Predicción del rendimiento de los estudiantes y diagnóstico usando redes neuronales

Zulma Cataldi, Fernando Salgueiro, Fernando Javier Lage

Universidad Tecnológica Nacional. Facultad Regional Buenos Aires. Dirección de Postgrado.
Medrano 951. (C1179AAQ) Cdad. Autónoma de Bs. As.
LIEMA: Laboratorio de Informática Educativa y Medios Audiovisuales. Facultad de Ingeniería. UBA.
Av. Paseo Colón 850, Piso 4º (C1063ACV) Cdad. Autónoma de Bs. As.
liema@fi.uba.ar, fsalgueiro@fi.uba.ar flage@fi.uba.ar

Resumen

Según se expuso en publicaciones previas se ha observado el bajo rendimiento de los estudiantes que comienzan una carrera universitaria en sus evaluaciones parciales y finales. Esta investigación surgió para encontrar una solución a tal problema. Se toman los datos de las evaluaciones parciales y a partir del análisis de los mismos se busca efectuar un diagnóstico con base en los errores cometidos. De este modo, en función de los datos de los primeros exámenes se podrán predecir errores futuros y dar sugerencias para realizar una ejercitación correctiva a fin de mejorar la producción. Para efectuar la predicción de los próximos errores se usa una red neuronal y sobre esta base se le sugiere al estudiante una secuencia de ejercicios y problemas a fin de mejorar su producción y por lo tanto sus calificaciones.

Palabras Clave -- Sistemas Tutores Inteligentes, Redes neuronales, evaluación y predicción.

1. Introducción

En publicaciones previas se ha señalado, el bajo rendimiento de los estudiantes de Algoritmos y Programación I en sus exámenes parciales y finales durante los últimos seis cuatrimestres [1]. Este trabajo de investigación surge con fines prácticos tratando de encontrar una solución a este problema a través de la predicción tomando como datos los resultados en las evaluaciones y su análisis para poder efectuar luego un diagnóstico, a fin de sugerir estrategias de refuerzo, ya que se piensa que un sistema asesor inteligente que podría efectuar la predicción y asistir a los estudiantes en su evolución [2].

En la actualidad no existen sistemas expertos que resuelvan en forma eficiente éste problema, ya que sólo se encuentran disponibles algunos programas para enseñanza de programación, pero de tipo tutorial [3]-[4].

2. Justificación del tema

Un sistema así planteado resolvería el problema de la predicción del comportamiento de los alumnos en el marco universitario. Es decir; a partir de los primeros exámenes se podrían predecir posibles errores futuros a través de diagnóstico, pudiendo sugerir la ejercitación correctiva a fin de mejorar su rendimiento y encauzar su aprendizaje hacia conceptualizaciones incorporadas de forma más significativa y permanente [5].

Se toman los errores cometidos en cada uno de los exámenes como datos de partida, se cargan en una base de datos y permiten efectuar una primera categorización en grandes grupos.

A partir de estos datos el sistema que se propone deberá ser capaz de predecir las próximas fallas de un alumno y en consecuencia de ello deberá determinar la siguiente secuencia de problemas y/o ejercicios que el estudiante debería realizar para mejorar su rendimiento y por ende sus calificaciones [6].

El problema del diagnóstico y la predicción incluye:

- una primera etapa de diagnóstico, que está determinada por las respuestas dadas por los alumnos en sus exámenes parciales y finales,
- una segunda parte basada en la predicción de los errores futuros y en la determinación de la ejercitación que deberá resolver un alumno a fin de mejorar su rendimiento.

Este problema está relacionado directamente con los contenidos de la asignatura y además debe considerar, desde el punto de vista didáctico, las acciones del tutor y del estudiante que conforman el triángulo didáctico, en tanto es un proceso comunicacional [4].

Un sistema asesor de este tipo debe plantearse a través de la aplicación de los sistemas inteligentes [7]-[8]-[9]. En general estos sistemas suelen adoptar la forma de tutoriales en los que el estudiante puede tomar la iniciativa residiendo las diferencias más notables con respecto a los programas tutoriales convencionales en su diseño. Un programa tutorial tradicional trata de inducir en el estudiante la respuesta correcta mediante una serie de estímulos que han sido cuidadosamente planificados. En cambio un programa de este tipo, intenta predecir comportamientos futuros sugiriendo líneas de acción, es decir, debe tomar alguna decisión pedagógica [9]-[10].

Existe un *modelo de dominio* donde se encuentra el conocimiento sobre el dominio que deberá ser recomendado, que se puede dividir en: declarativo (los primeros principios, la comprensión del dominio) y procedural (el conocimiento que es utilizado para realizar una tarea) [11]. Utiliza los mismos métodos que se usan en la construcción de la base del conocimiento de los sistemas basados en el conocimiento tales como: las reglas de producción, las redes semánticas, los *frames*, etc. Este conocimiento del dominio consiste en los hechos y en las relaciones entre los hechos que, generalmente debe ser fortalecido en general por uno o más especialistas [12]-[13].

Este módulo, tiene algunas funciones básicas, ya que sirve como fuente de conocimiento a ser presentado al estudiante (lo que incluye la generación del material, generación de preguntas y respuestas, entre otras cosas) y forma un patrón que permitirá evaluar el conocimiento del estudiante. Para eso, el sistema debe ser capaz de generar soluciones a los problemas en el mismo contexto de un estudiante, para que sus respectivas respuestas puedan ser evaluadas. La base del conocimiento del dominio es un componente clave en el *sistema predictor,* ya que es ahí donde está representado el material de enseñanza [14]-[15].

En los casos en que el dominio sea de naturaleza descriptiva y teórica (como en geografía ó física), la representación utilizada es la declarativa (a través de redes semánticas ó *"frames"*). En los casos en que el dominio esté orientado a una tarea (por ejemplo: la programación) la representación tiende a ser procedural (ya que son típicamente reglas de producción) [16]. El modelo del dominio es un tema que ha sido estudiado, así como el modo en que el sistema lo puede usar para razonar [15]-[16]. Algunas de las representaciones posibles son las redes semánticas, reglas de producción y "*constraints*" [15]-[16]. La elección depende en parte de como se lo usará y es común a todos los usuarios del sistema.

A fin de dar solución a la problemática planteada de predicción y diagnóstico, se indagará sobre la aplicación de sistemas inteligentes tales como las redes neuronales que han dado buenos resultados en diversas áreas [17].

3. Las redes neuronales

Las redes neuronales (RN) son conjunto de elementos más simples que se interconectan en paralelo en forma jerárquica y que interactúan como los sistemas neuronales psicológicos [18]. A fin de poder utilizarlas para representar sistemas de mayor complejidad pueden tener retroalimentación. Una de sus características diferenciales es que pueden aprender de la experiencia a través de la generalización de casos [19].

Una red neuronal se caracteriza por cuatro elementos básicos: *su topología, el mecanismo de aprendizaje, tipo de asociación realizada entre la información de entrada y salida y la forma de representación de estas informaciones.*

Las neuronas se distribuyen en la red formando capas de un número determinado de elementos básico. Es decir, existe una capa de: *entrada* que recibe directamente la información proveniente de las fuentes externas de la red, capas *ocultas* que son internas a la red y no tienen contacto directo con el exterior (desde cero niveles hasta un número elevado), pudiendo estar interconectadas de distintas maneras, lo que determina junto a su número, las distintas topologías y una capa de *salida* que transfiere la información de la red hacia el exterior.

La topología de las redes neuronales es la forma de organización de las neuronas en la red formando capas o agrupaciones de neuronas más ó menos alejadas de la entrada y la salida de la red. Por lo tanto, los parámetros fundamentales de la red serán: el número de capas, el número de neuronas por capa, el grado de conectividad y el tipo de conexiones ente neuronas.

A1. El algoritmo backpropagation

Rumelhart, Hinton y Williams [20] desarrollaron un método de aprendizaje automático que permitió que una red neuronal basada en el

perceptron [21] aprendiera la asociación existente entre los patrones de entrada y las clases correspondientes de salidas. Como se buscaban formas de *"Aprendizaje Automático"* o "*Machine Learning*" (se puede definir como un conjunto de programas computacionales que mejoran con la experiencia), estos sistemas deben ser capaces de adquirir conocimientos de alto nivel para la resolución de problemas mediante ejemplos provistos por un instructor ó supervisor debiendo generar representaciones internas de los conceptos. Para lograr esto se modificó la red del *perceptron* de Rosenblatt [21] agregándole capas ocultas, con conexión hacia adelante y sin conexiones recurrentes [20]. Pero, no fue suficiente con introducir algunas modificaciones topológicas a la red, sino que se requerían modificaciones en el algoritmo de aprendizaje; por lo tanto fue desarrollado el método de aprendizaje no supervisado denominado también backpropagation, basado en la regla *delta generalizada* [20], logrando así, una ampliación del rango de aplicación de las redes neuronales.

El funcionamiento general de una red neuronal artificial del tipo *backpropagation*, así como el de otras redes neuronales, se puede dividir en dos partes: una etapa de entrenamiento y una etapa de puesta en marcha. La primera consiste en el aprendizaje de un conjunto predefinido de observaciones de entrada–salida dados como ejemplo (utilizando *n* atributos de entrada y un único atributo ó clase, de salida), empleando un ciclo propagación-adaptación de dos fases [17]-[22], donde:

En la *primera fase* se aplican los atributos de entrada a la capa de entrada de datos a la red y los valores generados se propagan desde esta capa hacia las superiores hasta generar una salida, en la capa de salida de la red. Para realizar el entrenamiento, se compara el resultado obtenido en cada neurona de salida con el valor deseado para cada neurona en particular y obteniéndose un error para cada una de las unidades de salida.

En la *segunda fase*, los errores de las unidades de salida se transmiten hacia atrás, pasando por todas las neuronas de las capas intermedias que contribuyan directamente a la salida, recibiendo el porcentaje de error aproximado a la participación de las neuronas intermedias en la salida original. Este proceso se repite capa por capa hasta llegar a la capa de entrada y hasta que cada neurona haya recibido un error que describa su aporte al error total. Debido a ello el algoritmo se denomina también de *retro-propagación* o propagación hacia atrás, donde los errores se calculan con respecto a los aportes de las neuronas desde la capa de salida hasta la capa de entrada y es con respecto al valor del error recibido que se reajustan los pesos de las conexiones entre cada par de neuronas en la red, de manera de que el error total cometido para ese patrón disminuya.

Dado que la fase de funcionamiento es similar a otras redes neuronales artificiales se debe realizar un análisis más profundo del método de aprendizaje. El método de *backpropagation* utiliza una función o superficie de error asociada a la red, buscando el estado de mínimo error estable a través del camino descendente de la superficie de error [20]. Es por esto que se debe realizar la retroalimentación para realizar las modificaciones en los pesos iniciales en un valor proporcional al gradiente decreciente de dicha función de error. En la Figura 1 se puede ver un esquema de este tipo de redes neuronales artificiales.

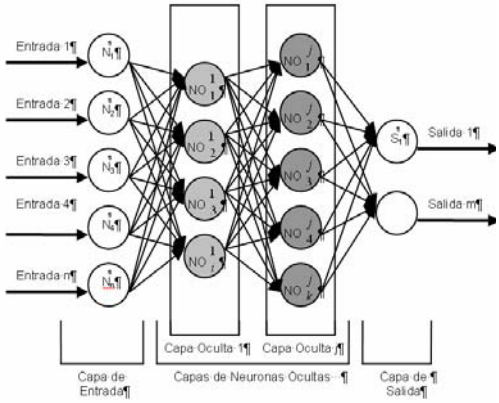


Fig.1: Modelo backpropagation de red neuronal artificial Es una red multicapa, con conexiones hacia adelante y sin conexiones recurrentes [20].

1) La etapa de funcionamiento

En esta etapa se ingresa un patrón p de entrada X_p: $x_{p1},...,x_{pi},...,x_{pN}$, que se transmite a través de los pesos w_{ji} desde la capa de entrada hacia la capa oculta. Las neuronas de esta capa intermedia transforman las señales por aplicación de una *función de activación* un valor de salida que se transmite a través de los pesos v_{kj} hacia la capa de salida. Repitiendo la misma operación que en el caso anterior, las neuronas de la última capa dan la salida de la red.

La función de activación utilizada en cada una de las neuronas debe ser derivable de primer orden.

Algunos autores como del Brio y Sanz Molina, [17] sostienen que el mejor procedimiento para entrenar una red neuronal es el *cross validation,* es decir entrenar y validar a la vez, usando el ochenta por ciento de los patrones para entrenar y el veinte restante como conjunto de pruebas.

2). La etapa de aprendizaje

En la etapa de aprendizaje, se busca minimizar el error entre la salida obtenida por la red y la salida deseada luego del entrenamiento con el conjunto de datos patrones. Es por ello, que en las redes *backpropagation* el aprendizaje es de tipo supervisado, ya que es el usuario (o supervisor) quien determina cuál es la salida deseada ante la presentación de un patrón de entrada dado

Para modificar los pesos se sigue la fundamentación matemática del algoritmo *backpropagation* basado la técnica del *gradiente decreciente* [20]. Suponiendo una red formada por dos pesos, se puede visualizar como un espacio de dos dimensiones y como el error cometido es función de los pesos de la red; en este caso, para cualquier combinación de valores de los dos pesos, le corresponderá un valor de error para el conjunto de entrenamiento donde estos valores de error se pueden visualizar a través de una superficie del error que puede diferentes topografías (especie de sábana). El proceso de entrenamiento comienza, en este caso un punto de la sábana, representado por los pesos iniciales de la red y el algoritmo de aprendizaje se centra en obtener la información local de la pendiente de la superficie o sea el gradiente. A partir de esa información se pueden ir modificando los pesos en forma iterativa en forma proporcional a dicha pendiente, a fin de asegurar un descenso a través de la superficie del error hasta alcanzar el mínimo más cercano al punto de partida. Cuanto mayor sea el número de pesos considerados el espacio se va convirtiendo en un plano multidimensional en el que aplicarán los principios mencionados.

El error o valor delta asociado a una neurona oculta j está determinado por la suma de los errores que se cometen en las k neuronas de salida que reciben como entrada la salida de esa neurona oculta j, por *propagación del error hacia atrás*.

3). El sobreajuste

Para obtener una aproximación funcional óptima se deben elegir cuidadosamente las variables a emplear, es decir se trata de incluir en el modelo las variables que realmente predigan la variable dependiente o de salida, pero que no covaríen entre sí [23], debido a que podrían provocar un sobreajuste (*overfitting*) innecesario. Esto sucede cuando el número de parámetros o de pesos de la red resulta excesivo en relación al problema a tratar y al número de patrones de entrenamiento disponibles. El sobreajuste disminuye la capacidad de la red de proporcionar una respuesta correcta ante patrones que no han sido empleados en su entrenamiento. Se entiende por generalización de la red a la capacidad de dar una respuesta correcta ante patrones que no han sido empleados n su entrenamiento [17].

Del Brio y Sanz Molina [17] recomiendan dos formas de bajarlo: un parada temprana (usando *cross validation*) o limitando el tamaño de la arquitectura de la red. La variable cuya eliminación causa el menor decremento en la ejecución de la red es eliminada. Este procedimiento se debe repetir sucesivamente hasta que la eliminación de más variables involucra una disminución sensible en la ejecución del modelo [22].

4. Objetivos del Trabajo

El objetivo general se desglosó en tres objetivos específicos:

– Implementar sistemas inteligentes (redes neuronales) para predecir errores en futuros exámenes y a partir de éstos determinar que clase de problemas o ejercicios debe resolver el alumno para mejorar su rendimiento, en un dado dominio.

– Determinar el tipo de sistema inteligente a utilizar, ya sea una red neuronal o un algoritmo genético que permita determinar en forma más eficiente la asesoría luego del diagnostico realizado.
– Evaluar el sistema utilizando datos estadísticos almacenados y con un grupo de docentes y de estudiantes disponibles.

5. Metodología

1) Se determinó la herramienta inteligente más apropiada para resolver la cuestión de la predicción en el comportamiento del alumno y la determinación de la actividad futura que deberá realizar, entre las disponibles en un dado dominio.

2) Se creó una base de datos con todos los ejercicios tomados en las evaluaciones parciales y finales de una Cátedra de Algoritmos y Programación I, indicando su tipo (ya sea práctico o teórico), su dificultad, su obligatoriedad o no en un examen, su puntaje y su objetivo (es decir a qué se apunta en dicho ejercicio). El tamaño de la muestra fue de 450 alumnos, usándose 2/3 para entrenamiento y 1/3 para pruebas.

3) Se elaboraron las bases para la creación de un sistema predictor que cumpla con los objetivos que se han propuesto en este trabajo. Para este punto se determinarán las herramientas metodológicas más apropiadas que provee la ingeniería de software y los sistemas inteligentes en cuanto a métodos, técnicas y herramientas disponibles.

4) Se realizó una prueba piloto (evaluación interna y externa del software) con un grupo estudiantes y docentes de la cátedra disponibles.

6. Parte Experimental

Las etapas seguidas fueron las que se describen en la Tabla 1.

Se utilizó la herramienta *NNclass* [25]. La misma posee un instructivo que permite efectuar el ingreso de los datos. El valor máximo es de 50 variables de entrada y 40 de ellas pueden de tipo categóricas.

Please make sure that there are no more than 50 neurons in Input Layer.
There should be exactly 1 Output variable - application will treat it as Categorical.
There should be no more than 40 Categorical Variables.

Omit	Cat	Omit	Omit	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat	Cat
Errores	**Sig**	**Carrera**	**Padron**	**Sexo**	**E1**	**E2**	**E3**	**E4**	**E5**	**E6**	**E7**	**E8**	**E9**	**E10**	**E11**	**E12**	**E13**	**E14**	**E15**	**E16**	**E17**
		9	84585	M	0	1	0	1	0	0	1	0	1	0	0	0	0	0	0	1	0
		10	84722	M	0	1	0	0	0	0	0	1	1	1	0	0	0	0	0	1	0
		10	84653	M	0	1	1	1	0	0	0	0	0	0	1	1	0	0	0	1	0
		9	79600	M	1	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	1
		10	84078	M	0	0	1	0	0	0	1	0	1	1	1	0	0	0	0	1	0
		10	84134	M	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
		10	84870	M	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
		10	84620	M	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
		10	85032	F	0	1	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0
		10	85413	M	1	1	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0
		10	84962	F	0	0	0	1	0	0	1	1	0	0	1	0	0	0	0	0	0
		9	84926	M	1	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0
		9	84734	M	1	1	1	1	0	1	1	0	0	0	0	1	0	0	0	0	0
		10	84091	M	0	1	0	1	0	0	1	0	0	0	0	1	0	0	0	0	0
		10	82557	M	0	0	1	0	0	0	0	0	1	0	1	0	0	0	0	1	0
		10	84690	M	0	0	1	1	0	0	0	0	1	1	0	1	0	0	0	0	1
		10	84087	M	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	1
		10	84587	M	0	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	1
		10	84371	M	0	0	1	1	0	0	0	0	0	0	0	1	0	0	0	0	1

Fig. 2: Pantalla ingreso de datos codificados.

A. El ingreso de los datos.

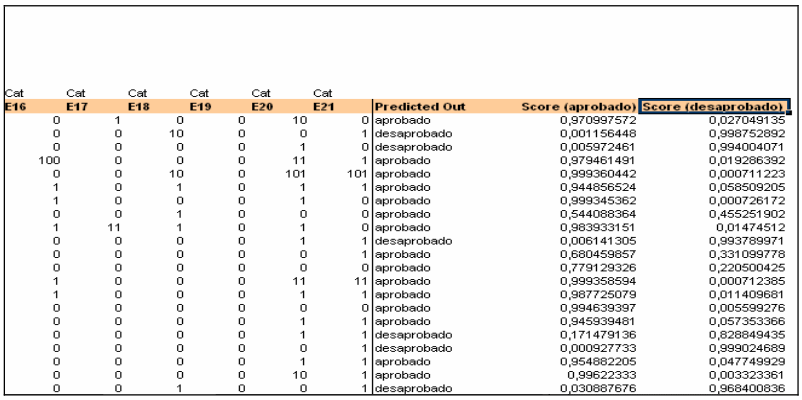
Existen tres tipos de datos que se ingresaron en forma codificada:

– *Categóricos*: son las palabras, que no son la salida esperada de la red.

Se observa que

– *Continuos*: son datos numéricos que representan números reales.

De *Salida (Output):* son los valores que se desea que la red prediga, los que a su vez pueden ser categóricos ó continuos.



Cat E16	Cat E17	Cat E18	Cat E19	Cat E20	Cat E21	Predicted Out	Score (aprobado)	Score (desaprobado)
0	1	0	0	10	0	aprobado	0,970997572	0,027049135
0	0	10	0	0	1	desaprobado	0,001156448	0,998752892
0	0	0	0	1	0	desaprobado	0,005972461	0,994004071
100	0	0	0	11	1	aprobado	0,979461491	0,019286392
0	0	10	0	101	101	aprobado	0,999360442	0,000711223
1	0	1	0	1	1	aprobado	0,944856524	0,058509205
1	0	0	0	1	0	aprobado	0,999345362	0,000726172
0	0	1	0	0	0	aprobado	0,544088364	0,455251902
1	11	1	0	1	0	aprobado	0,983933151	0,01474512
0	0	0	0	1	1	desaprobado	0,006141305	0,993789971
0	0	0	0	0	1	aprobado	0,680459857	0,331099778
0	0	0	0	0	0	aprobado	0,779129326	0,220500425
1	0	0	0	11	11	aprobado	0,999358594	0,000712385
1	0	0	0	1	1	aprobado	0,987725079	0,011409681
0	0	0	0	0	0	aprobado	0,994639397	0,005599276
0	0	0	0	1	1	aprobado	0,945939481	0,057353366
0	0	0	0	1	1	desaprobado	0,171479136	0,828849435
0	0	0	0	0	1	desaprobado	0,000927733	0,999024689
0	0	0	0	1	1	aprobado	0,954882205	0,047749929
0	0	0	0	10	1	aprobado	0,99622333	0,003323361
0	0	1	0	0	1	desaprobado	0,030887676	0,968400836

Fig. 3: Pantalla predicción evaluación final

Paso	Entrada	Acción	Salida
1	Datos de los errores de los alumnos de la materia en exámenes	Confección de base de datos de errores de cometidos	Errores codificados
2	Datos de entrenamiento	Entrenamiento de la red	Parámetros con error mínimo. Red entrenada
3	Datos de prueba	Aplicación de la red	Pronóstico
4	Errores codificados	Aplicación de la red	Diagnóstico

TABLA 1 ETAPAS Y ACCIONES SEGUIDAS

En el caso que se muestra en la Figura 2, se muestra la salida que es de tipo categórica ya que será: *aprobado* ó *desaprobado*.

Predicted Output	Score (aprobado)	Score (desaprobado)
Aprobado	0,970997572	0,027049135
Desaprobado	0,001156448	0,998752892

TABLA 2 EJEMPLO DE PREDICCIÓN APROBADO–DESAPROBADO

Por lo tanto, la red utilizará dos neuronas para ello; una cuya salida, a través de un número real, representa a los *aprobados* y otra, que también devuelve un número real, cuya salida está asociada a los *desaprobados.* En la primera fila del ejemplo de la Tabla 2 se observa que la para la salida de 0.970 la neurona está asociada a *"aprobado"* y sólo 0.027 la asociada a *"desaprobado",* por lo que la red da como resultado el primero de éstos. En la segunda fila se presenta el caso inverso. Los datos de las evaluaciones de los estudiantes (a través de 6 instancias de aprobación: un parcial con dos recuperatorios y tres oportunidades para el examen final) han sido codificados como se observa en la Tabla 3. El resto son combinaciones de las opciones presentadas.

000	no se equivocó nunca
001	se equivocó en el parcial
010	se equivocó en el primer recuperatorio
100	se equivocó en el segundo recuperatorio

TABLA 3: CODIFICACIÓN DE LAS INSTANCIAS DE APROBACIÓN.

Con respecto a las columnas, E1 a E21 de la Figura 1, son los tipos de errores detectados en cada una de las evaluaciones agrupados luego del análisis realizado.

B. La selección de los parámetros de entrenamiento de la red.

Se recomienda utilizar valores similares a los que están en la Figura 4 repitiendo el proceso cambiando los valores hasta que se encuentre un error mínimo entre 0%-5%.

7. Resultados

En la Figura 4 se observan las características de la red. Cuando el error es inaceptable, mayor al 5%, se debe descartar la red y comenzar el proceso de entrenamiento nuevamente variando los distintos parámetros que la definen (entre los que se pueden citar el parámetro de aprendizaje α (el valor inicial y el régimen de modificación a lo largo de los ciclos), el momentum β, los pesos aleatorios iniciales, el vecindario gaussiano, la cantidad

de atributos que se utilizarán para el entrenamiento, la cantidad de las observaciones que se utilizarán y las que se descartarán, justificando por qué se descartarán.

En promedio, el tiempo de entrenamiento de una red neuronal para los valores utilizados es de aproximadamente 1.5 horas. Se entrenaron redes con más de 600 variaciones en los parámetros hasta encontrar los adecuados y los valores mínimos correspondientes al error, lo que da un tiempo total neto de 38 días de entrenamiento.

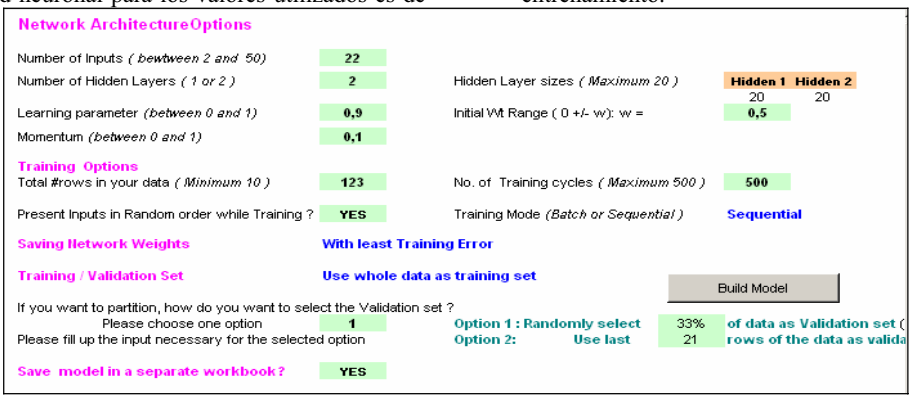


Fig. 4: Características de la red

A. La predicción de la red

Cuando un estudiante se equivoca (o no) en algunos temas, aunque no haya completado aún todo el curso, es decir, si solo rindió el parcial (y no los recuperatorios, por ejemplo) se puede predecir si aprobará o no. La idea es utilizar esta red como primer paso para el uso de una serie de dos redes. Cuando la red prediga que no aprobará, una segunda red, basada en los errores cometidos, le puede indicar qué temas debe estudiar, a fin de recomendarle los ejercicios por núcleos temáticos. Para este primer entrenamiento predictivo la red operó con un error del 4%, lo que es más que aceptable para el trabajo con grupos humanos [9].

8. Grado de Avance

Hasta ahora se trabajó en el sistema de predicción donde se buscó indagar cómo puede diagnosticar la red a partir del rendimiento obtenido en las evaluaciones parciales cómo será la *"performance"* en el final. Este dato permitirá diseñar un sistema recomendador de modo que el alumno en situación de preparar su evaluación final pueda acceder al sistema que le sugerirá una serie de ejercicios y problemas para poder internalizar los errores clave cometidos en las evaluaciones previas.

9. Conclusiones y Trabajos Futuros

Se prevé trabajar en el diseño del sistema recomendador de ejercicios y problemas de modo que el estudiante pueda adquirir una cierta autonomía en la preparación de sus exámenes finales. Se busca brindar a los estudiantes una herramienta a fin de que puedan tomar conciencia de sus propios errores para no cometer las mismas fallas en las evaluaciones finales. Esta forma de *autoevaluación* resulta un acercamiento, hacia la autonomía del alumno y la mejora del proceso de aprendizaje.

Como trabajos posteriores se propone: a) Ampliar los contenidos disponibles para la autoevaluación, b) Escalar el sistema informático de tal forma que permita realizar un seguimiento del alumno, de esta forma el docente puede tener una clusterización de su clase en cuanto a necesidades cognitivas, c) Escalar el sistema informático de tal forma que evolucione hacia bases de datos e interfaces capaces de interactuar con el alumno de manera autónoma y d) incluir un módulo de autoevaluación en los Sistemas Tutores Inteligentes cuya arquitectura se está desarrollando.

Agradecimientos

Esta comunicación forma parte de los proyectos de investigación: *Sistemas inteligentes aplicados a la predicción del comportamiento de los estudiantes y diagnóstico* 2005-2006 LIE-DC/04-07 del Laboratorio de Informática Educativa y Medios Audiovisuales (LIEMA) de la Facultad de Ingeniería, de la Universidad de Buenos Aires y C099 *Modelado del tutor basado en redes*

neuronales para un Sistema Tutor Inteligente, de la Facultad Regional Buenos Aires de la Universidad Tecnológica Nacional 2007-2008. Los autores agradecen a los alumnos que participaron de la experiencia.

Referencias

[1].Lage, F.; Cataldi, Z. y Denazis, J. M. (2000). *The Scripts of University Students and Experts in the Preparation of the Examinations: A Study in Process.* FIE 2000: 30th ASEE/IEEE Frontiers in Education Conference, Kansas City Missouri, 18-21 de octubre. Paper 1154. Proceedings en CD-ROM. ISBN 0-7803-6242/0/

[2].Copello, G.; Cataldi, Z. y Lage, F. (1999). *La comprensión de los errores.* Proceedings del V Congreso Internacional de Ingeniería Informática. Páginas 210-217. Editado por Departamento de Publicaciones de la Facultad de Ingeniería.

[3].Pozo, J. I. (1998). *Teorías cognitivas del aprendizaje*. Morata.

[4].Pozo, J. I. (1999). *Aprendices y Maestros*. Alianza.

[5].Ausubel, D.; Novak, J. y Hanessian, H. (1983) *Psicología educativa: un punto de vista cognitivo.* 2ª Ed. México: Trillas. 624p.

[6].Ohlsson, S. (1996) *Learning from performance of errors.* Psychological Review 3 (2) p. 241-262.

[7].Khuwaja, R.A. (1994) A Model of Tutoring: Facilitating Knowledge Integration Using Multiple Models of the Domain. *Ph.D., Illinois Institute of Technology*

[8].Giraffa, L.M.M.; Nunes, M. A.; Viccari, R.M. (1997) *Multi-Ecological: an Learning Environment using Multi-Agent architecture*. MASTA'97: Multi-Agent System: Theory and Applications. Proceedings. Coimbra: DE-Universidade de Coimbra.

[9]. Cataldi, Z. 2005. *Sistemas tutores inteligentes: los estilos del estudiante para selección del tutorizado.* WICC 2005. 13 y 14 de mayo. Universidad Nacional de Río Cuarto. Córdoba. RED UNCI

[10]. Salgueiro, F. A, Costa, G., Cataldi, Z., García Martinez, R. y Lage, F. J. 2005. *Sistemas inteligentes para el modelado del tutor.* GCETE'2005**,** Global Congress on Engineering and Technology Education. marzo 13-15

[11].Abbas, H. (1998) Designing a New Domain Knowledge Base for an Intelligent Tutoring System. *Ph.D., Illinois Institute of Technology.*

[12].Brachman, R.J. (1988) *The basis of knowledge representation and reasoning.* AT&T. Technical Jurnal. 67, 1:15.

[13].Brachman; R.J. (1985) On the epistemological status of semantic networks. En Brachman, R. y Levesque, H. (Eds.) *Readings in knowledge representation* (191-215). Los altos. Morgan kaufman Pub. Inc.

[14].Viccari, R. M. (1993). Inteligência Artificial e Educação: Indagações Básicas. IV Simpósio Brasileiro de Informática e Educação.

[15].Viccari, R.M. y Girafa, L.M. (1996). Sistemas Tutores Inteligentes: Abordagem Tradicional x Abordagem de Agentes. XIII Simpósio Brasileiro de Inteligência Artificial, Curitiba.

[16].Casas, M. (1999) *contribuições para a modelagem de um ambiente inteligente de educação baseado em realidade virtual.* Tesis Doctoral Universida de Federal de Santa Catarina. Programa de Pós-graduação em Engenharia de Produção.

[17].del Brio, B. M. y Sanz Molina, A. (2001) *Redes neuronales y sistemas difusos.* Paraninfo.

[18].Kohonen, T. (1988) Self-Organizing Maps Springer Series in Information Sciences, Vol. 30, Springer, Berlin, Heidelberg, NY(pp 236)

[19].Hilera González; R. y Martínez Hernando, A. (2000) Redes Neuronales Artificiales: Fundamentos, modelos y aplicaciones. Ra-ma, Madrid.

[20].Rumelhart, D. E.; Hinton, G. E.; Williams, R. J. (1986). *Learning internal representations by back-propagating errors in Parallel Distributed Processing*: *Explorations in the Microstructure of Cognition*. Eds. Cambridge, MA: MIT Press, vol. 1, p. 318-362.

[21].Rosenblatt, F. (1958), *The perceptron: A probabilistic model for information storage and organization in the brain*. Psychological Review, 65, 386-408.

[22].Palmer, A., Montaño, J.J. y Jiménez, R. (2001) *Tutorial sobre Redes Neuronales Artificiales: El Perceptrón Multicapa.* Revista Electrónica de Psicología Vol. 5, No. 2, Julio ISSN 1137-8492.

[23].Smith, M. (1993). *Neural networks for statistical modeling*. New York: Van Nostrand Reinhold.

[24].Masters, T. (1993). *Practical neural networks recipes in C++*. London: Academic Press

[25].Saha, A. (1998) *Application of Ridge Regression for Improved Estimation of Parameters in Compartmental Models;* Tesis Doctoral. Departamento de Estadística;