CAPITULO 4. ENFOQUE A LA CLASIFICACION DE PATRONES

En este capítulo se presenta la descripción de los distintos algoritmos que permiten a la red, funcionar con mayor eficiencia como clasificadora de patrones, y se presentan las distintas técnicas de diseño que permiten hallar el mejor modelo para la detección de intrusos.

4.1 DISEÑO ENFOCADO A LA CLASIFICACION.

En el capítulo anterior presentamos la versión clásica del algoritmo de Propagación hacia atrás, sin embargo esta versión del algoritmo no es la forma más eficiente de llegar al error mínimo o de lograr que la red generalice. A lo largo de este capítulo se presentarán las variaciones de este algoritmo que permiten obtener un mejor desempeño y se realizará un enfoque de las redes neuronales adaptando los principios que las definen, al problema de la detección de intrusos que finalmente es un problema de clasificación de patrones.

Hasta este momento el lector tiene una noción de cómo es que el algoritmo de propagación hacia atrás converge en busca del error mínimo y se tiene una idea básica de los principios de diseño necesarios para implementar una red neuronal.

Sin embargo, cuando hablamos de redes para clasificación, no estamos hablando necesariamente de un modelo particular de red neuronal. De hecho, existen muchos modelos que nosotros se puede aplicar para esta tarea como lo es el Perceptrón Multicapa y por otro lado no es la única tarea que este tipo de red puede realizar, ya que también puede ser usada para problemas de auto asociación, predicción de series de tiempo, o aproximación de funciones por ejemplo.

El problema más simple de clasificación es la decisión binaria que se ocupa de decidir si una muestra pertenece a una clase o a otra. Como se puede inferir, este tipo de problemas son mejor implementados ocupando únicamente una neurona de salida. La red será entrenada para producir una salida de activación alta para una decisión y una salida baja para otra [MAS93].

Cuando una red entrenada se pone a prueba con una muestra desconocida, se decidirá a que categoría corresponde basados en el nivel de activación logrado por la neurona de salida. Si la activación es al menos tan grande como un umbral predefinido, entonces estamos en capacidad de hacer la decisión. En muchos casos, el umbral se ubica justo en medio de los dos posibles límites. Esto tiene sentido lógico y en la práctica es usualmente efectivo en la ausencia de información adicional. Sin embargo puede ser loable prejuiciar la decisión al ubicar más cerca el umbral de un lado que del otro.

No obstante existen otros métodos como la decisión bayesiana que son más adecuados para decidir en base a la probabilidad de que el nivel de activación pertenezca a una clase o a otra. Sin embargo este tipo de métodos requieren información a priori que en muchos casos es difícil de conocer, como es nuestro caso ya que no sabemos cuales son las posibilidades que existen de que una muestra caiga en una clase o en otra ya que el flujo de datos en una red de comunicación sucede aleatorio.

En general, cuando hablamos de que ocupamos una red para aproximar una función, debemos utilizar neuronas lineales en la capa de salida. Sin embargo para el caso de clasificación de patrones se recomienda el uso de neuronas de sigmoideas ya que los problemas de clasificación tratan de que la neurona de salida ocupe valores cercanos a los límites 0 y +1 indicando así si la muestra pertenece a uno u otra clase.

De manera que no necesitamos que se use todo el rango de valores lineales para este tipo de problema. Esto presentará un problema debido a que como ya se sabe, el algoritmo de Gradiente Descendiente, en el que se basa la propagación hacia atrás, trabaja muy lento en la zona no lineal de la función sigmoidea debido a que el cambio de la pendiente se hace mínimo en estas regiones haciendo que la pendiente tienda a cero y provocando que la delta de los pesos sea mínima debido a que depende proporcionalmente de la derivada de la función en este punto. No obstante, el algoritmo de Propagación hacia atrás sufrirá una modificación para enfrentar este problema como se verá en el capítulo 4.2.

Además de esto y debido a que es importante obtener tantos patrones y características de los datos como sea posible, se recomienda que la red trabaje con dos capas ocultas, de manera que si algunos patrones fueron omitidos en la primera capa, sea posible que la segunda capa los detecte. A esto se puede aunar que, independientemente del número de neuronas ocultas que cada capa tenga, se recomienda que se mantenga el mismo número de neuronas ocultas en una capa que en la otra.

Así pues con la información que se ha recopilado concluimos que la red a utilizar para este problema deberá ser una red Perceptrón Multicapa con cuatro capas, en las que la capa de salida tenga cinco neuronas para seleccionar una salida de cinco posibles clasificaciones, la capa de entrada tenga un número de neuronas equivalente a ocho por tamaño máximo de muestra y las dos capas ocultas tengan un número de neuronas que tentativamente se pondrán como diez y a partir de los resultados obtenidos nos moveremos hacia el mejor modelo. Por otro lado nos queda definir, cuales van a ser los parámetros libres de la red y que por el momento se dirá que serán inicializados aleatóriamente en un rango uniformemente distribuido y pequeño.

En cuanto al momento y al coeficiente de velocidad de aprendizaje se establecerá que serán dependientes de conexión y quedará definida en el capítulo 4.2 la forma en que se utilizarán.

A continuación nos resta definir cual va a ser la forma de presentar los datos a la red, entendiendo que la tarea más importante es la de lograr la generalización de nuestra red para realizar una correcta clasificación de patrones. Como mencionamos anteriormente el método más eficiente computacionalmente es el método de Batch en el que el conjunto de datos es presentado a la red y al final de cada epoch se lleva a cabo el ajuste acumulado de pesos. Además de esto en el capítulo 6 se definirá la forma en que estos datos serán preprocesados y postprocesados para lograr un mejor desempeño de la red.

Finalmente nos queda definir el algoritmo de entrenamiento más adecuado para la red Multicapa que permita a la red clasificar adecuadamente. Para esto en el siguiente capítulo se presentarán las distintas variaciones del algoritmo de propagación hacia atrás que en base a distintos enfoques llevan a la convergencia del error a un valor mínimo. Finalmente se evaluarán y compararán para tomar la decisión respecto a que algoritmos se pondrán a prueba en la fase práctica donde finalmente se revelará la eficiencia de cada uno de ellos para el problema en cuestión.

4.2 ALGORITMOS DE RAPIDO APRENDIZAJE.

La función de costo que se usa comúnmente para entrenar a las redes Multicapa es la media del error cuadrado. A continuación se muestran algoritmos que, basados en el algoritmo de Propagación hacia atrás, usan el gradiente de la función de costo para determinar como ajustar los pesos para minimizar el error.

La ventaja respecto al modelo clásico es que estos algoritmos presentan distintos enfoques para atacar a un mismo problema de manera que las heurísticas de las que estos algoritmos nacen se enfocan a lograr una solución particular a cada tipo de problema haciendo que el error converja más rápido o que la red generalice mejor.

4.2.1 DELTA-BAR-DELTA, SOLUCION AL COEFICIENTE DE APRENDIZAJE VARIABLE.

Como habíamos mencionado anteriormente, el uso de un coeficiente de velocidad de aprendizaje que sea dependiente de la conexión permite a la red acelerar el aprendizaje donde lo requiere y estabilizarlo en los pesos que no requieran ser modificados ampliamente. Se recuerda que el uso de una velocidad de aprendizaje constante acarrea problemas de oscilación si el valor del coeficiente es muy alto, y si el coeficiente es muy lento se puede tardar mucho tiempo en converger. Además de esto, la literatura nos indica que el coeficiente de velocidad óptimo cambia durante el proceso de entrenamiento por lo que no es factible usar un coeficiente fijo [HAY94].

El uso de un coeficiente de velocidad de aprendizaje adaptivo permite al aprendizaje mantenerse estable como observamos en la Figura 4.1 donde se muestra el recorrido de los pesos en la superficie de error para un coeficiente η variable.

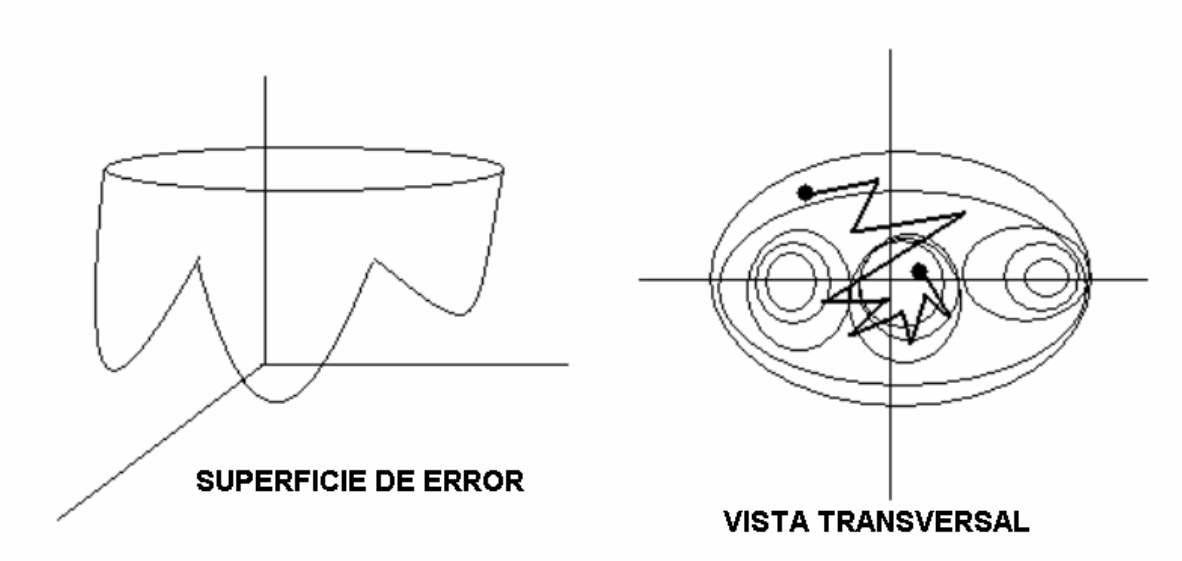

Figura 4.1. Recorrido de los pesos con η variable.

El algoritmo funciona evaluando el error con cada η, si el error es mayor que el anterior, entonces el coeficiente se reduce, y si el error es menor que el anterior error, entonces el coeficiente se incrementa. De esta manera el valor de η se aproxima a su valor óptimo.

4.2.2 PROPAGACIÓN ELÁSTICA HACIA ATRÁS.

Este algoritmo presenta una solución al problema de la lentitud de aprendizaje en los límites de la tangente sigmoidea debido a que en la clasificación de patrones nos interesa trabajar a las neuronas en estos límites. Como se sabe, la función tiene una pendiente que tiende a cero al acercarse a estos límites por lo que los cambios de los pesos son proporcionales a la pendiente y el aprendizaje se hace lento. Para solucionar este problema, el algoritmo no toma en cuenta la magnitud de la derivada para modificar los cambios y tomo únicamente el signo de la derivada de la función, de esta manera el algoritmo propone la magnitud de cambio y convierte al algoritmo de gradiente descendiente en un indicador de dirección ya que únicamente nos interesa el signo de la

derivada. La magnitud del cambio que se aplicará a cada peso estará determinada por otro coeficiente que se llamará delta de cambio, el cual se irá actualizando en base a la función de error producida.

Similar a lo que ocurre en el algoritmo de Delta-Bar-Delta, la propagación elástica incrementa la magnitud de cambio cuando la anterior derivada y la actual coincidan en el signo, es decir, que si la dirección del gradiente no presenta cambios, el algoritmo incrementa la delta de cambio para que avance más rápido en esa dirección y así se acerque rápidamente al punto mínimo, en cambio cuando se presenta una variación de signos, el algoritmo entiende que se ha cambiado la dirección de convergencia y debe frenar el avance por lo que reduce el valor de la delta de cambio, de manera que busca en esa dirección con más cautela para hallar el punto de convergencia. De esta forma, siempre que el cambio en la dirección de los pesos esté oscilando el algoritmo se frenará [BEA03].

4.2.3 ALGORITMOS DE GRADIENTE CONJUGADO, PRECISIÓN EN LA DIRECCION DE CAMBIO.

El algoritmo de propagación hacia atrás realiza los cambios en la dirección del negativo del gradiente, que es la dirección donde se presenta el mayor cambio de pendiente, y es donde la función decrecerá más rápido. Sin embargo, resulta que aunque la función decrece más rápido en esta dirección, no significa que llega más rápido al punto mínimo. Esto se debe a que la superficie de error, puede estar llena de pendientes grandes de descenso que luego terminen en un plano o en una subida que provoque estancamiento.

El paradigma del gradiente conjugado, propone que la dirección de cambio esté definida por el resultado de una búsqueda en distintas direcciones conjugadas del gradiente que indique al algoritmo la dirección en la que se convergerá más rápido. Esta búsqueda a

su vez definirá el tamaño de cada incremento en el algoritmo, dejando a un lado el coeficiente de velocidad de aprendizaje η ya que el ajuste se modifica a cada iteración [BEA03]. Existen distintos algoritmos que realizan esta búsqueda del gradiente conjugado en la superficie de error, sin embargo en este proyecto se estudiará únicamente el algoritmo de actualización Polak Ribiere presentado a continuación.

4.2.3.1 ACTUALIZACIÓN POLAK RIBIÉRE.

Todos los algoritmos de gradiente conjugado, comienzan por usar la magnitud del gradiente en la primera iteración. Posteriormente, se determina la dirección óptima con una línea de búsqueda que determina también la distancia óptima de movimiento. Esta línea de búsqueda esta determinada por la combinación del gradiente calculado anteriormente con el nuevo gradiente calculado actualmente.

Para este algoritmo la combinación esta determinada entre el producto de la diferencia del cambio en el gradiente por el gradiente actual, y el producto del gradiente anterior por sí mismo. La dirección de búsqueda se reiniciara periódicamente al valor del negativo del gradiente, permitiendo que el algoritmo retome la búsqueda por la dirección de mayor pendiente. Esto sucede cuando el número de iteraciones es igual al número de parámetros libres de la red.

4.2.4 OTROS ALGORITMOS.

Antes de continuar es importante mencionar a otro tipo de algoritmos cuya operación se concentra en calcular matrices hessianas que agreguen un segundo orden al algoritmo de propagación hacia atrás. Es decir que la matriz hessiana es la matriz de derivadas de segundo orden de la función de costo, en los valores actuales de pesos, para la búsqueda de la dirección óptima del gradiente.

Sin embargo este tipo de procedimiento implica cálculos demasiado complejos, ya que el cálculo de la matriz hessiana toma mucho tiempo de procesador, y es proporcional al número de parámetros de la red. Por lo que para redes de gran tamaño es poco recomendable el uso de estos algoritmos. A estos algoritmos se les conoce como del tipo Quasi-Newton, debido a que se basan en el método de Newton para optimizar la dirección de cambio.

En base a estos algoritmos, se desarrolló también un algoritmo conocido como de Levenberg- Marquardt, el cual también busca hallar la dirección de cambio en base a derivadas de segundo orden, pero sin requerir del cálculo del hessiano, sino de una aproximación de este. De esta forma el algoritmo de Levenberg- Marquardt, puede aproximar al método de Newton en cuanto se encuentre avanzando en la dirección correcta, y por otro lado puede aproximar al método del gradiente descendiente para cuando viaja en dirección incorrecta permitiendo frenar al algoritmo, ya que como se sabe, el algoritmo del gradiente descendiente es más lento que los algoritmos de optimización de dirección y magnitud de cambio [BIS95].

En la práctica este algoritmo obtiene el mejor desempeño, frente a todos los algoritmos antes mencionados, para los problemas de aproximación de funciones y ajuste de curvas, la razón es que este algoritmo esta diseñado para trabajar con la función de costo de media de errores cuadrados MSE que es una función aproximadamente linear. No obstante, su desempeño es pobre para la clasificación de patrones al igual que los métodos quasi-newton, por lo que su uso no será aplicado al problema en cuestión ya que, el lector recordará, que no es nuestro interés realizar un ajuste de curva sino una clasificación de patrones.

4.3 COMPARACION DE ALGORITMOS PARA CLASIFICACION.

En este momento contamos con tres enfoques distintos para el entrenamiento de una red Multicapa. El primer enfoque, el del algoritmo delta-bar-delta, supera las dificultades de limitar a la red a usar un coeficiente de velocidad de aprendizaje constante, al hallar mediante aproximaciones el valor óptimo de este coeficiente y permitir que la red avance rápidamente cuando tenga que hacerlo y que se detenga cuando este cerca del punto mínimo de error.

Este enfoque, sin embargo, mantiene que la dirección óptima de cambio y la magnitud de cambio a usar es la que esta dada por el negativo del gradiente, basándonos en la premisa de que la función tiene la mayor pendiente de decrecimiento en esta dirección. Sin embargo aunque esta premisa suele ser cierta, y aún cuando la dirección finalmente nos lleve a un mínimo, la magnitud del gradiente puede no ser suficiente para alcanzar una velocidad adecuada, aún cuando la velocidad de aprendizaje este en el valor máximo permitiendo que toda la magnitud del gradiente represente el cambio y sin atenuarse.

Esto sucede principalmente en el caso de la clasificación de patrones debido a que cuando entrenamos a las neuronas procuramos que la salida esperada se encuentre en los límites de activación de la neurona para saber si la muestra pertenece a una clase, u a otra completamente distinta. El problema al trabajar a las neuronas en los límites es que el algoritmo de propagación hacia atrás, calcula el gradiente en base a las derivadas parciales de las funciones de las que depende la función de error.

Una de estas funciones es la de activación de las neuronas, si recordamos la gráfica de la función sigmoide, se verá que en los límites la derivada tiende a cero, debido a que la derivada expresa la pendiente y la pendiente en estos puntos es casi horizontal es decir,

que tiende a un valor muy pequeño, por lo que al calcular la magnitud del gradiente, esta no logra ser suficientemente grande para hacer que la configuración de pesos viaje rápidamente hacia el valor de esta configuración que haga el error mínimo.

Para enfrentar este problema, el algoritmo de propagación elástica permite que la magnitud del gradiente sea omitida y este algoritmo propone un valor con el cuál ira comprobando cuál será el valor óptimo de este valor para que el algoritmo converja más rápido, haciendo más grande el valor cuando la dirección de cambio se comprueba correcta y atenuando el valor de cambio cuando la dirección es errónea, o se ha pasado del valor correcto.

Sin embargo este enfoque no contempla un factor importante que sucede en las superficies de error de una red Multicapa, y es que en este tipo de redes, las superficies de error varían significativamente y no se puede establecer una tendencia clara en la cual nosotros podamos definir que la dirección del negativo del gradiente es la dirección en la que vayamos a llegar más rápido a un punto mínimo.

Aunque desde luego la propiedad de que esta dirección es la de más rápido decrecimiento es verdadera, y retomando la analogía con la montaña y el alpinista en la Figura 4.2, se puede descender rápidamente de la punta de la montaña por una de las veredas, y sin embargo nos podríamos topar con una meseta de gran longitud, y posteriormente nos podríamos topar con una nueva cresta, y sin embargo, si hubiéramos tomado el otro descenso que aunque no era tan rápido como el primero, al final nos hubiera permitido llegar más rápido a las faldas de la montaña que el lado que nos proponía mayor inclinación.

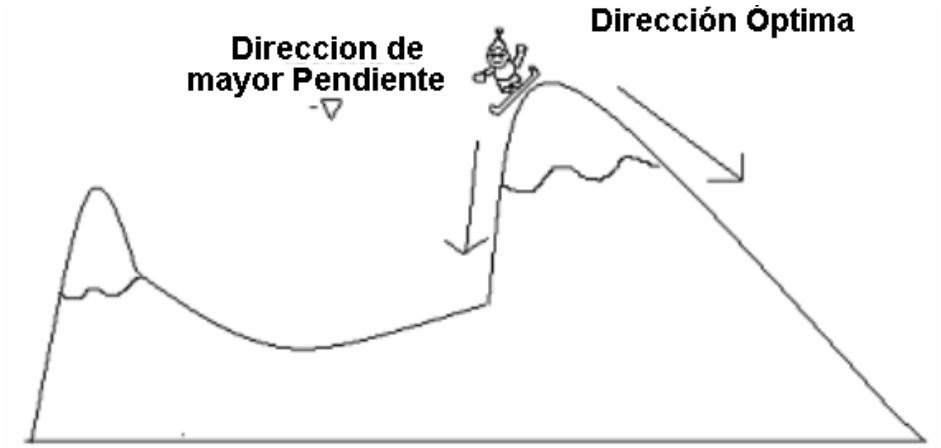

Figura 4.2. Dirección del negativo del gradiente contra Dirección óptima.

Para enfrentar este problema, surge el tercer enfoque el cuál maneja el uso de un gradiente conjugado para evaluar la dirección óptima de descenso. Esta dirección se calcula en base a una línea de búsqueda que procura hallar, en base a la dirección del gradiente y del conjugado de éste, cual será la dirección óptima.

A continuación se presentará una comparación de estos tres enfoques basados en los resultados obtenidos por la empresa Math Works, creadora del software de MATLAB © y que publicaron en el manual de usuario de la herramienta de programación de redes neuronales.

En esta comparación se diseñaron tres redes neuronales distintas para un problema de clasificación de patrones de baja complejidad, uno de mediana complejidad, y uno de gran complejidad, y se entrenaron con los tres tipos de enfoques para observar el tiempo de convergencia.

En el primer caso la red a diseñar debía distinguir la paridad de números de 3 bits. El diseño de la red consta de tres neuronas de capa de entrada, dos capas ocultas con 10 neuronas y una capa de salida de una neurona y se buscó un error global de .001. En el

segundo caso la red debía aprender a clasificar tumores entre benignos y malignos basados en la descripción celular arrojada por un análisis de microscopio. La red diseñara constaba de nueve neuronas de entrada dos capas ocultas de cinco neuronas y una capa de salida de dos neuronas, el error global se busco menor a 0.012. Finalmente el tercer problema se encargo de entrenar a una red para decidir si un individuo tenía diabetes basados en información de su historial médico, familiar, estilo de vida y un examen médico de análisis de sangre, presión, etc. La red diseñada constó de ocho neuronas de entrada dos capas ocultas de 15 neuronas y dos neuronas de salida buscando un error menor a .05. Los resultados arrojados por este estudio se muestran en la tabla 4.1 a continuación.

Tabla 4.1. Comparación de eficiencia entre Algoritmos [BEA03].

	PARIDAD	LUGAR	CANCER	LUGAR	DIABETES	LUGAR	TOTAL	LUGAR
Propagación Elástica	1	1	1.04	2	1	1	3.04	1
G.C. Polak	1.38	3	1.09	4	1.28	4	3.75	4
Delta bar delta	7.26	6	3.9	6	23.73	6	34.89	6

Las columnas indican el problema enfrentado y el tiempo normalizado de convergencia de cada uno de los algoritmos para dicho problema, a continuación la siguiente columna muestra el lugar obtenido para cada algoritmo en relación a los otros. El tiempo normalizado se obtiene de dividir el tiempo que tardo cada algoritmo en converger entre el tiempo que tardo en converger el algoritmo más rápido para ese problema específico. La última columna muestra el lugar global que cada algoritmo obtuvo en la resolución de problemas de clasificación de patrones.

Como se puede observar, el algoritmo que más rápido converge para la mayoría de los problemas es el de Propagación elástica, y que seguramente se debe a que este algoritmo fue diseñado para afrontar las dificultades de trabajar a las neuronas en sus límites para clasificación de patrones. Por otro lado y muy cerca de este algoritmo se encuentra el de gradiente conjugado escalado el cual se dice que se desempeña mejor que otros cuando el número de parámetros libres de la red es alto, y a diferencia de la propagación elástica, este algoritmo no se degrada cuando se busca un error global muy bajo. El uso de memoria del algoritmo de gradiente conjugado es relativamente modesto por lo que se recomiendan para trabajar con redes grandes.

Como se puede observar de estos tres ejemplos, la mayoría de los algoritmos trabaja rápidamente para la clasificación de patrones, sin embargo, el algoritmo de η variable, delta-bar-delta, parece ser el más lento de todos ya que converge en mucho más tiempo que los demás, y en los tres casos fue el más lento. No obstante, no debe ser descartado ya que como mencionamos en la sección 3.4, el hecho de que un algoritmo converja rápidamente no representa que realizará una buena generalización.

4.4 RED ELMAN

Existe también, otro tipo de arquitectura de red neuronal, derivada de las redes Perceptrón Multicapa, que se conoce como redes recurrentes, cuyo nombre se deriva de los lazos de retroalimentación que existen entre sus salidas y entradas. Este tipo de arquitectura es de especial interés debido a que su uso en clasificación de patrones es significativo.

La retroalimentación existente entre las salidas y entradas de las neuronas permite a la red efectuar reconocimiento de patrones que tengan una relación secuencial entre sí. La Figura 4.3 nos muestra la arquitectura de dicha red neuronal.

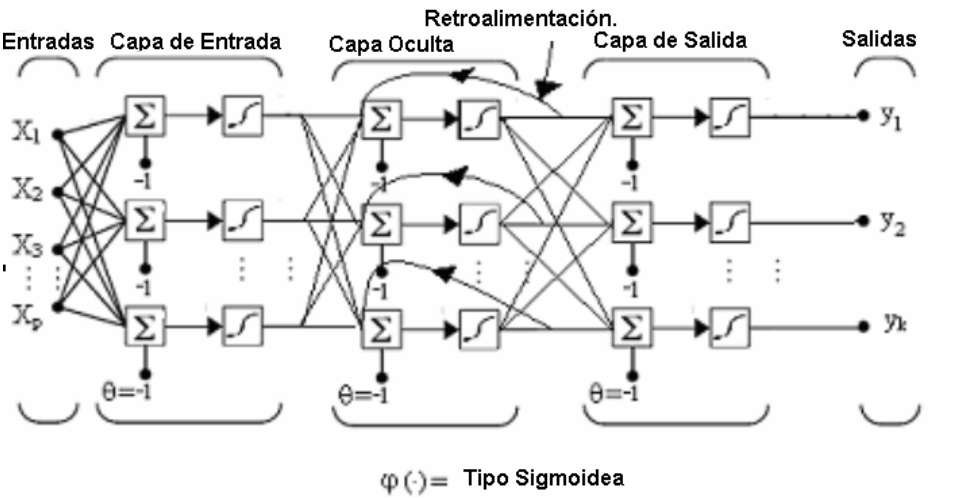

Figura 4.3. Arquitectura de la red Elman.

Como se puede ver de la Figura 4.3 la retroalimentación se lleva a cabo de las salidas de las neuronas ocultas a las entradas de las mismas. Es decir, que se le agrega una nueva entrada a las neuronas ocultas que es la salida anterior de las mismas multiplicada por un peso ω. De esta manera la red puede incluir los patrones de las entradas anteriores a estos nuevos patrones y así tener una cierta memoria de los sucesos que van aconteciendo. De tal forma que puede recordar patrones y establecer patrones de secuencias en los datos [BEA03].

Este hecho será de gran importancia debido a que se usará este tipo de red para identificar si existen patrones secuenciales entre los datos de entrenamiento que se usarán para diseñar nuestro sistema de detección de intrusos. Por otro lado es importante que esta red pueda ser entrenada con los mismos algoritmos que una red Perceptrón Multicapa, de manera que no hay necesidad de diseñar nuevos algoritmos para este tipo de red.

El único inconveniente será que, el entrenamiento de esta red, será un poco menos exacto que el de una red Perceptrón Multicapa debido a que el entrenamiento usa una aproximación del gradiente, al no incluir el efecto de los pesos de retroalimentación en el

cálculo del mismo. Sin embargo este ligero inconveniente es menguado por la capacidad que la red tiene para hallar patrones distintos a los que una red Perceptrón Multicapa puede encontrar.

En la etapa de implementación práctica de este proyecto se pondrán a prueba los dos tipos de redes neuronales para observar su desempeño y decidir cual de las dos tiene un mejor resultado y por lo tanto resulta más adecuada para implementarse como solución al problema.