NEUROX: ASIGNACIÓN DE AULAS MEDIANTE REDES NEURONALES

Ruau, Kevin; Chardon, Andrés; Xodo, Daniel; Bueno, Moisés; Dos Reis, María

INTIA, Universidad Nacional del Centro de la Provincia de Buenos Aires
Paraje Arroyo Seco s/n - CP 7000 – Tandil – Bs. As. – Argentina - daniel.xodo@gmail.com

RESUMEN.

La asignación de aulas en el dictado de clases suele ser un problema de la gestión administrativa donde se deben atender las necesidades de alumnos, docentes y condicionantes físicos existentes.
NeuroX es una herramienta interactiva desarrollada en el marco de trabajo final para la materia Investigación Operativa, que permite establecer una configuración para una posible clase o curso, y mediante el uso de redes neuronales, determinar en qué aula de la facultad debería desarrollarse, maximizando la eficiencia en términos de distintos factores.
La utilización de *redes neuronales* permite aplicar un paradigma de aprendizaje, posibilitando la generación de una serie de casos de prueba que permiten entrenar las neuronas en las decisiones a futuro en consultas sobre asignaciones de aulas.

Palabras Claves: Asignación de aulas – Redes Neuronales – Investigación Operativa.

ABSTRACT

The allocation of classrooms in teaching classes usually a problem of administrative management which should meet the needs of students, teachers and existing physical conditions.
NeuroX is an interactive tool developed as part of final work for the subject Operations Research, which allows for a possible configuration for a class or course, and by using neural networks, determine which classroom faculty should be developed, maximizing efficiency in terms of different factors.
The use of neural networks to apply a learning paradigm, enabling the generation of a series of test cases that allow neurons to train future decisions in consultations on classroom assignments.

1. INTRODUCCIÓN

La asignación de aulas en el dictado de clases suele ser un problema de la gestión administrativa de cada facultad donde se deben atender las necesidades de alumnos, docentes de las diferentes cátedras y condicionantes físicos existentes. A su vez, esta tarea se complejiza aún más cuando se tiene en cuenta la gestión inter-facultades en la distribución y asignación de horarios y aulas para cada facultad.

Este trabajo se desarrolló en base al relevamiento de datos de las aulas disponibles para la facultad de Ciencias Exactas en los pabellones del campus de la UNCPBA, teniendo en cuenta algunas de sus características fundamentales: nombre del aula, capacidad, disponibilidad de proyector, disponibilidad de aire acondicionado y cantidad de computadoras.

El objetivo del trabajo contempla realizar una aplicación que permita determinar en cuál de todas las aulas debería desarrollarse una hipotética clase para maximizar la eficiencia en el uso de los recursos mencionados anteriormente.

Como requerimiento adicional, desde la cátedra se propuso la utilización del paradigma de aprendizaje denominado redes neuronales.

1.1. Reseña metodológica

Las redes neuronales artificiales están inspiradas en la forma en que funciona el sistema nervioso de los animales: ante un estímulo de entrada, un conjunto de neuronas interconectadas colaboran entre sí para producir un estímulo de salida. Utilizando un aprendizaje supervisado, la red neuronal debe ser entrenada con un conjunto de datos de entrada de los que ya se sabe el resultado esperado.

Utilizando la potencia que nos brindan las redes neuronales, es posible generar una serie de casos de prueba, y de esta manera entrenar a las neuronas para que decidan en un futuro a posibles consultas sobre asignaciones de aulas.

La herramienta fue desarrollada en el lenguaje *Java* [1] y utilizando las librerías del framework de código abierto *Neuroph* [2], que proveen una implementación de redes neuronales sencilla y completamente adaptable a las necesidades de este proyecto.

2. DESARROLLO

2.1. Diseño de la red neuronal

El perceptrón multicapa [3] es el modelo elegido para la red neuronal. Es un modelo que toma un determinado número de entradas para generar su salida apropiada. Consiste en múltiples capas de nodos en un grafo dirigido, en donde cada capa está totalmente conectada con la siguiente. Todos los nodos, excepto los nodos de entrada, son "neuronas"[1] (ver Figura 1).

El perceptrón multicapa utiliza una técnica de aprendizaje supervisado denominada *retro propagación*: al recibir las entradas, se propagan por todas las capas hasta generar una salida, la cual es comparada con la salida deseada, y se calcula una señal de error. Ésta se propaga hacia atrás, partiendo desde la capa de salida, hacia todas las neuronas de las capas anteriores que contribuyeron en la generación de la salida. Cada neurona sólo recibe una fracción de la señal total del error, en base a la contribución relativa que haya aportado al error total. Esto se repite capa por capa hasta que todas las neuronas de la red hayan recibido una señal de error que describa su contribución relativa al error total.

A medida que se entrena la red, las neuronas se organizan para reconocer las características de las entradas, y así obtener resultados de salida que difieran en la menor medida posible de las salidas originales propuestas por el caso de testeo. Un problema a tener en cuenta, es que si el perceptrón multicapa no se entrena bien, la salida puede ser imprecisa, es por ello que cuantos más casos de entrada para entrenar a las neuronas sean provistos, más preciso y acertado será el entrenamiento, y consecuentemente el resultado ante posibles consultas a la red neuronal.

[1] Neuronas: elementos encargados de un determinado procesamiento

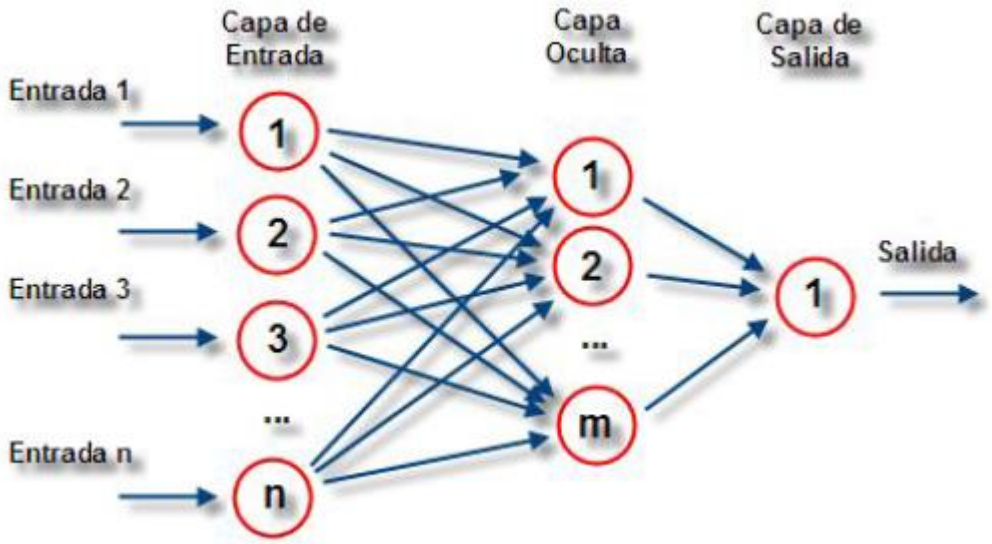

Figura 1 *Esquema de red neuronal*

2.2. Entrenamiento de la red

Para entrenar una red neuronal correctamente, se necesita una gran cantidad de casos de prueba (incluyendo entradas y salidas esperadas; en este caso la salida esperada es la eficiencia). En esta aplicación no fue posible contar con dichos casos de prueba, entonces se optó por generar computacionalmente una entrada que consistiera de una gran cantidad de casos generados al azar. Para ello se idearon las capacidades de las aulas y los requisitos de las clases, y luego se calculó su eficiencia mediante una fórmula estudiada a tal fin.

En post del objetivo de entrenamiento, se creó una aplicación muy sencilla en C++ [4], utilizando el framework multiplataforma Qt [5]. La misma consiste, básicamente, de varias funciones cuyo principal objetivo es la generación y organización de datos aleatorios de tal forma de generar la forma de los datos de entrada; es decir, generar un número al azar para la ocupación del aula (número decimal entre 0 y 1), otro número al azar para la cantidad de las computadoras (número decimal entre 0 y 1) y dos valores booleanos al azar representado la presencia de aire acondicionado y de proyector en las aulas (los valores son representados como números 0 o 1, donde 0 es no contar con la característica y 1 es el caso contrario). Es importante destacar que tanto el valor de ocupación del aula como el valor de cantidad de computadoras son números que deben estar normalizados una vez que se presenten como casos de entrenamiento ante las neuronas, es por eso que oscilan en un rango [0; 1].

Cuando el proceso de generación de datos de entrada termina, están dadas las condiciones para generar la salida. Es decir, es necesario ingresar los datos que permitan evaluar la eficiencia. De la misma manera que en el proceso anterior, los datos fueron generados a partir de una fórmula creada que simula (teóricamente)[2] el comportamiento de la realidad.

La fórmula utilizada para la generación de la salida es la Ecuación (1):

$$Eficiencia = 0{,}5 * A + 0{,}2 * B + 0{,}15 * C + 0{,}15 * D \quad (1)$$

$$A = \frac{CapacidadB}{CapacidadA} \quad (2)$$

$$B = \frac{ComputadorasB}{ComputadorasA} \quad (3)$$

En la Ecuación (2) CapacidadB es la capacidad del aula en evaluación y CapacidadA es la capacidad de aula requerida. En la Ecuación (3) ComputadorasB es la cantidad de computadoras disponibles en el aula ofrecida y ComputadorasA la cantidad requerida.

$$C = ProyectorA\ XNOR\ ProyectorB \quad (4)$$

[2] El desarrollo de este trabajo es teórico, debido a la imposibilidad de disponer con datos completamente tomados de la realidad, por este motivo se generan los casos de entrenamiento de las neuronas.

$$D = AireA\ XNOR\ AireB \quad (5)$$

En las ecuaciones (4) y (5) las letras A y B identifican al recurso requerido/ofrecido, respectivamente. Como se puede apreciar la Ecuación (1) contempla la posibilidad de distintas ponderaciones para cada característica a analizar. Analizando su estructura, podemos decir que un aula solo tendrá una eficiencia completa de utilización en el caso de que la capacidad del aula sea igual o mayor a la requerida, que se cumpla el requerimiento de cantidad de computadoras y que el aula tenga proyector y aire cuando se necesiten. Si alguna de esta cuatro condiciones no se cumple, entonces la eficiencia tendrá un valor menor a uno. Por otra parte en el caso de los dos primeros atributos, si la capacidad disponible supera la capacidad requerida o la cantidad de computadoras disponibles en el aula supera las demandadas, el factor A o B respectivamente se hace uno, para no sobrecargar la ponderación de ese atributo.

Es importante destacar que si bien en la Ecuación (1) no se tienen en cuenta factores negativos (es decir, nunca se resta eficiencia), los mismos son tenidos en cuenta al no sumar eficiencia, lo que genera un efecto similar. Es decir, si un aula tiene no tiene proyector, pero las necesidades del curso lo demandan, no se restará valor a la eficiencia, pero tampoco se sumará, logrando el efecto deseado, ya que el 15% del valor total correspondiente al atributo "proyector" no será sumado.

Una vez generado el valor de salida, se puede generar una gran cantidad de casos de pruebas y volcar los datos en un archivo. Este archivo tiene un formato específico, ya que cada dato debe ser separado por una coma, y los mismos deben tener un orden determinado. Esto se debe a que, posteriormente, esta entrada es analizada automáticamente por *Neuroph*, lo que facilita el parsing de la información de entrada.

Cuando se tiene el archivo que contiene todos los casos de prueba necesarios para entrenar la red neuronal, están dadas las condiciones para utilizar la aplicación central de este proyecto.

Una vez en la aplicación principal, se selecciona el archivo generado con los casos de prueba (también existe un archivo "aulas.txt" que por defecto es generado y utilizado en caso de que no se seleccione archivo de entrenamiento), y mediante la retro propagación, la red neuronal es entrenada aproximándose a la función estimada de eficiencia anteriormente descrita.

Es importante resaltar que la cantidad de entradas de las neuronas es un número fijo de ocho entradas, y la cantidad de salidas es un número fijo también, de una sola salida.

Para un ejemplo cualquiera, se puede comparar el resultado de la eficiencia obtenido al aplicar la fórmula descrita anteriormente con la eficiencia obtenida al ingresarlo en la red neuronal. Si la red está bien entrenada, la diferencia entre ambos es mínima.

3. RESULTADOS

3.1. Selección de aula óptima

Una vez implementada y entrenada correctamente la red neuronal, deben cargarse los datos reales de las aulas de la facultad. Estos datos fueron obtenidos haciendo un relevamiento de un subconjunto del total de aulas del Campus Universitario de la UNICEN, se incluyeron en el análisis las aulas de la Facultad de Ciencias Exactas y las Aulas Comunes.

La aplicación abre un archivo de texto en donde cada una de sus líneas cuenta con el nombre del aula, el pabellón en el que se encuentra, su capacidad máxima, si posee aire acondicionado (AA), si posee proyector incorporado, y la cantidad de computadoras totales. La ventana principal de la aplicación permite crear una hipotética clase y definir sus parámetros.

En la Tabla 1 se puede observar una distribución particular de aulas, cada aula cuenta con un nombre identificatorio, además de información relativa a las características relevantes para este proyecto.

Tabla 1 *Ejemplo de distribución de aulas*

Aula	Capacidad	AA	Proyector	Computadoras
Pabellón 1				
Aula 1	224	No	Sí	No
Aula 2	224	No	No	No
Aula 3	224	Sí	No	No
Pabellón 2				
Aula 1	156	Sí	No	No
Aula 2	156	Sí	No	No
Aula 3	20	Sí	No	20
Aula 4	50	Sí	No	No

Aula 5	16	Sí	No	16
Aula 6	60	Sí	No	No
SUM	130	Sí	No	No
Pabellón 3				
Aula 1	180	Sí	No	No
Aula 2	111	Sí	No	No
Aula 3	111	Sí	No	No
Aula 4	130	Sí	No	No
Aula 5	84	Sí	No	No
Aula 6	97	Sí	No	No
Aula 7	180	Sí	Sí	No
Exactas				
Aula 1	80	Sí	Sí	No
Aula 2	64	Sí	Sí	No
Aula 3	40	Sí	Sí	No
Aula 4	40	Sí	Sí	No
Lab Física 1	25	No	No	No
Lab Física 2	10	No	No	No
Laboratorio 1	44	Sí	Sí	27
Laboratorio 2	12	No	Sí	12

Una vez que se le pide encontrar el aula con la mejor eficiencia para los datos ingresados la aplicación acciona de la siguiente manera:

- En primera instancia se obtiene la información ingresada por el usuario según las características deseadas que tenga el aula. Esta información provee cuatro de las futuras entradas para consultar a las neuronas.
- En segundo lugar, se itera sobre el contenido del archivo de las aulas, buscando las características de cada una (teniendo en cuenta dos condiciones fundamentales: la capacidad del aula y la cantidad de las computadoras deben ser mayores o a lo sumo iguales a los ingresados por el usuario).
- Una vez halladas todas las aulas candidatas, se normalizan los valores de capacidad de aula y cantidad de computadoras (recordemos que los mismos son procesados por las neuronas sólo si tienen valores normalizados entre 0 y 1). La normalización ocurre de forma muy sencilla, simplemente dividiendo el valor a normalizar por el mayor valor encontrado en el archivo de aulas para esa característica.
- Cuando se tienen estos datos, se suman a la entrada cuatro valores más, teniendo en total los ocho valores de entrada necesarios para consultar a la red neuronal por una salida.
- Por último, al disponer de todos los datos y la red neuronal entrenada, se prosigue a hacer una consulta para cada aula. Cada consulta arrojará un valor eficiencia asociado a cada aula, por lo que simplemente la aplicación elige el mayor entre todos los valores, y muestra el resultado al usuario a través de la interfaz gráfica.

En la siguiente Figura se aprecia el funcionamiento de la aplicación, donde se ingresan los parámetros deseados por el usuario. En este caso se pretende encontrar un aula para 54 alumnos, que cuente con proyector para dictar la clase, sin embargo no es menester disponer de aire acondicionado ni tampoco de computadoras. Al presionar el botón “Calcular!” simplemente la aplicación busca el aula más eficiente que se encuentre en su base de datos de aulas que satisfaga los requerimientos ingresados.

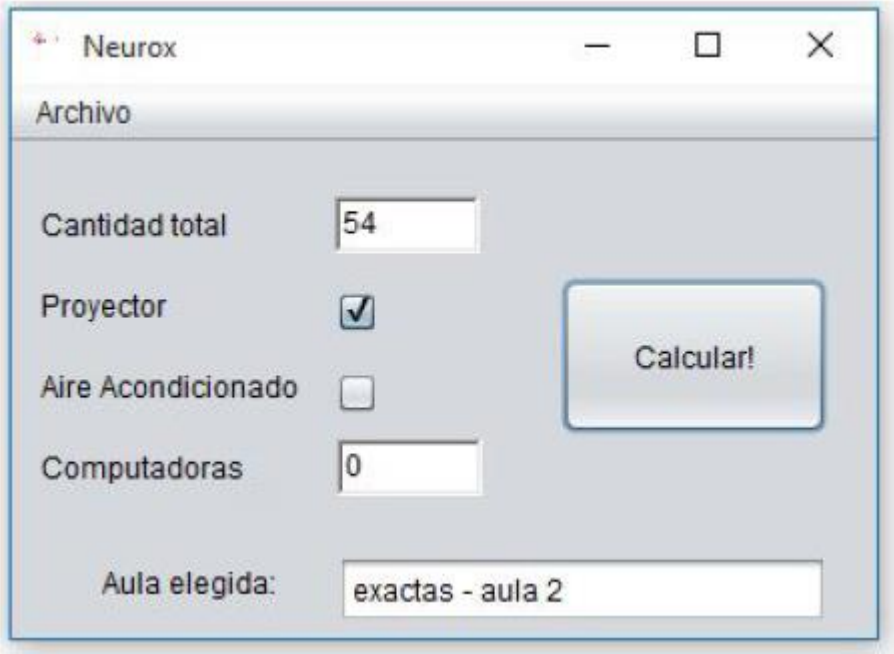

Figura 2 *Interfaz de Programa*

3.2 Variación de resultados

Utilizando la aplicación es posible evidenciar de forma muy clara cuando un aula posee características altamente compatibles con las necesidades ingresadas por el usuario. Sin embargo, existen ciertos casos donde la compatibilidad entre los datos del aula con los datos ingresados por el usuario son similares para varias aulas, o incluso pueden existir aulas con características idénticas.

En los casos donde la semejanza de las aulas es muy estrecha, el resultado de la elección de la aplicación puede variar de ejecución en ejecución. Este efecto tiene una explicación analizando la forma en que trabajan las redes neuronales, y cómo están implementadas las mismas.

Para distintas ejecuciones, los valores de eficiencia final arrojados por la aplicación varían levemente. Es por eso que si la diferencia en la elección es muy marcada, es decir, existe un aula que evidentemente es la mejor, las distintas ejecuciones pueden arrojar variaciones del valor de la eficiencia, pero siempre se va a elegir la misma aula.

En cambio, en los casos en los que no es tan evidente la elección, o las características de las aulas son muy similares, las distintas ejecuciones producen una variación en el valor de salida, y esa variación de los valores de eficiencia puede ser determinante a la hora de elegir la mejor aula.

Esta variación entre ejecución y ejecución se explica mediante la forma en que el **framework** *Neuroph* implementa las redes neuronales. Las mismas son implementadas mediante *threads*, es decir, a cada neurona se la trata como si fuese un hilo de ejecución independiente de las demás neuronas. Estos *threads* se comunican entre sí para lograr el objetivo deseado, que es generar una salida. Sin embargo, la utilización de *threads* hace que los cambios de contexto de ejecución entre *thread* y *thread* dependan de las condiciones particulares de la computadora y la elección del siguiente *thread* a ejecutar. Al no tener una ejecución lineal, sino más bien concurrente (o incluso paralela), existe un factor de azar en el resultado de la salida final. Este factor de azar relacionado con el orden de ejecución de los *threads*, el tiempo de ejecución, la comunicación entre los mismos, etc., hace que el resultado de la ejecución (es decir, el valor de salida) varíe entre distintas ejecuciones.

4. TRABAJOS FUTUROS

Si bien el problema planteado consistió en encontrar un aula óptima para el dictado de una clase, la aplicación puede extenderse fácilmente. Se podría optar por implementar un sistema más complejo de asignación por horarios y fechas, o añadirle mayor cantidad de características a la hora de calcular la eficiencia; por ejemplo si el aula cuenta con calefacción, si tiene suficientes toma corrientes, la distancia entre la clase y el aula de la siguiente clase, etc. Por cada uno de estos atributos habría que agregar dos entradas a la red neuronal, los valores para cada caso de prueba al archivo de texto y sus correspondientes accesos en la interfaz gráfica.

Otro factor a tener en cuenta es que la aplicación permite elegir un archivo de entrenamiento distinto, que puede ser generado externamente por un programa, siempre y cuando respete el mismo

formato. De esta manera se pueden comparar los resultados obtenidos probando distintos entrenamientos fácilmente.

5. CONCLUSIONES

El desarrollo de NeuroX fue satisfactorio y cumplió con los objetivos planteados inicialmente.
El diseño de la herramienta priorizó la extensibilidad de la misma para que en un futuro pueda ser implementada a mayor escala que la original, por ejemplo sumándole atributos o utilizándola en una escuela o universidad.
El proceso de desarrollo de la aplicación permitió el aprendizaje del funcionamiento de las redes neuronales artificiales, paradigma muy utilizado actualmente tanto en investigación como en resolución de problemas complejos. Esta cuestión es de gran relevancia, debido a que tanto la aplicación de conocimientos y nuevas tecnologías como el intercambio docente - alumnos en el proceso de aprendizaje, es fundamental en la formación de los estudiantes para una inserción más sencilla en grupos de investigación, formación de recursos humanos o trabajos especializados en conceptos de gran relevancia a nivel mundial actualmente.

6. REFERENCIAS BIBLIOGRÁFICAS

[1] Java Platform, Standard Edition 7, API Specification.
http://docs.oracle.com/javase/7/docs/api/

[2] Neuroph API documentation.
http://neuroph.sourceforge.net/javadoc/index.html

[3] HAYKIN, S. (1999): “Neural Networks: A Comprehensive Foundation*”. McMaster University, Canada.*

[4] C++ documentation. http://www.cplusplus.com/

[5] QT Framework documentation. http://doc.qt.io/