Aprendizaje de escalas musicales: Un trabajo en equipo entre dos perspectivas de la IA.

Guevara, Miguel.
Ciencias de la Informática.
Universidad Nacional Experimental de Guayana.
migue.guev@gmail.com

Resumen.

Este trabajo pretende hacer uso de dos paradigmas de la *Inteligencia Artificial* (**IA**) aplicados al campo de la música, específicamente al aprendizaje de las escalas musicales más usadas por la humanidad: la *escala natural mayor* y la *escala natural menor*. Se hace uso del entorno informático MATLAB® para simular la red neuronal encargada de clasificar una sucesión de notas (entrada o patrón), dando como resultado la tonalidad (o primera nota de la escala) que rige esta secuencia de sonidos musicales. Se hace uso del lenguaje de programación lógica PROLOG® para la generación de los patrones (conjunto de sucesiones de notas) que serán presentados a la red neuronal artificial.

Palabras clave: IA, IA tradicional, IA conexionista, RNA, MLP, perceptrón, retro-propagación, neurona, aprendizaje supervisado, patrones, sinapsis, vector, MATLAB®, PROLOG®, escala mayor, escala menor, tonalidad, escala cromática, escala heptáfona.

Abstract.

Musical Scales Learning: A teamwork between two perspectives of AI.

This work intends to make use of two paradigms of *Artificial Intelligence* (**AI**) applied to the study of music, specifically to the learning of the musical scales most used by mankind: the *major scale* and the *minor scale*. This work makes use of MATLAB®, a computer environment to simulate the neural network in charge of classifying a succession of notes (input or pattern), resulting in the tonality (or first note of the scale) that governs this sequence of musical sounds. We make use of PROLOG®, a logic programming language for the generation of patterns (set of successions of notes) that will be presented to the artificial neural network.

Keywords: AI, Symbolic AI, Connectionism AI, ANN, MLP, perceptron, supervised learning, back-propagation, neuron, patterns, synapses, vector, MATLAB®, PROLOG®, major scale, minor scale, tonality, chromatic scale, heptatonic scale.

I. Introducción.

Las *redes neuronales artificiales* (**RNA**), al igual que los sistemas expertos, son muy populares en la actualidad. Podemos encontrar gran variedad de aplicaciones de las RNA, desde el sector económico e industrial, hasta en campos como en la medicina y psicología. En particular, las RNAs del tipo Perceptrón Multicapa son muy utilizadas para la clasificación de patrones en gran variedad de problemas (difíciles de resolver mediante otros enfoques), tales como el reconocimiento de habla, reconocimiento de imágenes,

predicción de datos, problemas de optimización, etc.

A pesar de la existencia de muchas arquitecturas de RNAs, se optó por implementar el Perceptrón multicapa (MLP) para los fines de éste trabajo, dado su correcta capacidad de respuesta frente al procesamiento de patrones cuyo comportamiento es no lineal.

El objetivo de éste trabajo es simular el proceso de aprendizaje de escalas musicales (específicamente la escala natural mayor y la escala natural menor) mediante el uso de IA simbólica para la extracción de la información, la cual será procesada por una red neuronal artificial (perteneciente al enfoque de la IA conexionista).

II. Desarrollo

El proceso de aprendizaje musical suele ser un poco tedioso en sus primeras fases. Un estudiante novato de música, dada una secuencia de notas, puede no ser capaz de acertar la *tonalidad* o *escala* en la que éstas se encuentran. Toda pieza de música se rige por una tonalidad, la cual define el conjunto de notas (o sonidos) que se van a usar en un trozo de música. Una escala es una secuencia de notas, un conjunto de sonidos puestos en determinado orden. En el ámbito occidental, las escalas más usadas son la *escala mayor* y la *escala menor*. La escala mayor, tiende a crear una sensación de paz y alegría; la escala menor es algo tenebrosa, transmite tristeza. A estas escalas se les suele llamar *escalas heptatónicas o heptáfonas*, ya que están formadas por 7 notas.

La *escala cromática* es una escala formada por la sucesión de 12 sonidos (notas). Con estas 12 notas, músicos y compositores desde tiempos inmemoriales hasta la actualidad, han creado gran variedad de melodías, armonías, técnicas y géneros musicales para satisfacer todo tipo de gustos y exigencias.

Cada sonido tiene una vibración específica, y el oído humano lo percibe de un modo distinto. Se llama *tono* (**T**), a la diferencia de altura entre dos sonidos sucesivos, habiendo de por medio un sonido entre ellos. Se llama *semitono* (**S**), a la diferencia de altura entre dos sonidos sucesivos, sin que exista una nota de por medio entre ellos.

Cabe destacar que para la formación de escalas, tanto mayores como menores (y cualquier escala en general), se tomará como punto de partida la escala cromática, presentada a continuación:

Tabla I. Escala cromática.

Cifrado	Sonido
C	DO
C♯/D♭	DO sostenido/RE bemol
D	RE
D♯/E♭	RE sostenido/MI bemol
E	MI
F	FA
F♯/G♭	FA sostenido/SOL bemol
G	SOL
G♯/A♭	SOL sostenido/LA bemol
A	LA
A♯/B♭	LA sostenido/SI bemol
B	SI

Nos referiremos a cada nota o sonido por su cifrado. Para las notas sostenidas (♯) y bemoles (♭) sólo haremos referencia al cifrado con su sostenido. Fuente: Author.

La escala natural mayor se encuentra estructurada de la siguiente forma:

Figura I. Escala mayor.

Fuente: Author.

En cambio, la escala natural menor se estructura como sigue:

Figura II. Escala menor.

Fuente: Author.

Básicamente, la escala mayor se diferencia de la escala menor en la 3ra, 6ta y 7ma nota, las cuales en una pieza de música crean la sensación de alegría o tristeza previamente mencionada.

En total, podemos formar 12 escalas mayores, así como también, 12 escalas menores. Existe una gran variedad de escalas, pero para este estudio sólo se tomarán en cuenta las dos más usadas.

El problema.

Un estudiante principiante de música, cuenta frecuentemente con un profesor o con un material de apoyo, los cuales desde la perspectiva simbólica de la IA hacen el papel de base o fuente del conocimiento. El alumno extraerá cierta información de interés de la base de conocimiento, información que su cerebro percibirá como un conjunto de entradas o patrones (secuencia de sonidos musicales) y la respectiva salida deseada (escala o tonalidad en la que se encuentra dicho conjunto de notas). El aprendizaje se lleva a cabo una vez el cerebro humano realiza un proceso electroquímico en el que ajusta la sinapsis existente entre un conjunto de neuronas, para que dado un conjunto de entradas se satisfagan un conjunto de salidas. Una vez ajustados los pesos sinápticos de las neuronas involucradas, se puede decir que se ha aprendido, por lo que luego de un proceso de aprendizaje, un estudiante de música estará en la capacidad de conocer distintos tipos de escalas o tonalidades.

Se pretende simular el proceso de aprendizaje de escalas musicales mediante una red neuronal con un conjunto de entradas, donde cada entrada es una secuencia de notas musicales. La red neuronal debe ser capaz de identificar la escala o tonalidad a la que esta secuencia de notas pertenece. Se intenta simular mediante un perceptrón multicapa, como un estudiante principiante de música aprende escalas musicales. Cabe resaltar que existen muchos métodos de aprendizaje musical así como también, gran variedad de escalas musicales, pero para los límites de éste estudio, la RNA estará en capacidad de dar respuesta únicamente a las escalas naturales mayor y menor.

Obtención de los datos.

La idea fundamental de un sistema experto es almacenar conocimiento de un experto. Un experto puede ser una persona con aptitudes en algún tema; un libro o cualquier contenedor de información con sentido en un determinado contexto. Para un estudiante de música, el experto es la persona que se encarga de transmitir conocimiento. En la mayoría de los casos, tanto profesor y alumno hacen uso de un material didáctico (que a su vez fue escrito por un experto), que sirve de soporte para el conocimiento. Una vez el alumno

aprenda este conocimiento, estará en la capacidad de responder a patrones que se asemejen a los aprendidos, y podrá aprender más.

Las escalas musicales se usan como una importante guía para que el compositor cree música. Las notas pertenecientes a una determinada escala o tonalidad, sonarán bien, totalmente en contraposición a que se ejecuten notas al azar. Debido a que la escala natural mayor, así como la escala natural menor, tienen una estructura fija, simplemente basta con elegir una de las doce notas de la escala cromática (previamente descrita), aplicarle dicha estructura y se podrá obtener una secuencia de siete notas que sonarán de manera perfecta, donde la nota elegida pasará a dar nombre a la escala o tonalidad generada.

Ya que este trabajo se centra en las escalas naturales mayor y menor, para un conjunto de doce sonidos de la escala cromática se generan un total de 24 escalas; 12 escalas mayores y 12 escalas menores. Estas 24 escalas serán las entradas o patrones que recibirá la red neuronal a implementar.

Gracias a la naturaleza recursiva de la programación lógica, se eligió el lenguaje de programación PROLOG® para la generación de escalas. El proceso de generación de escalas se puede encontrar en la rutina "Escalas.pl", anexada junto a éste trabajo, y se explica a continuación:

Una escala musical tiene una estructura fija. Para obtener todas las escalas dado un conjunto de notas, basta con aplicar dicha estructura a cada nota del conjunto de notas. El nombre de la escala lo dará la primera nota, sonido generatriz o raíz. El conjunto de notas a usar estará dado por las 12 notas de la escala cromática. Estas 12 notas se insertan en una lista (estructura de datos proveída por PROLOG®), por lo que la escala cromática queda de la forma:

Escala cromática.

$$[C, C\sharp, D, D\sharp, E, F, G, G\sharp, A, A\sharp, B]$$

La estructura de la escala a generar es incluida en una lista de 7 elementos, donde cada elemento es un índice que resulta de aplicar dicha estructura a la escala cromática. Las listas correspondientes a las escalas tratadas en éste estudio son:

Estructura para la escala mayor.

$$[0, 2, 4, 5, 7, 9, 11]$$

Estructura para la escala menor.

$$[0, 2, 3, 5, 7, 8, 10]$$

*El método **generar_escala(I, [Strc], [N], [Scl?])** recorre recursivamente la lista de notas de una escala cromática **[N]**; luego, se encarga de insertar en la lista de la escala a generar **[Scl?]** los elementos de la escala cromática, cuyo índice **I** se encuentra en la lista de estructura de la escala patrón **[Strc]**.*

*El método **generar_escala([N], Type, [Scl?])** recorre recursivamente la lista de notas de la escala cromática **[N]** y ejecuta el método anterior para cada nota de **[N]**, generando 12 escalas del tipo **Type ('Mayor' o 'Menor')**, una para cada nota de la escala cromática **[N]**. Cada vez que una escala **[Scl?]** es generada, en la siguiente iteración, la tónica de esta escala generada pasa a ser la última nota de la nueva escala cromática **[N]**, dándole oportunidad a otra nota de ser la nueva tónica, y por ende creando una nueva escala (método **next_note**). De esta forma*

se asegura la creación de todas las escalas.

Para generar los 24 patrones que han de ser presentadas a la RNA, el método sin parámetros ***generar_escalas*** *hace uso del método previo, invocándolo con el parámetro* ***Type = 'Mayor'*** *para generar las 12 escalas mayores, y con* ***Type = 'Menor'*** *para generar las 12 escalas menores.*

Las redes neuronales artificiales requieren una representación de datos numérica, razón por la cual, no se le puede presentar como entradas una serie de cadenas de texto. A continuación, se presenta una tabla con la representación numérica en tipo double de los diferentes sonidos que serán clasificados por la RNA:

Tabla II. Representación numérica de las notas para ser presentadas a la RNA.

Nota	X
C	-1
C♯/D♭	-0.83
D	-0.66
D♯/E♭	-0.49
E	-0.33
F	-0.16
F♯/G♭	0.16
G	0.33
G♯/A♭	0.49
A	0.66
A♯/B♭	0.83
B	1

$(-\mathbf{1} \leq \boldsymbol{x} \leq \mathbf{1})$ *Fuente: Author.*

El método sin parámetros ***generar_data*** *genera las 24 escalas en la representación numérica requerida por la red neuronal. A continuación un ejemplo de esta representación numérica:*

Escala de C mayor:

$$[C, D, E, F, G, A, B]$$

$$[-1, -0.66, -0.33, -0.16, 0.33, 0.66, 1]$$

Escala de C menor:

$$[C, D, D\sharp, F, G, G\sharp, A\sharp]$$

$$[-1, -0.66, -0.49, -0.16, 0.33, 0.49, 0.83]$$

Algoritmo backpropagation.

Una red neuronal artificial (**RNA**) es un modelo matemático que pretende imitar el comportamiento de la vasta red de neuronas (interconectadas entre sí) que forman parte del cerebro humano. Kohonen define las RNAs como "redes interconectadas masivamente en paralelo de elementos simples (usualmente adaptativos) y con organización jerárquica, las cuales intentan interactuar con los objetos del mundo real, del mismo modo que lo hace el sistema nervioso biológico".

Una RNA consta de las siguientes características:

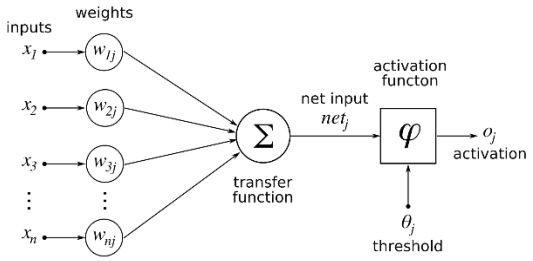
- Aprendizaje adaptativo. Capacidad de aprender basado en un entrenamiento o experiencia previa.
- Auto-organización. Capacidad de organizar y almacenar el conocimiento aprendido de manera autónoma.
- Tolerancia a fallos. Capacidad de respuesta en ambientes con

información difusa (ruido), sin que la red se sature.

- Operación en tiempo real. Uso de las ventajas del paralelismo para ejecución de cálculos.
- Facilidad de implementación. Variedad de modelos que se pueden adaptar para casos específicos.

Podemos encontrar diversos modelos de RNAs, desde redes con una única capa de neuronas, como el Perceptrón simple de Frank Rosenblatt (1958); redes con unas cuantas capas ocultas de neuronas, como el Perceptrón multicapa; hasta redes neuronales con muchas más capas de neuronas (Deep Learning), con magníficos resultados en problemas de reconocimiento de imágenes, voz, datos, etc.

Figura III. Neurona artificial.



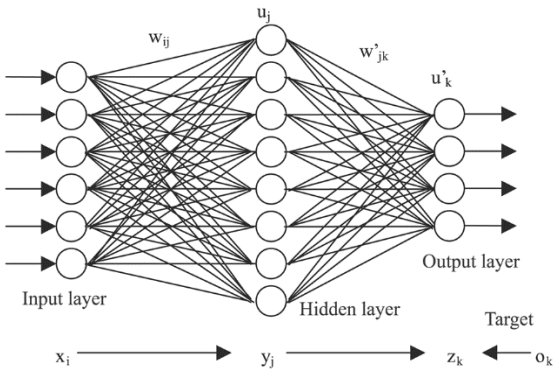
Fuente: Internet.

El modelo de red neuronal elegido para simular el aprendizaje de escalas musicales es el Perceptrón multicapa (MLP: Multi-Layer Perceptron), el cual usa el algoritmo de retro-propagación (backpropagation) para ajustar los pesos de la red. Una RNA puede aprender de distintas maneras, siendo el aprendizaje supervisado (Supervised Learning) el ideal para la arquitectura de red escogida en este estudio. Ya que se requiere que la red sea capaz de clasificar un conjunto de patrones de entrada, se debe suministrar la respectiva salida deseada, para que una vez culminado el proceso de entrenamiento se satisfagan dichos patrones.

En su forma más genérica (representada en la *figura IV*), una RNA basada en el Perceptrón multicapa se encuentra estructurada de la siguiente forma:

- Capa de entrada. Conjunto de neuronas cuya función es recibir la información relativa al problema tratado y propagarla a la capa siguiente.
- Capas ocultas. Número finito de capas de neuronas artificiales en las cuales se esconde el conocimiento representado por las conexiones existentes.
- Capa de salida. Capa de neuronas cuya función es representar el resultado del procesamiento de la RNA, dando respuesta a un conjunto de entradas específicas.

Figura IV. Perceptrón Multicapa (MLP).



Fuente: Internet.

El algoritmo de entrenamiento que utiliza el Perceptrón multicapa es el backpropagation, propuesto por Paul Werbos en 1974 (aunque también fue descubierto por Rumelhart, Hinton y

Williams en 1986). "El nombre de éste algoritmo se debe a la idea de que el error es propagado hacia atrás una vez que se calcula haciendo propagación hacia adelante" (Paletta, 2010). El algoritmo de backpropagation se explica a continuación:

Las neuronas de la capa de entrada reciben el vector de estímulo $\boldsymbol{X} = [\boldsymbol{x_0}, \boldsymbol{x_1}, \dots, \boldsymbol{x_n}]^{\boldsymbol{T}}$ (con $\boldsymbol{x_0} = \boldsymbol{1}$), el cual será ajustado por un vector de pesos sinápticos $\boldsymbol{W} = [\boldsymbol{w_0}, \boldsymbol{w_1}, \dots, \boldsymbol{w_n}]^{\boldsymbol{T}}$ (con $\boldsymbol{w_0} = \boldsymbol{bias}$) para formar la respuesta $\boldsymbol{y} = \boldsymbol{X^T W} = \boldsymbol{X W^T}$ de una neurona. La salida obtenida de la primera capa es propagada a través de las capas ocultas, hasta llegar a la capa de salida. En las capas ocultas, la respuesta $\boldsymbol{Y}$ de cada neurona es evaluada a través de una función de activación tangente sigmoidal hiperbólica, la cual es continua y derivable (importante para la ejecución de éste algoritmo). En la capa de salida, la respuesta $\boldsymbol{Y}$ de cada neurona puede ser evaluada con cualquier otro tipo de función de activación (escalón, lineal, etc).

Entonces, en la primera fase de propagación se compara el resultado obtenido en las neuronas de la capa de salida con la salida esperada, posteriormente, se calcula el valor del error para cada neurona de la salida original.

En la segunda fase (adaptación), estos errores calculados se propagan de la capa de salida hacia las capas anteriores (capas ocultas) que contribuyen directamente a la salida, donde cada neurona de cada capa oculta involucrada recibe un porcentaje del error aproximado de su participación en la salida original. Esta propagación de errores considera un gradiente local de la neurona *j* "$\boldsymbol{\delta_j}$", que contiene información del error. El gradiente es calculado de manera recursiva en el sentido de la propagación. Los pesos sinápticos de la red se modifican siguiendo la regla delta, junto con la información del gradiente local de cada neurona, expresado por la siguiente ecuación:

$$\frac{\boldsymbol{\partial \varepsilon(n)}}{\boldsymbol{\partial w_{ji}(n)}} = -\boldsymbol{e_j(n)\varphi'_j(V_j(n))y_i(n)}$$

$$\boldsymbol{\delta_j(n)} = \boldsymbol{e_j(n)\varphi'_j(V_j(n))}$$

Quedando el factor de corrección de la siguiente forma:

$$\boldsymbol{\Delta w_{ji}(n)} = \boldsymbol{\eta \delta_j(n) y_i(n)}$$

Las reglas para actualizar los pesos sinápticos de la red se muestran a continuación:

Para los pesos sinápticos de la capa de salida:

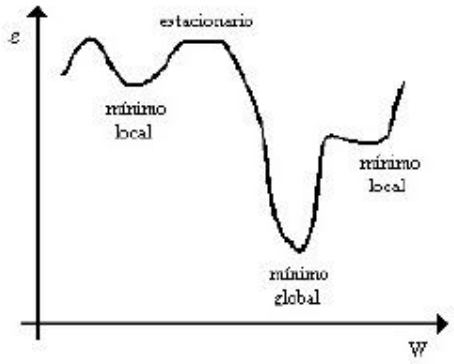
$$\boldsymbol{w^s_{jk}(t+1)} = \boldsymbol{w^s_{jk}(t)} + \boldsymbol{\Delta w^s_{jk}(t+1)}$$

Para los pesos sinápticos de las capas ocultas:

$$\boldsymbol{w^h_{ij}(t+1)} = \boldsymbol{w^h_{ij}(t)} + \boldsymbol{\Delta w^h_{ij}(t+1)}$$

A lo largo del proceso de aprendizaje, la función de error describe una curva en la que se pueden hallar varios mínimos locales, un mínimo global y puntos estacionarios (ver *figura V*). Lo ideal es encontrar el mínimo global, pero al usar un algoritmo de descenso de gradiente (como el backpropagation), existe la posibilidad de caer en un mínimo local o punto estacionario, lo que dificultaría la convergencia del algoritmo. En muchos casos no es necesario conseguir un mínimo global, ya que se puede llegar a una situación en la cual las salidas de la red sean aceptables.

Figura V. Curva de error en función de los pesos de las conexiones.



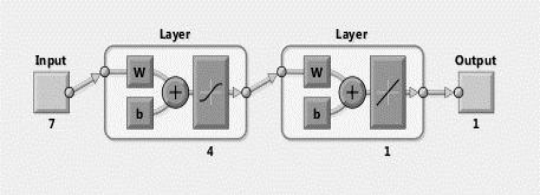
Fuente: Paletta, M. Inteligencia Artificial Básica.

Arquitectura y simulación de la RNA.

La escala natural mayor y la escala natural menor, son escalas heptáfonas, es decir, son una sucesión de 7 notas o sonidos musicales. Es por ello, que la red neuronal a implementar tendrá un total de 7 neuronas en la capa de entrada (Input = 7), ya que se quiere saber a qué escala pertenece una sucesión de 7 notas. Se provee a la red de un total de 24 posibles patrones (12 escalas mayores y 12 escalas menores), haciendo una matriz de 7x24 (7 neuronas, 24 patrones) de tipo *double*, donde cada valor de la entrada se encontrará en el rango $-1 \leq x \leq 1$ (ver ***tabla II*** para entender la representación numérica de las notas).

No hay un criterio estándar para calcular el número de neuronas que debe tener la capa oculta, aunque existe un criterio de selección muy usado llamado *criterio de la pirámide*. Para fines de éste estudio, a la capa oculta se le dio un total de 4 neuronas, basándose en problemas parecidos, como el procesamiento de patrones cuyo número de entradas, patrones presentados y neuronas de salida se asemejan al problema de estudio, dando muy buenos resultados en un corto período de entrenamiento. Para las neuronas de la mencionada capa oculta, se utilizó la función tangente sigmoidal hiperbólica (*tansig*) $\boldsymbol{f(x)} = \frac{e^x - e^{-x}}{e^x + e^{-x}}$. Ésta función *tansig* fue elegida dado que es una función derivable y es más exacta que *logsig* (función tangencial sigmoidal), y para los requerimientos de éste trabajo la exactitud en la salida es necesaria.

Figura VI. ***Arquitectura de la RNA.***



Fuente: Author.

La respuesta que se quiere obtener de la red neuronal, una vez culminado el proceso de entrenamiento, es la escala a la que pertenece una secuencia de 7 notas, por lo que la capa de salida de la RNA contará con una única neurona de salida. Ésta neurona de salida tendrá una función *lineal* $\boldsymbol{f(x) = x}$ como función de activación, ya que se necesita la salida tal y como la envía la capa oculta, para así poder representar el nombre de la escala resultante. Por ende, si se tiene un total de 24 escalas a clasificar, se necesitarán 24 salidas, ya que cada salida corresponderá al nombre de la escala (sea mayor o menor) a la que pertenece la entrada dada.

Para hacer referencia a la escala natural mayor, se escribirá en mayúsculas el cifrado de la nota raíz (nota que da el nombre de la escala), como sigue:

Tabla III. ***Representación del cifrado de la escala natural mayor.***

Cifrado	Escala
C	DO mayor
C♯/D♭	DO sostenido mayor /RE bemol mayor
D	RE mayor
D♯/E♭	RE sostenido mayor /MI bemol mayor
E	MI mayor
F	FA mayor
F♯/G♭	FA sostenido mayor /SOL bemol mayor
G	SOL menor
G♯/A♭	SOL sostenido mayor/LA bemol mayor
A	LA mayor
A♯/B♭	LA sostenido mayor /SI bemol mayor
B	SI mayor

Fuente: Author.

Para hacer referencia a la escala natural menor, se escribirá el cifrado de la nota raíz en mayúsculas, seguido de la letra *m* en minúscula, como sigue:

Tabla IV. ***Representación del cifrado de la escala natural menor.***

Cifrado	Escala
C*m*	DO menor
C♯*m*/D♭*m*	DO sostenido menor/RE bemol menor
D*m*	RE menor
D♯*m*/E♭*m*	RE sostenido menor/MI bemol menor
E*m*	MI menor
F*m*	FA menor
F♯*m*/G♭*m*	FA sostenido menor/SOL bemol menor
G*m*	SOL menor
G♯*m*/A♭*m*	SOL sostenido menor/LA bemol menor
A*m*	LA menor
A♯*m*/B♭*m*	LA sostenido menor/SI bemol menor
B*m*	SI menor

Fuente: Author.

A continuación, se muestra la representación numérica en tipo *double* de los nombres de las escalas deseadas:

Tabla V. ***Representación numérica del cifrado de la escala natural mayor, para ser presentado como salida deseada de la RNA.***

Escala mayor	X
C	-1
C♯/D♭	-0.83
D	-0.66
D♯/E♭	-0.49
E	-0.33
F	-0.16
F♯/G♭	0.16
G	0.33
G♯/A♭	0.49
A	0.66
A♯/B♭	0.83
B	1

$(-\mathbf{1} \leq \boldsymbol{x} \leq \mathbf{1})$ *Fuente: Author*.

Tabla VI. ***Representación numérica del cifrado de la escala natural menor, para ser presentado como salida deseada de la RNA.***

Escala menor	X
C*m*	-2
C♯*m*/D♭*m*	-1.83
D*m*	-1.66
D♯*m*/E♭*m*	-1.49
E*m*	-1.33
F*m*	-1.16
F♯*m*/G♭*m*	1.16
G*m*	1.33
G♯*m*/A♭*m*	1.49
A*m*	1.66
A♯*m*/B♭*m*	1.83
B*m*	2

$(-\mathbf{2} \leq \boldsymbol{x} \leq -\mathbf{1.6} \cup \mathbf{1.6} \leq \boldsymbol{x} \leq \mathbf{2})$ *Fuente: Author.*

A continuación, los patrones de la red con su respectiva salida deseada:

Tabla VII. ***Secuencia de notas de entrada con su respetiva tónica de la escala mayor (salida).***

Entradas	Salida deseada
C D E F G A B	C
C♯ D♯ F F♯ G♯ A♯ C	C♯/D♭
D E F♯ G A B C♯	D
D♯ F G G♯ A♯ C D	D♯/E♭
E F♯ G♯ A B C♯ D♯	E
F G A A♯ C D E	F
F♯ G♯ A♯ B C♯ D♯ F	F♯/G♭
G A B C D E F♯	G
G♯ A♯ C C♯ D♯ F G	G♯/A♭
A B C♯ D E F♯ G♯	A
A♯ C D D♯ F G A	A♯/B♭
B C♯ D♯ E F♯ G♯ A♯	B

Fuente: Author.

Tabla VIII. ***Secuencia de notas de entrada con su respetiva tónica de la escala menor (salida).***

Entradas	Salida deseada
C D D♯ F G G♯ A♯	C*m*
C♯ D♯ E F♯ G♯ A B	C♯*m*/D♭*m*
D E F G A A♯ C	D*m*
D♯ F F♯ G♯ A♯ B C♯	D♯*m*/E♭*m*
E F♯ G A B C D	E*m*
F G G♯ A♯ C C♯ D♯	F*m*
F♯ G♯ A B C♯ D E	F♯*m*/G♭*m*
G A A♯ C D D♯ F	G*m*
G♯ A♯ B C♯ D♯ E F♯	G♯*m*/A♭*m*
A B C D E F G	A*m*
A♯ C C♯ D♯ F F♯ G♯	A♯*m*/B♭*m*
B C♯ D E F♯ G A	B*m*

Fuente: Author.

La red neuronal artificial modelada debe recibir conocimiento de tipo continuo, razón por la cual, también debe responder con conocimiento de tipo continuo. En la rutina "Escalas.pl" se generan las 24 entradas o patrones que recibirá la red, los cuales se tabulan en las tablas IX y X, para las escalas mayores y menores respectivamente, así como también sus respectivas salidas deseadas.

Tabla IX. ***Entradas y Salidas deseadas a presentar en la red neuronal, referentes a la escala mayor.***

Entradas	Salida deseada
-1 -0.66 -0.33 -0.16 0.33 0.66 1	C
-0.83 -0.49 -0.16 0.16 0.49 0.83 -1	C♯/D♭
-0.66 -0.33 0.16 0.33 0.66 1 -0.83	D
-0.49 -0.16 0.33 0.49 0.83 -1 -0.66	D♯/E♭
-0.33 0.16 0.49 0.66 1 -0.83 -0.49	E
-0.16 0.33 0.66 0.83 -1 -0.66 -0.33	F
0.16 0.49 0.83 1 -0.83 -0.49 -0.16	F♯/G♭
0.33 0.66 1 -1 -0.66 -0.33 0.16	G
0.49 0.83 -1 -0.83 -0.49 -0.16 0.33	G♯/A♭
0.66 1 -0.83 -0.66 -0.33 0.16 0.49	A
0.83 -1 -0.66 -0.49 -0.16 0.33 0.66	A♯/B♭
1 -0.83 -0.49 -0.33 0.16 0.49 0.83	B

$(-\mathbf{1} \leq \boldsymbol{x} \leq \mathbf{1})$ *Fuente: Author*.

Tabla X. ***Entradas y Salidas deseadas a presentar en la red neuronal, referentes a la escala menor.***

Entradas	Salida deseada
-1 -0.66 -0.49 -0.16 0.33 0.49 0.83	C*m*
-0.83 -0.49 -0.33 0.16 0.49 0.66 1	C♯*m*/D♭*m*
-0.66 -0.33 -0.16 0.33 0.66 0.83 -1	D*m*
-0.49 -0.16 0.16 0.49 0.83 1 -0.83	D♯*m*/E♭*m*
-0.33 0.16 0.33 0.66 1 -1 -0.66	E*m*
-0.16 0.33 0.49 0.83 -1 -0.83 -0.49	F*m*
0.16 0.49 0.66 1 -0.83 -0.66 -0.33	F♯*m*/G♭*m*
0.33 0.66 0.83 -1 -0.66 -0.49 -0.16	G*m*
0.49 0.83 1 -0.83 -0.49 -0.33 0.16	G♯*m*/A♭*m*
0.66 1 -1 -0.66 -0.33 -0.16 0.33	A*m*
0.83 -1 -0.83 -0.49 -0.16 0.16 0.49	A♯*m*/B♭*m*
1 -0.83 -0.66 -0.33 0.16 0.33 0.66	B*m*

$(-\mathbf{2} \leq \boldsymbol{x} \leq -\mathbf{1.6} \cup \mathbf{1.6} \leq \boldsymbol{x} \leq \mathbf{2})$ *Fuente: Author.*

Haciendo uso del entorno informático MATLAB® se procede a simular la red. MATLAB® provee muchas facilidades para implementar redes neuronales, permitiendo adaptar los modelos por defecto a nuestra conveniencia para

resolver problemas específicos. Se procede a entrenar la red con las entradas y salidas deseadas previamente indicadas en las tablas *IX* y *X*.

Resultados: *Red neuronal de prueba.*

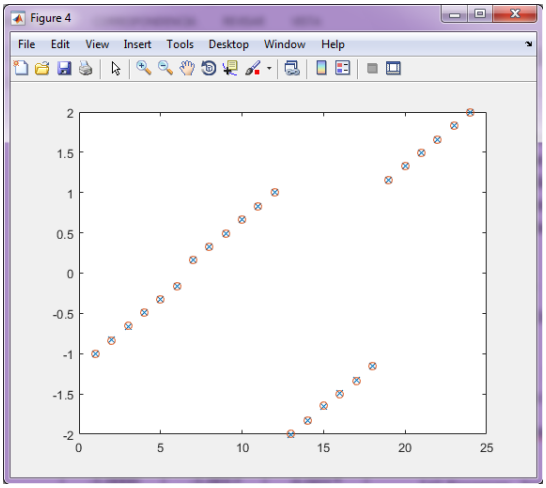
Para la red de prueba, se procedió a realizar el entrenamiento con los 24 patrones previamente explicados y sus respectivas salidas. La función de activación de las neuronas de la capa oculta a usar es *tansig*, mientras que la de la neurona de la capa de salida es la función lineal *purelin*. Como parámetro de entrenamiento solo se modificó el número de épocas (300mil épocas) para obtener un error aceptable, dejando los demás parámetros con los valores proporcionados por defecto. Cabe resaltar que con este número de épocas, la red no estaba entrenada por completa. El tipo de algoritmo de backpropagation usado es *traingd*, el cual usa descenso de gradiente. El error obtenido con los parámetros establecidos es mínimo, por lo que es posible entender la clasificación de las escalas que realiza la red (ver tabla *XI*). En la figura *VII* podemos visualizar una gráfica de *salida deseada vs salida de la red*, donde es visible que el error es mínimo. En la figura *VIII* se visualiza los resultados del entrenamiento de la red.

Tabla XI. Error esperado por la red.

Salida esperada	Salida de la red	Error calculado
-1.0000	-1.0017	0.0017
-0.8300	-0.8362	0.0062
-0.6600	-0.6566	-0.0034
-0.4900	-0.4890	-0.0010
-0.3300	-0.3263	-0.0037
-0.1600	-0.1603	0.0003
0.1600	0.1602	-0.0002
0.3300	0.3307	-0.0007
0.4900	0.4899	0.0001
0.6600	0.6606	-0.0006
0.8300	0.8306	-0.0006
1.0000	1.0002	-0.0002
-2.0000	-1.9988	-0.0012
-1.8300	-1.8302	0.0002
-1.6600	-1.6451	-0.0149
-1.4900	-1.4990	0.0090
-1.3300	-1.3376	0.0076
-1.1600	-1.1597	-0.0003
1.1600	1.1599	0.0001
1.3300	1.3299	0.0001
1.4900	1.4895	0.0005
1.6600	1.6598	0.0002
1.8300	1.8298	0.0002
2.0000	1.9996	0.0004

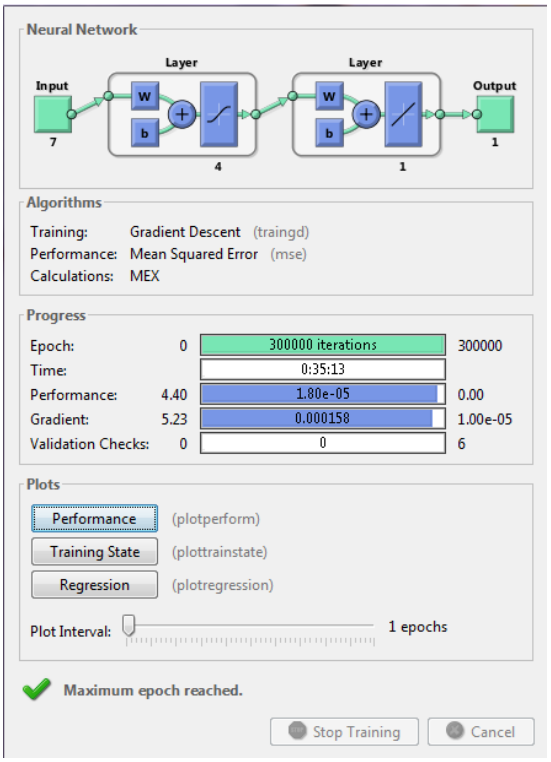
Fuente: Author.

Figura VII. ***Gráfica de salida deseada vs salida de la red.***



Fuente: Author.

Figura VIII. ***Entrenamiento de la red de prueba.***



Fuente: Author.

Figura IX. ***Pesos de la capa oculta.***

```
 1.8758   0.9944   1.7269   2.4403   0.1765   0.3664   1.4463
-1.0406   0.3591   0.9965  -0.2385  -0.4547   0.3134   3.2602
-0.7617   0.8116  -1.2616   2.6754   0.1125   0.1204  -0.7064
 1.1826   0.8296   2.1305   0.0335   1.2384   1.3278   0.2046
```

Fuente: Author.

Bias de la capa oculta: -1.7872, -0.1428, -1.1430, 1.7390.

Pesos de la capa de salida: 2.0056, -2.4093, -3.5133, 1.6121.

Bias de la capa de salida: -0.8462.

Resultados: ***Red neuronal definitiva.***

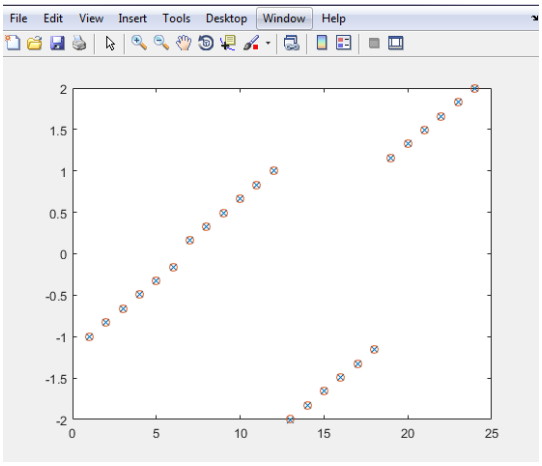
La red neuronal definitiva cuenta con los mismos parámetros de entrenamiento que la red de prueba, con la única diferencia en el número de épocas, que para la red definitiva converge en *604180* iteraciones, durando el entrenamiento *2 horas, 28 minutos y 33 segundos*. En la tabla *XII* se puede visualizar la relación entre la salida deseada, a salida generada por la red y el error, el cual es casi irreconocible. En la figura *X* podemos visualizar una gráfica *salida deseada vs salida de la red*. En la figura *XI* se visualizan los resultados del entrenamiento de la red.

Tabla XI. Error esperado por la red.

Salida esperada	**Salida de la red**	**Error calculado * 1.0e-03**
-1.0000	-1.0001	0.0904
-0.8300	-0.8303	0.2653
-0.6600	-0.6599	-0.0605
-0.4900	-0.4899	-0.0620
-0.3300	-0.3298	-0.2494
-0.1600	-0.1600	0.0148
0.1600	0.1600	-0.0100
0.3300	0.3300	-0.0331
0.4900	0.4900	0.0011
0.6600	0.6600	-0.0452
0.8300	0.8300	-0.0492
1.0000	1.0000	-0.0095
-2.0000	-1.9999	-0.0602
-1.8300	-1.8300	0.0119
-1.6600	-1.6593	-0.7350
-1.4900	-1.4903	0.3226
-1.3300	-1.3305	0.5063
-1.1600	-1.1600	-0.0188
1.1600	1.1600	0.0136
1.3300	1.3300	0.0069
1.4900	1.4895	0.0005
1.6600	1.6598	0.0002
1.8300	1.8298	0.0002
2.0000	1.9996	0.0004

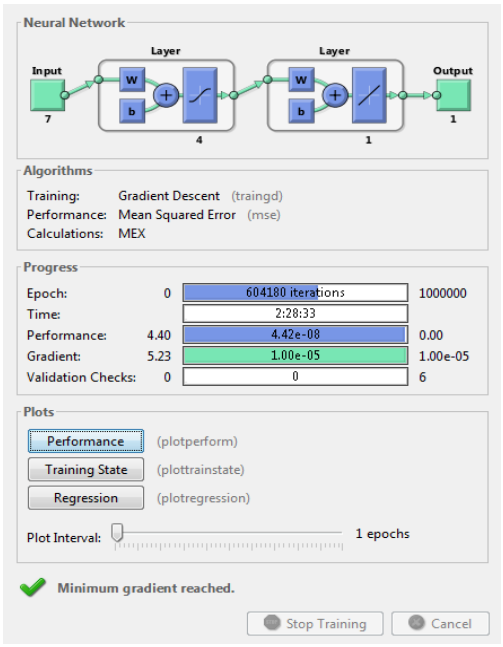
Fuente: Author.

Figura X. Gráfica de salida deseada vs salida de la red.



Fuente: Author.

Figura XI. Entrenamiento de la red definitiva.



Fuente: Author.

Figura XII. Pesos de la capa oculta.

1.8746	1.0070	1.6918	2.4083	0.1949	0.4353	1.3350
-1.0687	0.3766	1.0091	-0.2274	-0.4035	0.2994	3.2900
-0.7331	0.7848	-1.2824	2.6913	0.1296	0.1172	-0.7118
1.1745	0.8480	2.1409	0.0245	1.2805	1.3244	0.1778

Bias de la capa oculta: -1.7385, -0.1046, -1.1473, 1.7593.

Pesos de la capa de salida: 2.0182, -2.4266, -3.5381, 1.6224.

Bias de la capa de salida: -0.8506.

III. Conclusiones.

La capacidad de las redes neuronales (una vez entrenadas) de responder a un conjunto de patrones abre paso a la resolución de muchos problemas que son difíciles (en ocasiones imposibles) de resolver con programación tradicional. La versatilidad de las redes neuronales artificiales nos permite aplicarlas a una gran cantidad de problemas, en todo tipo de áreas. En éste estudio se eligió la música.

Los resultados obtenidos fueron positivos, por lo que se propone para posteriores estudios ampliar la cantidad de escalas musicales que debe aprender la red neuronal artificial, ya que podemos encontrar una gran variedad de escalas, y sería de gran utilidad para músicos y compositores contar con una herramienta que les brinde este beneficio.

IV. Referencias.

[1] Anderson, James. Redes neuronales (2007).

[2] Paletta, Mauricio. Inteligencia Artificial Básica (2010).

[3] A. Danhauser. Teoría de la música (2008).

[4] Freeman, James A. & Skapura, David M. Redes neuronales: Algoritmos, aplicaciones y técnicas de programación (1993).

[4] H. Demuth and M. Beale. Neural Networks Toolbox: User's guide. Version 6, (2009).

[5] Wielemaker, Jan. SWI-Prolog for MS-Windows.

Rutina: Escalas.pl

```
at_end(X, [], [X]).
at_end(X, [H|R], [H|L]) :- at_end(X, R, L).
next_note([H|R], L):- at_end(H, R, L).


generar_escala(_, [], _, []).
generar_escala(X, [X|R_i],
        [H_n|R_n],
      [H_n|Escala]) :- X1 is X+1,
            generar_escala(X1, R_i, R_n, Escala).
generar_escala(X, Int,
        [_|R_n],
        Escala) :-  X1 is X+1,
            generar_escala(X1, Int, R_n, Escala).

generar_escala(Notas, 'Mayor', Escala) :- generar_escala(0,
            [0,2,4,5,7,9,11],
          Notas,
          Escala),!.
generar_escala(Notas, 'Menor', Escala) :- generar_escala(0,
            [0,2,3,5,7,8,10],
          Notas,
          Escala).
generar_escalas(0, _, _).
generar_escalas(N, Notas, Tipo) :- generar_escala(Notas, Tipo, Escala),
          write(Escala),nl,
next_note(Notas, Notas_n),
          N1 is N-1,
                generar_escalas(N1, Notas_n, Tipo).

generar_escalas :- write('ESCALAS MAYORES:'),nl,nl,
            generar_escalas(12,
        ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'],
        'Mayor'),
      nl, write('ESCALAS MENORES:'),nl,nl,
        generar_escalas(12,
        ['C', 'C#', 'D', 'D#', 'E', 'F', 'F#', 'G', 'G#', 'A', 'A#', 'B'],
        'Menor'),!.

generar_data :- write('ESCALAS MAYORES:'),nl,nl,
            generar_escalas(12,
        [-1, -0.83, -0.66, -0.49, -0.33, -0.16, 0.16, 0.33, 0.49, 0.66, 0.83, 1],
        'Mayor'),
      nl, write('ESCALAS MENORES:'),nl,nl,
        generar_escalas(12,
        [-1, -0.83, -0.66, -0.49, -0.33, -0.16, 0.16, 0.33, 0.49, 0.66, 0.83, 1],
        'Menor'),!.
```

Nota: invocar los procedimientos *generar_escalas* y *generar_data*, para generar las escalas en su representación de cifrado musical y en la representación continua (para ser presentada a la red) respectivamente.

Rutina: Scales_Learning.m

```
p = [-1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1;
   -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83;
   -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66;
   -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33;
   0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16;
   0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.49 0.66 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33;
   1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 0.83 1 -1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66];

t = [-1 -0.83 -0.66 -0.49 -0.33 -0.16 0.16 0.33 0.49 0.66 0.83 1 -2 -1.83 -1.66 -1.49 -1.33 -1.16 1.16 1.33 1.49 1.66 1.83 2];

net = newff(minmax(p), [4 1], {'tansig', 'purelin'}, 'traingd');
net.trainParam.epochs = 700000;
[net, tr] = train(net, p, t);
a = sim(net, p);
e = t - a;
```

Éste trabajo forma parte del requisito del curso ***Redes Neuronales***, dictado por el profesor **Manuel Paniccia** de la *Universidad Nacional Experimental de Guayana*.

Puerto Ordaz, Venezuela. Julio del 2017.