Comparación de Redes Neuronales Utilizados en Sistemas de Soportes de Desiciones

Insfrán Juan Carlos, Quintana Darío Abelardo, Vicentín Carlos Manuel

1 Marco Teórico

1.1 Red Hopfield

Una de las mayores contribuciones al área de las redes neuronales fue realizada en los años 1980 por John Hopfield, quien estudió modelos auto-asociativos que presentaban algunas similaridades con los perceptrones, pero incluía también grandes diferencias [1].

Las redes Hopfield son redes monocapa con N neuronas. Sus valores de salida son binarios: 0/1 ó 1/1 [1].

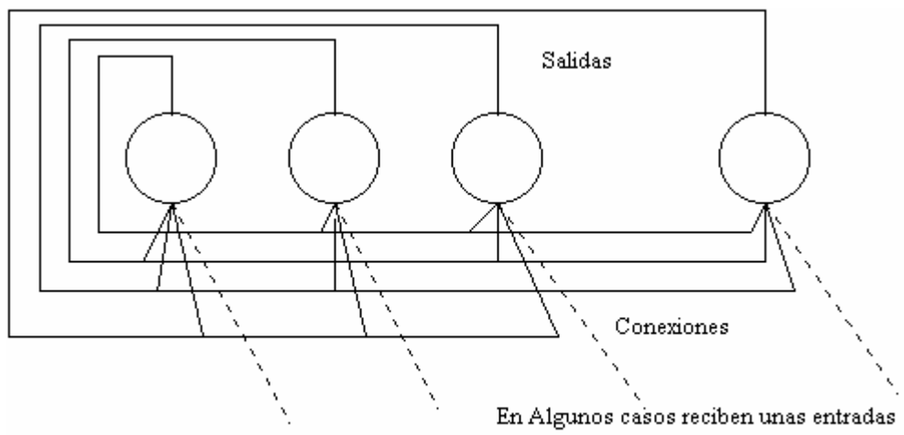
En el modelo original (DH: Discret Hopfield) las funciones de activación de las neuronas eran del tipo escalón [1].

La función escalón se asocia a neuronas binarias en las cuales cuando la suma de las entradas es mayor o igual que el umbral de la neurona, la activación es 1, si es menor, la activación es 0 (ó –1). Las redes formadas por este tipo de neuronas son fáciles de implementar en hardware, pero sus capacidades están limitadas, debido a que la función escalón no puede definir la derivada en el punto de transición y esto no ayuda a los métodos de aprendizaje en los cuales se usan derivadas [2].

$$f(x)=\begin{cases}+1 & x > \Theta_i \\ -1 & x < \Theta_i\end{cases}$$

Posteriormente Hopfield desarrollo una versión continua con entradas y salidas analógicas, utilizando neuronas con activación de tipo sigmoidal (CH: Continous Hopfield) [1]. Esta versión no será explicada en más detalle porque no fue la utilizada para la realización del trabajo práctico.

Otra característica de este tipo de redes es que no son auto-recurrentes, es decir que las salidas de las neuronas se comunican con todas las demás pero no consigo misma [1].



MODELO DE HOPFIELD

1.1.1 Aprendizaje

El mecanismo de aprendizaje utilizado en una red Hopfield es OFF-LINE, por lo tanto se distingue una etapa de aprendizaje o entrenamiento y una etapa de operación o funcionamiento. El aprendizaje en este mecanismo supone la desconexión de la red [1].

Una vez que la red aprendió los pesos de la red permanecerán fijos. Gracias a este carácter estático, no presenta problemas de estabilidad en su funcionamiento [1].

Utiliza un aprendizaje no supervisado de tipo hebbiano, de tal forma que el peso de una conexión entre una neurona i y otra j se obtiene mediante el producto de los componentes i-ésimo y j-ésimo del vector que representa la información o patrón que debe almacenar. Si el numero de patrones a aprender es M, el valor definitivo de cada uno de los pesos se obtiene mediante la suma de los M productos obtenidos por el procedimiento anterior, un producto por información a almacenar [1].

El algoritmo de aprendizaje suele expresarse utilizando una notación matricial. En tal caso podría considerarse una matriz W de dimensiones NxN que representase todos los pesos de la red:

$$W=\begin{pmatrix} W_{11} & W_{21} & W_{31} \ldots. & W_{N1} \\ W_{12} & W_{22} & W_{32} \ldots. & W_{N2} \\ & & \ldots\ldots\ldots\ldots & \\ W_{1N} & W_{2N} & W_{3N} \ldots & W_{NN} \end{pmatrix}$$

Esta matriz es simétrica, al cumplirse que wij=wji y tiene una diagonal principal con valores nulos debido a la no existencia de conexiones auto-recurrentes (wii=0) [1].

Además se tendría el conjunto de los M vectores que representas las informaciones que ha de aprender la red:

E1 = [e1(1), e2(1),…, eN(1)]

E2 = [e1(2), e2(2),…, eN(2)]

……..

EM = [e1(M), e2(M),…, eN(M)]

A partir de estos datos de entrada la matriz de peso se calcularía de la siguiente manera:

$W=\Sigma[EkTEk - I]$ con k=1 a M.

Donde I es la matriz identidad de dimensiones NxN que anula los pesos de las conexiones auto-recurrentes y EkT es la transpuesta de Ek.

1.1.2 Funcionamiento

Las redes Hopfield son muy utilizadas en tareas denominadas auto asociación (por ello se dice que es una red auto-asociativa), que tiene que ver con regenerar información que se presenta en la entrada de la red en forma incompleta o distorsionada [1].

Para ello varias informaciones (patrones) diferentes pueden ser almacenadas en la red, durante la etapa de aprendizaje. Esta información debe presentarse como vector (con una configuración binaria si la red es discreta, y como conjunto de valores reales si es continua) con tantas componentes como neuronas (N) tenga la red [1].

En el instante inicial cada neurona recibe el elemento del vector que le corresponde como entrada. En este momento, la salida de las neuronas coinciden con los valores den entrada [1].

Luego la red inicia iteraciones en las que los valores de entrada de cada neurona de la red será igual a la sumatoria de los productos entre las salidas de cada una de las otras neuronas por los pesos correspondientes a las conexiones. A cada entrada se le aplicará, dentro de la neurona, la función de transferencia obteniéndose el valor de salida correspondiente, 0/1 ó -1/1 si la red es discreta [1].

Las iteraciones concluirán cuando las salidas de las neuronas se estabilizan, lo cual ocurrirá cuando dejen de cambiar de valor. En este instante se dice que la red ha convergido y la salida de la misma representará la información almacenada que más se parece a la información presentada en la entrada.

Por otro lado, la actualización de las salidas de las neuronas de la red se pueden dar de 2 formas: simultánea (red Hopfield con funcionamiento paralelo o síncrono) y secuencial (red Hopfield con funcionamiento secuencial o asíncrono). En el primer caso las salidas de las neuronas es generada al mismo tiempo por todas ellas en cada iteración y utilizadas como entrada en la siguiente, actualizándose de esta manera todas las neuronas a la vez. En el segundo caso las neuronas trabajan de forma secuencial, actualizándose solo la salida de una neurona en cada iteración. En este caso la salida a la que converge la red puede ser diferente en función del orden de la secuencia de activación de las neuronas [1].

1.2 La Red Backpropagation

La Red Backpropagation es un método para que una red neuronal aprenda la asociación que existe entre los patrones de entrada y las clases correspondientes, utilizando varios niveles de neuronas.

El funcionamiento de la red Backpropagation (BPN) consiste en el aprendizaje de un conjunto predefinido de pares de entradas-salidas dados como ejemplo: primero se aplica un patrón de entrada como estímulo para la primera capa de las neuronas de la red, se va propagando a través de todas las capas superiores hasta generar una salida, se compara el resultado en las neuronas de salida con la salida que se desea obtener y se calcula un valor de error para cada neurona de salida. A continuación, éstos errores se transmiten hacia atrás, partiendo de la capa de salida hacia todas las neuronas de la capa intermedia que contribuyan directamente a la salida, recibiendo el error aproximado a la neurona intermedia a la salida original. Basándose en el valor del error recibido, se reajustan los pesos de conexión de cada neurona, de manera que en la siguiente vez que se presente el mismo patrón, la salida esté más cercana a la deseada.

La importancia de la red backpropagation consiste en su capacidad de auto-adaptar los pesos de las neuronas de las capas intermedias para aprender la relación que existe ente un conjunto de patrones de entrada y sus salidas correspondientes. Es importante la capacidad de generalización (facilidad de dar salidas satisfactorias a entradas que el sistema no ha visto nunca en su fase de entrenamiento).

1.2.1 Funcionamiento del algoritmo

El método de regla delta generalizada utilizada por las redes BPN actualiza los pesos con la diferencia entre la salida deseada y la obtenida

Dada una neurona (unidad Ui) y la salida que produce, yi , el cambio que se produce en el peso de la conexión que une la salida de dicha neurona con la unidad Uj (wji) para un patrón de aprendizaje p determinado es:

$\Delta wji(t+1) = \alpha \delta pj ypi$

En donde el subíndice p se refiere al patrón de aprendizaje concreto, y α es la constante o tasa de aprendizaje.

En redes multinivel en principio no se puede conocer la salida deseada de las neuronas ocultas para poder determinar los pesos en la función de error cometido. Inicialmente podemos conocer la salida deseada de las neuronas de salida. Según esto, si consideramos la unidad Uj de salida, entonces definimos:

$\delta pi = (dpj - ypj).f'(netj)$

donde dpj es la salida deseada de la neurona j para el patrón p y netj es la entrada neta que recibe la neurona j.

Este término representa la modificación que hay que realizar en la entrada que recibe la neurona j. En caso de que dicha neurona no sea de salida, el error que se produce estará en función del error que se cometa en las neuronas que reciban como entrada la salida de dicha neurona. Esto es lo que se denomina como procedimiento de propagación de error hacia atrás.

Según esto, en el caso de que Uj no sea una neurona de salida, el error que se produce está en función del error que se comete en las neuronas que reciben como entrada la salida de Uj:

$$\delta pj = \sum_{k} (\delta pkwkj).f'(netj)$$

donde el rango de k cubre a todas las neuronas a las que está conectada la salida de Uj. De esta forma el error que se produce en una neurona oculta es la suma de los errores que se producen en las neuronas a las que está conectada la salida de ésta, multiplicado cada uno de ellos por el peso de la conexión.

1.2.2 Adición de un momento en la regla delta generalizada

En la implementación del algoritmo, se toma una amplitud de paso que viene dada por la tasa de aprendizaje. A mayor tasa de aprendizaje, mayor es la modificación de los pesos

en cada iteración, con lo que el aprendizaje será más rápido, pero por otro lado, puede dar lugar a oscilaciones. Rumelhart, Hinton y Williams sugirieron que para filtrar estas oscilaciones se añada en la expresión de incremento de los pesos un término (momento),β , de manera que dicha expresión quede:

$wji(t+1) = wji(t) + \alpha\delta pjypi + \beta(wji(t) - wji(t+1))$

$\Delta wji(t+1) = \alpha\delta pjypi + \beta\Delta wji(t)$

donde β es una constante que determina el efecto t+1 del cambio de los pesos en el instante t.

Con este momento se consigue la convergencia de la red en menor número de iteraciones, ya que si en t el incremento de un peso era positivo y en t+1 también, entonces el descenso por la superficie de error en t+1 es mayor. Sin embargo, si en t era positivo y en t+1 es negativo, el paso que se da en t+1 es más pequeño, lo cual es adecuado, ya que significa que se ha pasado por un mínimo y que los pesos deben ser menores para poder alcanzarlo.

Resumiendo, el algoritmo Backpropagation queda finalmente:

$wji(t+1) = wji(t) + [\Delta wji(t+1)]$

$wji(t+1) = wji(t) + [\alpha\delta pjypi + \beta\Delta wji(t)]$

1.2.3 Estructura y aprendizaje de la red *backpropagation*

En una red Backpropagation existe una capa de entrada con n neuronas y una capa de salida con m neuronas y al menos una capa oculta de neuronas internas.

La aplicación del algoritmo tiene dos fases, una hacia delante y otra hacia atrás. Durante la primera fase el patrón de entrada es presentado a la red y propagado a través de las capas hasta llegar a la capa de salida. Obtenidos los valores de salida de la red, se inicia la segunda fase, comparándose éstos valores con la salida esperada para obtener el error. Se ajustan los pesos de la última capa proporcionalmente al error. Se pasa a la capa anterior con una retropopagación del error, ajustando los pesos y continuando con este proceso hasta llegar a la primer capa. De esta manera se han modificado los pesos de las conexiones de la red para cada patrón de aprendizaje del problema, del que conocíamos su valor de entrada y la salida deseada que debería generar la red ante dicho patrón.

La técnica Backpropagation requiere el uso de neuronas cuya función de activación sea continua, y por lo tanto, diferenciable. Generalmente, la función utilizada será del tipo sigmoidal.

1.2.4 Pasos para aplicar el algoritmo de entrenamiento

Paso 1

Inicializar los pesos de la red con valores pequeños aleatorios.

Paso 2

Presentar un patrón de entrada y especificar la salida deseada que debe generar la red.

Paso 3

Calcular la salida actual de la red. Para ello presentamos las entradas a la red y vamos calculando la salida que presenta cada capa hasta llegar a la capa de salida, ésta será la salida de la red. Los pasos son los siguientes:

Se calculan las entradas netas para las neuronas ocultas procedentes de las neuronas de entrada. Para una neurona j oculta:

$nethpj = \Sigma\ whjixpi + \theta hj$ -> con i = 1 a N

en donde el índice h se refiere a magnitudes de la capa oculta; el subíndice p, al p-ésimo vector de entrenamiento, y j a la j-ésima neurona oculta. El término θ puede ser opcional, pues actúa como una entrada más.

Se calculan las salidas de las neuronas ocultas: $ypj = fhj(nethpj)$.

Se realizan los mismos cálculos para obtener las salidas de las neuronas de salida:

$net0pk = \Sigma wokjypj + wok$

$ypk = fok(netopk)$

Paso 4

Calcular los términos de error para todas las neuronas.

Si la neurona k es una neurona de la capa de salida, el valor de delta es:

$\delta 0pk = (dpk - ypk).fok'(netopk)$

La función f debe ser derivable. En general disponemos de dos formas de función de salida:

La función lineal : fk(netjk) = netjk

La función sigmoidal : fk(netjk) = $1/1+e^{-net_{jk}}$

La selección de la función depende de la forma que se decida representar la salida: si se desea que las neuronas de salida sean binarias, se utiliza la función sigmoidal, en otros casos, la lineal.

Para una función lineal, tenemos: $f^{o}_{k}{}' = 1$, mientras que la derivada de una función sigmoidal es: $f^{o}_{k}{}' = f^{o}_{k}(1 - f^{o}_{k}) = ypk(1- ypk)$ por lo que los términos de error para las neuronas de salida quedan:

δ^{0}_{pk} = (dpk - ypk) para la salida lineal.

δ^{0}_{pk} = (dpk - ypk). ypk(1 - ypk) para la salida sigmoidal.

Si la neurona j no es de salida, entonces la derivada parcial del error no puede ser evaluada directamente, por tanto se obtiene el desarrollo a partir de valores que son conocidos y otros que pueden ser evaluados.

La expresión obtenida en este caso es: $\delta^{h}_{pk} = f^{h}_{j}{}'(net^{h}_{pj}) \sum_{k} \delta^{0}_{pk} w^{0}_{kj}$ donde observamos que el error en las capas ocultas depende de todos los términos de error de la capa de salida. De aquí surge el término propagación hacia atrás.

Paso 5

Actualización de los pesos: para ello utilizamos un algoritmo recursivo, comenzando por las neuronas de salida y trabajando hacia atrás hasta llegar a la capa de entrada, ajustando los pesos de la siguiente forma:

Para los pesos de las neuronas de la capa de salida:

$\Delta wij = yi.yj$

$\Delta w^{0}_{kj} = \alpha \delta^{0}_{pk} ypj$

Para los pesos de las neuronas de la capa oculta:

$w^{h}_{ji}(t+1) = w^{h}_{ji}(t) + \Delta w^{h}_{ji}(t+1)$

$\Delta w^{h}_{ji}(t+1) = \alpha \delta^{h}_{pj} xpi$

En ambos casos, para acelerar el proceso de aprendizaje se puede añadir un término momento.

Paso 6

El proceso se repite hasta que el término de error Ep = $\frac{1}{2}\sum_{k=1}^{M} \delta^{2}_{pk}$ resulta aceptablemente pequeño para cada uno de los patrones aprendidos.

1.2.5 Consideraciones sobre el algoritmo de aprendizaje

El algoritmo encuentra un valor mínimo de error (local o global) mediante una aplicación de pasos (gradiente) descendentes.

Uno de los problemas del algoritmo es que en busca de minimizar la función de error, puede caer en un mínimo local o en algún punto estacionario, con lo cual no se llega a encontrar el mínimo global de la función de error. Sin embargo, no tiene porqué alcanzarse el mínimo global en todas las aplicaciones, sino que puede ser suficiente con un error mínimo preestablecido.

1.2.6 Control de Convergencia

En las técnicas de gradiente decreciente es conveniente avanzar por la superficie del error con incrementos de pesos pequeños. Esto se debe a que tenemos una información local de la superficie y no se sabe lo lejos o cerca que se está del punto mínimo. Con incrementos grandes se corre el riesgo de pasar por encima del punto mínimo sin conseguir estacionarse en él. Con incrementos pequeños, aunque se tarda más en llegar, se evita que ocurra esto.

El incremento del paso adecuado influye en la velocidad de convergencia del algoritmo. La velocidad se controla con la tasa de aprendizaje α. Normalmente α, debe ser un número pequeño (del orden de 0,05 a 0,25), para asegurar que la red llegue a asentarse en una solución.

Lo habitual es aumentar el valor de α a medida que disminuye el error de la red durante la fase de aprendizaje. Así aceleramos la convergencia aunque sin llegar nunca a valores de α demasiado grandes, que hicieran que la red oscilase alejándose del mínimo.

En la práctica, si una red deja de aprender antes de llegar a una solución aceptable, se realiza un cambio en el número de neuronas ocultas o en los parámetros de aprendizaje, o simplemente, se vuelve a empezar con un conjunto distinto de pesos originales y se suele resolver el problema.

1.2.7 Dimensionamiento de la red. Número de neuronas ocultas

No se pueden dar reglas concretas para determinar el número de neuronas o número de capas de una red para resolver un problema concreto.

El tamaño de las capas, tanto de entrada como de salida, suelen venir determinado por la naturaleza de la aplicación. En cambio, decidir cuántas neuronas debe tener una capa oculta no suele ser tan evidente.

El número de neuronas ocultas interviene en la eficiencia de aprendizaje y de generalización de la red. No hay ninguna regla que indique el número óptimo, en cada problema se debe ensayar.

2.1 Propuesta de Implementación de una Red Neuronal

El desarrollo del software propuesto por la cátedra, consiste en nuestro caso en resolver acorde un modelo de cliente si una persona con intención de adquirir un crédito personal es apta, no apta o se encuentra en una situación dudosa para adquirirlo.

Para dicho desarrollo se requiere la implementación de una red neuronal, debido a que no existen relaciones directas entre cada posible configuración del modelo de cliente y su correspondiente perfil (Apto, No Apto, Dudoso), sino mas bien, existen ejemplos de distintos casos en los que a determinados clientes se le han otorgado o no créditos, o bien se requirió otro tipo de análisis mas subjetivo.

El hecho de que no existan relaciones directas entre cada posible configuración del modelo de cliente y su correspondiente perfil se da porque es imposible tener un ejemplo de cada configuración por que son demasiadas o porque pueden aparecer casos especiales de configuración que no fueron tenidos en cuenta por ser muy raros.

El modelo de cliente propuesto y utilizado en la practica por la Empresa VISAN S.A. para el otorgamiento de créditos personales, y sobre el cual se basa todo nuestro análisis se detalla a continuación. Dicho modelo parte de la base en que el cliente cumple con los requisitos esenciales y restrictivos impuestos por la financiera (ejemplo: Edad mínima y máxima, ingreso y/o garante, etc.).

Características	***Descripción***
Edad	Importante para evaluar el resto de las características según la cantidad de años de la persona.
Nacionalidad	Característica necesaria a la hora de evaluar la historia de un cliente.
Estado Civil	Refleja el grado de responsabilidades a cargo y cultura del cliente.
Personas a Cargo	Implica el compromiso de un egreso patrimonial del cliente.
Situación Laboral	Importante para determinar el grado en el que el

	cliente puede responder a la cancelación del crédito.
Años de Antigüedad	Refleja la solides del desempeño laboral del cliente, respecto a la edad actual.
Situación en el VERAZ	Refleja el grado de endeudamiento y compromisos financieros con otras empresas.
Ingreso Salarial	Importante para determinar la medida en la que el cliente puede cancelar la cuota con un ingreso mensual propio, y el grado de dependencia hacia un garante.
Inmuebles	Importante para ver el grado en que el cliente puede cancelar la deuda ante alguna irregularidad en cumplimiento del pago del crédito.
Muebles	Ídem Inmuebles, con la diferencia que éste último posee mayor liquides y es más utilizado.
Créditos Cancelados	Refleja la historia financiera y la responsabilidad con la cancelación de los créditos.
Créditos Vigentes	Importante para determinar el grado de compromisos financieros respecto a las características anteriores.

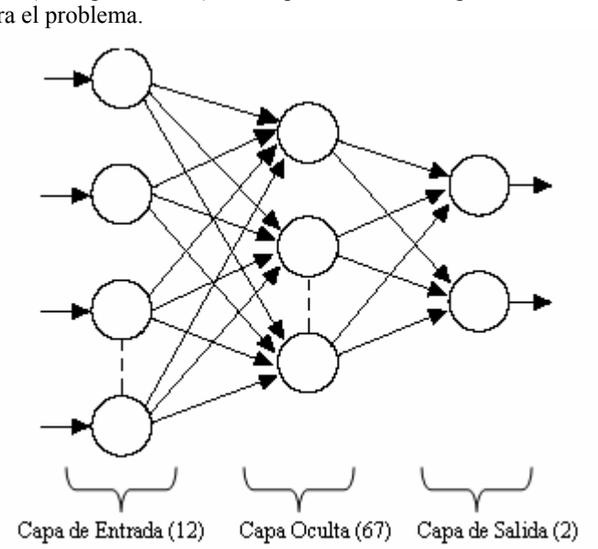
Modelo de Cliente utilizado en VISAN s.a. – Fuente: Silvia Inés Vicentín (*)

Implementación de una Red Neuronal mediante Hopfield

La primera propuesta de implementación del sistema se realiza a través del modelo de Hopfield. El modelo de cliente definido por la Empresa Financiera privada, esta determinado de tal manera que cada característica del modelo se corresponda con una neurona en Hopfield (46 neuronas).

Implementación de una Red Neuronal mediante Backpropagation (BPN)

La segunda propuesta de implementación del sistema es a través de la Red de Propagación hacia Atrás (BPN). El modelo de cliente definido por la Empresa Financiera privada, esta definido de tal manera que cada característica se corresponde con una neurona de entrada (12 neuronas), la salida deseada correspondiente a cada patrón de entrenamiento se corresponde con dos neuronas de salida (10 – apto, 01 – no apto, 11 – dudoso), luego la cantidad de neuronas de la capa oculta fue definida mediante prueba y error (67 capas ocultas) hasta que la red convergió a un umbral relativamente aceptable para el problema.



2.2 Arquitectura y herramienta de desarrollo

Las diferentes implementaciones realizadas fueron elaboradas con el lenguaje C# con Framework 2.0 (Visual Studio 2005).

La arquitectura de las implementaciones, en lo que a programación se refiere, es de 3 capas: presentación (interfaz), lógica de negocios (algoritmos implementados), capa de persistencia (se persistió la base de entrenamiento para BPN y las matrices de peso).

Por otro lado, para las operaciones de matrices se utilizó una librería que encontramos en Internet, la cual es de acceso libre [4].

2.3 Problemas encontrados / soluciones encontradas

Obtención de los perfiles y de los patrones de entrenamiento: La recolección de la información necesaria a cerca de las características que debíamos tener en cuenta para realizar el trabajo práctico se nos complicó bastante debido a que en los bancos, las personas encargadas de estos temas se excusaban por no tener tiempo para atendernos y nos pedían que volviéramos días mas tarde. Concurrimos 3 veces al Banco del Chaco y 2 veces al Banco Bisel hasta que pudimos lograr que nos atendiera la persona que necesitábamos en cada banco. Cuando por fin pudimos entrevistar a las personas encargadas de los créditos se mostraron reacios a brindarnos la información que necesitábamos aduciendo que no les era posible hacerlo. La única información que nos brindaron que pudimos utilizar fue con respecto al Veraz y a los montos de sueldo mínimos que cada banco tenía en cuenta para dar un crédito.

También concurrimos a varias financieras donde obtuvimos igual suerte que en los bancos, hasta que por fin en la financiera VISAN S.A. pudimos entrevistar a la persona Responsable de Finanzas, Srta. Vicentin Silvia Inés quien nos dio la información que necesitábamos, y que luego utilizamos para armar el modelo de clientes que estamos presentando.

La Financiera VISAN S.A. brinda créditos prendarios, ….. y personales. Nosotros tuvimos en cuenta solo los datos con respecto a créditos personales.

Luego de que teníamos listo el modelos de cliente debíamos obtener los patrones de entrenamiento, lo que fue aún mas difícil de conseguir, debido a que la persona que nos brindo la información tenia mucho trabajo y no podía armarnos la cantidad de ejemplos necesarios para poder entrenar la red.

Así volvimos a concurrir a bancos y financieras y ya presentando nuestro modelo de clientes armado pedíamos ejemplos de configuraciones, según nuestro modelo, para casos en los que se podía otorgar el crédito, para casos en los que no y para casos en que se debía realizar un estudio mas minucioso de la situación del solicitante. Si bien los datos no eran muy precisos, ya que mas bien nos dijeron “con veraz 3 para arriba no se da crédito”, “con veraz 2 se debe estudiar la situación según los otros datos del solicitante”, “con sueldo menor a $450 no se da crédito”. De igual manera, con estos datos más los ejemplos obtenidos por la persona contactada en VISAN S.A. pudimos armar patrones realistas y así completamos el número de patrones necesarios para entrenar las redes.

Implementación del modelo de cliente: luego de tener las características que utiliza la financiera al evaluar si proporciona o no el crédito a un cliente debíamos llevar esos datos a una forma de presentación que sea consistente entre cada uno de los patrones de ejemplo para el entrenamiento de la BPN. El principal inconveniente fue encontrar la mejor forma de presentar los datos de manera de evitar grandes oscilaciones de los valores entre un patrón y otro en determinadas características (Ej.: patrón1=…N_i = 48000,……; patrón 2 = …….N_i = 0).

La solución implementada fue representar mediante índices para características de ese tipo.

<u>Características proporcionadas por la Financiera:</u>

Inmuebles = 48000 (representa el valor del inmueble utilizado para evaluar la situación económica del cliente).

<u>Característica implementada por nosotros:</u>

Inmuebles = 48 (representa la cantidad de veces el monto del crédito solicitado, representa lo mismo que antes para la toma de decisiones).

Con Hopfield surgió el problema en la implementación a la hora de desglosar las características en cada uno de los patrones de entrenamiento para obtener la suficiente robustez a la hora que la red deba inferir o clasificar los patrones de entrada.

Ortogonalidad: el dominio del problema hizo que fuera muy difícil cumplir con la ortogonalidad en la red de Hopfield, por lo tanto las diferencias de configuraciones entre un solicitante y otro que debían tener a la postre distintos perfiles (Apto, No Apto, Deseado) a veces eran muy idénticas entre ellas. Esto hacía muy difícil el aprendizaje correcto de la red, ya que no había la diferencia necesaria entre entradas con distintos perfiles de salida. Debido a ello a la hora de su funcionamiento muchas veces asociaba

determinadas entradas con perfiles incorrectos. En nuestro caso particular, mas allá de que podíamos introducir cinco patrones de entrenamiento, estuvimos sujetos a representar solo tres como consecuencia de respetar la ortogonalidad.

Por ello hubo que replantear varias veces la presentación de los patrones de entrenamiento hasta que encontrar los que mejor entrenarían a la red.

Número de neuronas ocultas y valores para los factores (factor de momento y de aprendizaje): como ya se sabe no existe una forma de saber de antemano el número correcto de neuronas ocultas necesarias para el funcionamiento correcto de la red así como tampoco el valor que deberán tomar los factores. Este problema surgió obviamente en la implementación de la BPN.

Al principio sólo buscábamos el número correcto de neuronas ocultas pero los errores causados por los patrones con que entrenábamos la red oscilaban muchísimo y de esa manera era imposible que la red aprendiera. En ese momento nos dimos cuenta que debíamos utilizar los factores de momento y de aprendizaje si queríamos que en algún momento de esta vida funcionara la BPN.

Ahora, de igual manera surgía un nuevo problema, ya no era sólo el número correcto de neuronas ocultas lo que debíamos encontrar sino que también el valor de los factores.

De ahí en más comenzó un arduo trabajo de búsqueda por prueba y error, que consumió la mayor parte del tiempo de realización del trabajo práctico; ya que por cada número de neuronas con las que fue probada la red, al no convergir la misma, debíamos ir variando los factores hasta saber con exactitud que lo que necesitábamos era agregarle más capas ocultas a la red. Al agregar capas ocultas repetíamos la maniobra. Luego en la enésima prueba y luego de un par de semanas de buscar los valores correctos se oyó el "aprendió, parece que por fin aprendió" y efectivamente por fin habíamos dado con el número que nos resultó el mejor para el correcto aprendizaje y funcionamiento de la red.

2.4 Conclusión

Ante este dominio del problema, luego del análisis y la implementación del mismo tanto en Hopfield como en BPN y de las largas pruebas hechas con ambos algoritmos concluimos que:

La implementación de una red Hopfield no es adecuada para este dominio de problema debido a que es muy difícil cumplir con la ortogonalidad si queremos que la red aprenda los suficientes patrones para que en su funcionamiento infiera correctamente las salidas ante cada entrada. Además, aunque se hubiera podido obtener la cantidad de patrones con la ortogonalidad necesaria, el número de neuronas correspondientes a nuestro modelo de cliente limita el número de patrones para el entrenamiento correcto, para una posterior recuperación perfecta de la red en la etapa de funcionamiento. La ventaja que encontramos, pero que de todas maneras no nos sirvió en nuestro escenario, es que es un algoritmo muy simple de implementar. En otras situaciones este tipo de algoritmo podría ser el adecuado: reconocimiento de imágenes y de voz, el control de motores, resolución de problemas de optimización [1].

En contraposición el algoritmo BPN nos pareció lo suficientemente adecuado para este tipo de escenarios. No obstante tiene la gran desventaja de que no hay otra manera de obtener los datos que no sea por prueba y error lo que, como ya se dijo consume un tiempo considerable.

(*) EMPRESA VISAN S.A:
APELLIDO Y NOMBRE: Vicentín Silvia Inés
DNI: 26.508.011
DOMICILIO: Río Negro 394 - Goya (Ctes.)
CARGO: Responsable de Finanzas
CEL: 03777-15534937
EMPRESA: Horacio Joaquín Vilas - VISAN S.A.
ACTIVIDAD: Financiera de Créditos Personales
DOMICILIO: José E.Gómez 105 - Goya (Ctes.)

Bibliografía

[1]- Material de lectura de la cátedra
[2]- http://www.monografias.com/trabajos12/redneuro/redneuro.shtml
[3]- http://www.monografias.com/trabajos12/redneuro/redneuro2.shtml
[4]- http://www.codeproject.com/csharp/matrix.asp