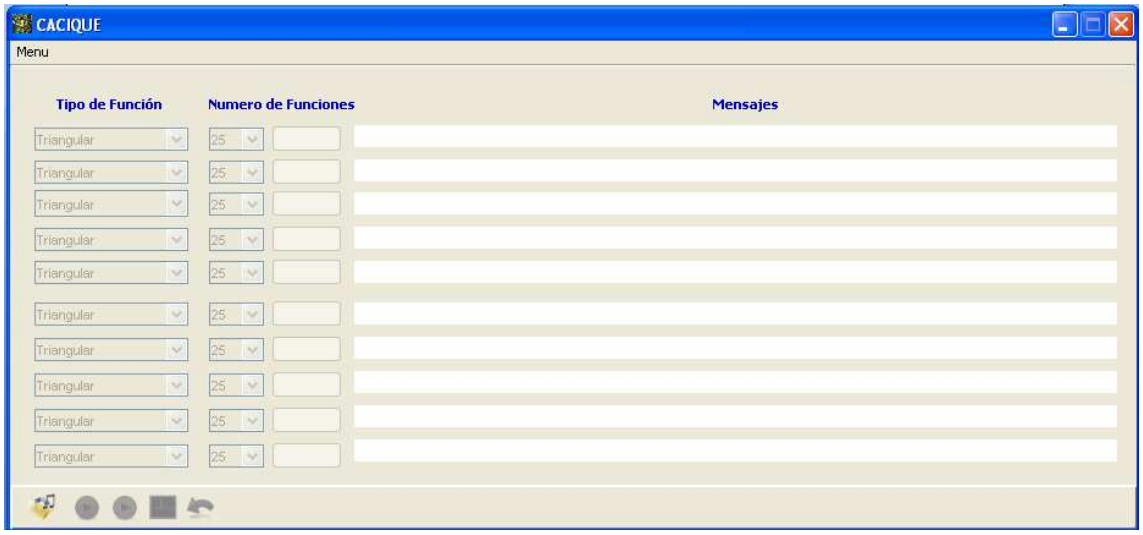
```
public class CL_Convertidor
{
   private boolean rwbformato;
   private byte[] raybits ;

public  CL_Convertidor(int lwitamano, boolean lwbformato)
{…….}

public void finalize() throws Throwable
{…….}

public void prllenarByte(byte[] laybits)
{…….}

public double[] fadconvertirByteADouble()
{…….}

public byte[] fayconvertirDoubleAByte(double []laddata)
{…….}
}
```

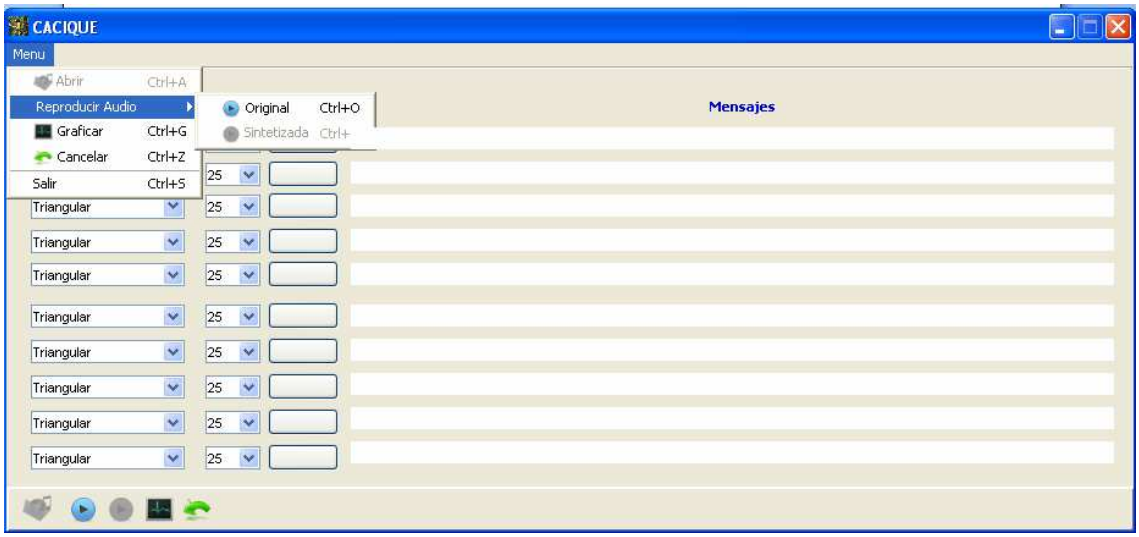
Anexo 6: Manual del Usuario

ENTRADA AL SISTEMA

MENUS DEL SISTEMA

Este sistema tendrá la siguiente barra de menú con sus correspondientes submenús:

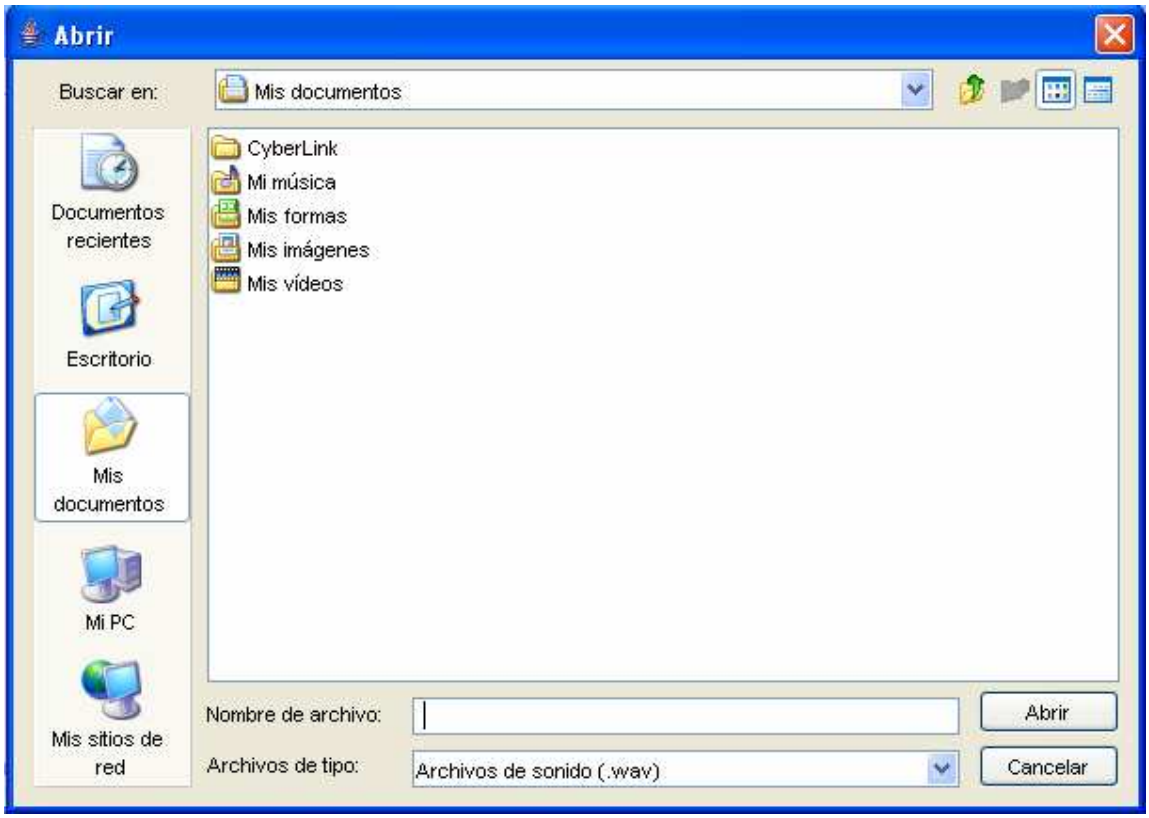




FUNCIONES DE LOS SUBMENUS

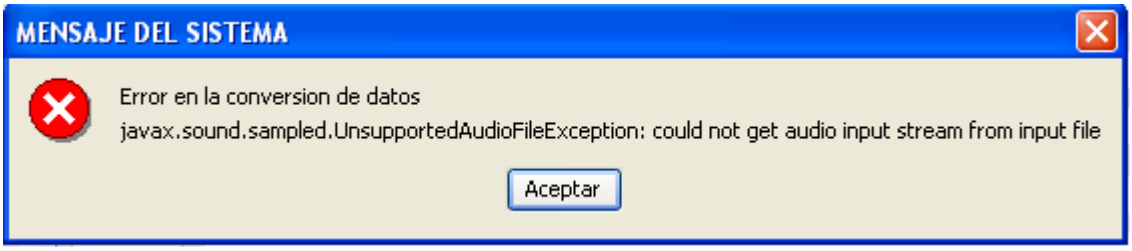
Abrir

Al ejecutar este submenú aparecerá una ventana que le permitirá abrir un archivo de audio .WAV para cargarlo al sistema



Mensajes de Errores

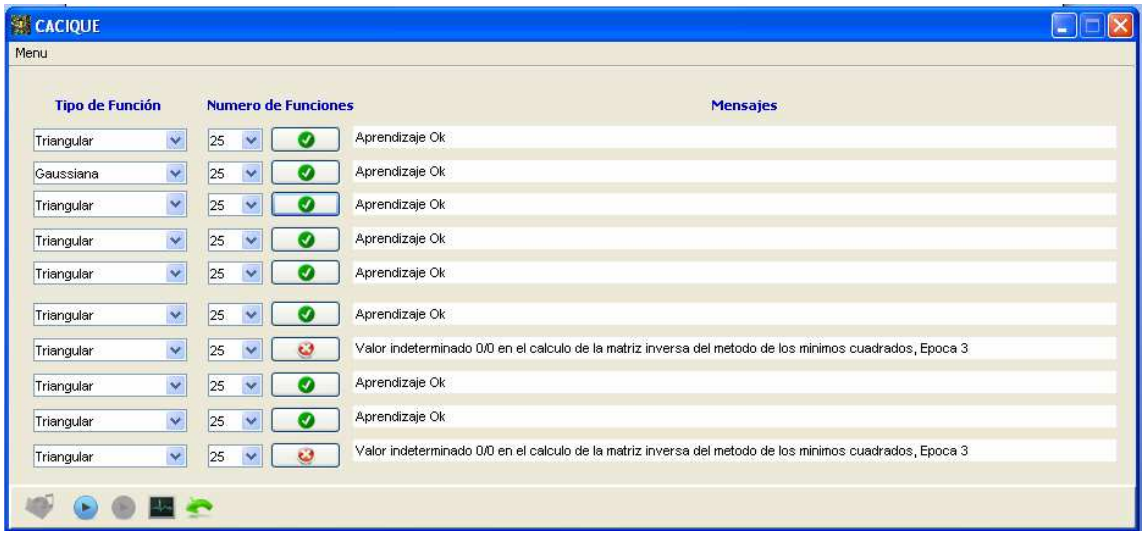
Si el archivo es inválido aparecerá el siguiente error:



Aprendizaje

Para el aprendizaje el usuario deberá seleccionar el tipo de función de pertenencia y cuántas de ellas va usar por cada segmento y luego presionar el botón.

Si hubo un error en el botón aparecerá una imagen roja de una equis y en el texto le indicara cual fue dicho error; de lo contario saldrá una imagen de un visto verde y un mensaje que el aprendizaje se realizo con éxito



Reproducción original

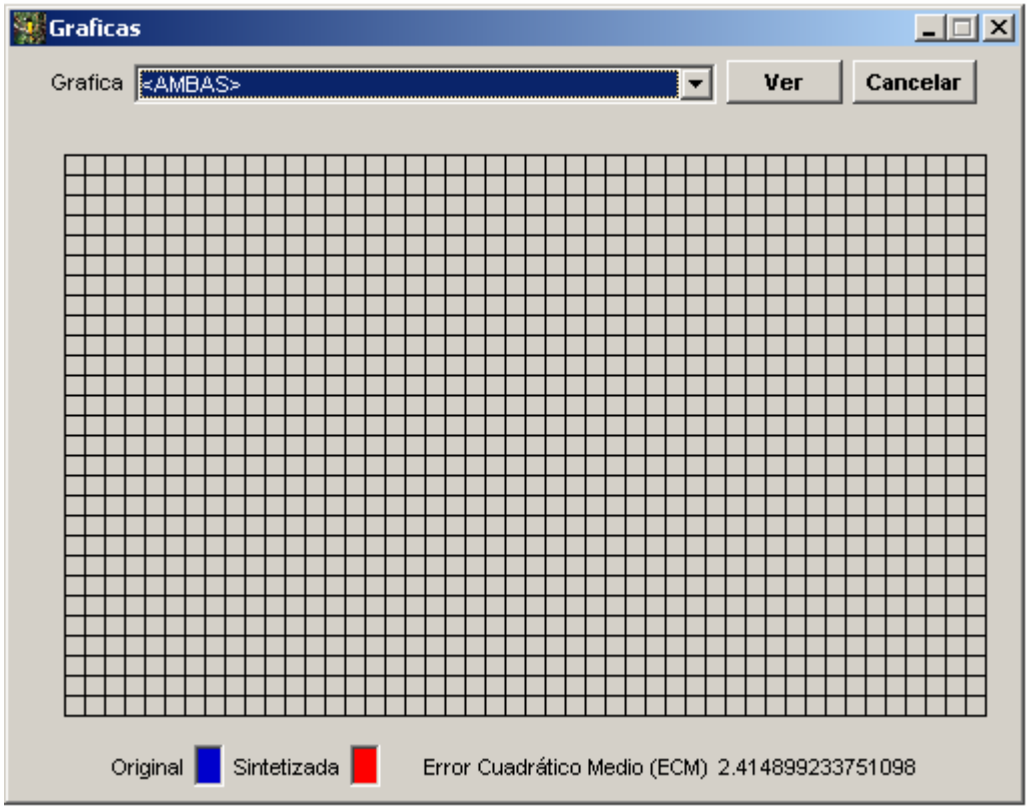
Para escuchar el audio original basta con presionar el botón azul de la barra de botones o en el menú o presionando las teclas crtl + o

Reproducción sintetizada

Para escuchar el audio sintetizado basta con presionar el botón verde de la barra de botones o en el menú o presionando las teclas crtl + t

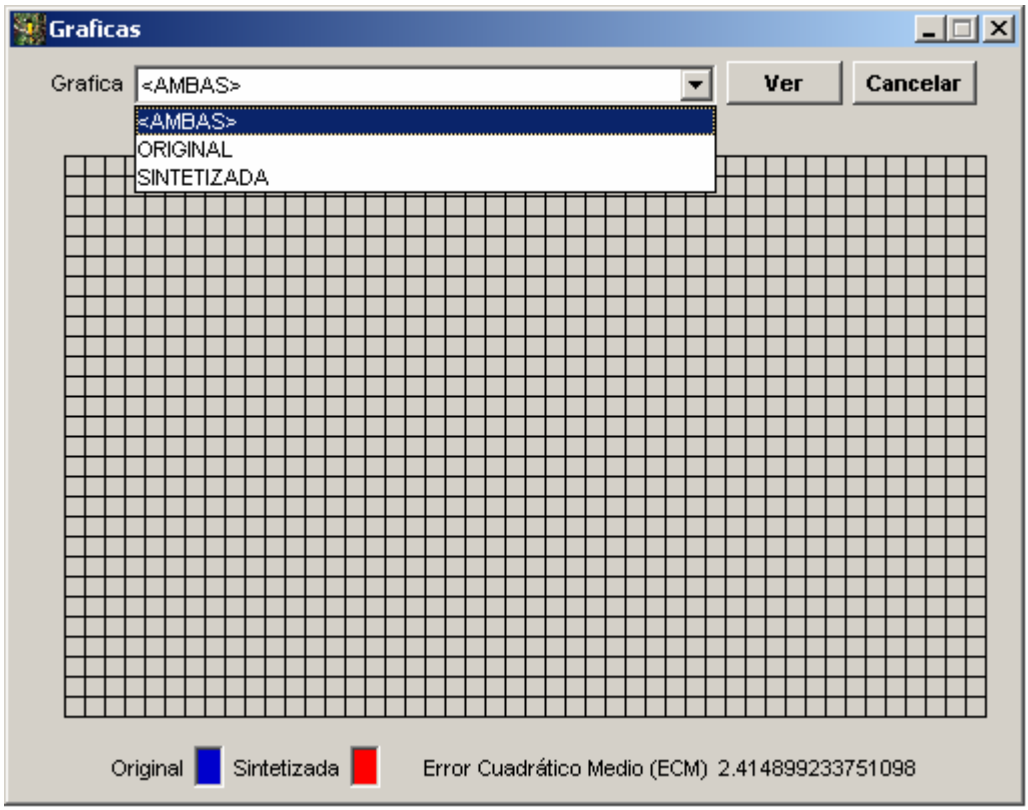
Grafica

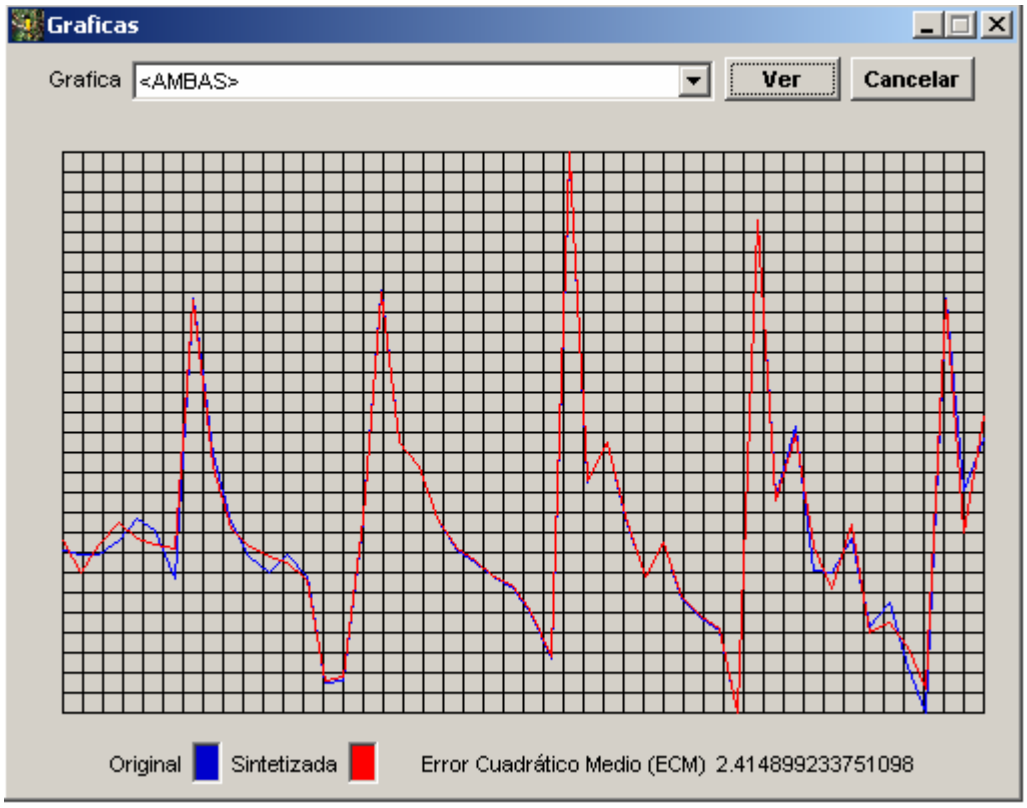
Al ejecutar este submenú le aparecerá una ventana que le pedirá cuantas unidades (eje x) quiere predecir y que le ponga un nombre al eje de las x y de las y, luego aparecerá el grafico que le indicará la predicción.



En esta pantalla se mostrará el Error Cuadrático Medio (ECM) del algoritmo, además podrá seleccionar que grafica desea ver: La original, la sintetizada o ambas y luego haga clic en Ver.

Para regresar a la pantalla anterior debe hacer clic en Cancelar


Graficas
Grafica
<AMBAS>
Ver
Cancelar
<AMBAS>
ORIGINAL
SINTETIZADA
Original
Sintetizada
Error Cuadrático Medio (ECM) 2.414899233751098



Cancelar

Al ejecutar este submenú el sistema volverá a su estado inicial, es decir con el submenú Aprendizaje habilitado.

Salir

Al ejecutar este submenú saldrá del sistema.

Bibliografía

- [1] Röbel Axel., “Neural Networks for Modeling Time Series of Musical Instruments”, Berlin, 1999
- [2] Takagi, T. y Sugeno, M., “Fuzzy identification of systems and its applications to modeling and control”, *IEEE Transactions on Systems, Man, and Cybernetics*, 1985, No. 15, pags. 116-132.
- [3] Jang, J-S. R., “Neurofuzzy Modeling: Architecture, Analyses and Applications”, Tesis de Doctorado, University of California*, Berkeley, CA*, Estados Unidos, 1992.
- [4] http://es.wikipedia.org/wiki/Sintetizador
- [5] http://es.wikipedia.org/wiki/S%C3%ADntesis_de_habla
- [6] http://ohm.utp.edu.co/neuronales/Download/Capitulo1.pdf
- [7] Choque Aspiaxu Guillermo., “Inteligencia Artificial Perspectiva y Aplicaciones”, México, 2002
- [8] Jang, J-S. R., “ANFIS: Adaptive-network-based fuzzy inference system”, *IEEE Transactions on Systems, Man, and Cybernetics*, No.23, pags. 665-685, 1993.
- [9] Riyanto, B., Febrianto, F., Machbub, C., “Adaptive-Network-Based Fuzzy Inference System for Forecasting Daily Gasoline Demand”, *Proceedings of the Sixth AEESEAP Triennial Conference*, Bali, Indonesia, 2000.