Glosario técnico

B

Bias: Parámetro de ajuste que permite a las redes neuronales artificiales ubicar correctamente la frontera de decisión en el hiperplano de soluciones. El valor numérico de este elemento puede llegar a ser un entero o decimal, tanto positivo como negativo o incluso cero.

Binario: Sistema de numeración en base 2, con el cual es posible representar cifras, palabras y archivos compuestos por los números 0 y 1, para que puedan ser interpretados por las computadoras.

C

Compilador: Tipo de traductor que permite la transformación de un lenguaje de programación en otro. Normalmente, traduce programas informáticos con lenguajes de alto y medio nivel a código de máquina.

F

Framework: Conjunto de módulos, librerías y rutinas que contienen soluciones generales para ser adaptadas a problemas específicos.

H

Hardware: Conjunto físico interconectado de elementos eléctricos, electrónicos y/o mecánicos que conforman un sistema informático.

L

Linux: Es un conjunto de instrucciones que funcionan como eje central de un sistema operativo de código abierto.

P

Peso sináptico: Parámetro que modifica la amplitud y el sentido de la entrada de una neuronal artificial.

R

Repositorio: Espacio virtual comúnmente centralizado en donde se almacenan programas informáticos para diferentes propósitos.

S

Script: Archivo que contiene instrucciones de un lenguaje de programación interpretado.

Software: Conjunto de instrucciones y rutinas que les indican a las computadoras qué hacer.

T

Terminal: Es una interfaz de línea de comandos con la cual un usuario puede interactuar con su máquina.

V

Virtualbox: Es un programa informático que permite alojar diversos ambientes virtuales de sistemas operativos en un solo computador.

W

Windows: Es un sistema operativo de carácter privativo y de uso muy extendido en el ámbito cotidiano y empresarial.

Prefacio

La presente obra ha sido elaborada con el objetivo de ser un punto de partida para las personas que quieran incursionar en el mundo de las redes neuronales artificiales (RNA). Así pues, se proporcionan las herramientas conceptuales y matemáticas para que el lector forme sus propias bases sólidas. Además, se incluyen códigos de fácil lectura para que sean replicados y se afiancen los conocimientos adquiridos.

El desarrollo de todos los capítulos recoge las experiencias de los autores, los cuales han participado en diferentes proyectos y avances referentes a la inteligencia artificial (IA) en los últimos años. De este modo, y considerando que en muchas ocasiones no se puede hacer uso o no se cuenta con librerías o frameworks para el desarrollo y la ejecución de algunas ideas o proyectos, se ha hecho necesario elaborar funciones o librerías propias con el fin de contribuir a la comunidad científica y académica. Por esta razón se proporciona la librería RNAlib, la cual es fácil de leer y contiene las funciones necesarias para que el lector pueda desarrollar herramientas propias

y complementarlas con las que pueda trabajar de forma autónoma.

Finalmente, queremos manifestar mucha gratitud a la comunidad del software y hardware libre, y en general a todos los proyectos colaborativos que de forma desinteresada proporcionan excelentes herramientas de desarrollo. Nuestro mayor deseo es que la información aquí consignada sea de utilidad y que inspire al lector para seguir desarrollando e investigando en el campo de las redes neuronales (RN).

Introducción

La IA ha tomado gran relevancia en nuestra sociedad actual, de tal forma que se ha convertido en un tema de conversación habitual en la vida cotidiana. Esto en gran medida se debe a la mejora que ha tenido el hardware en los dispositivos digitales, la disponibilidad de herramientas de código abierto y la difusión por redes sociales. Ahora bien, este proceso de transformación digital genera nuevo conocimiento e introduce nuevos conceptos que también clasifican como IA, tal como el aprendizaje automático *(machine learning)* y el aprendizaje profundo *(deep learning)*. Este último se caracteriza principalmente por el uso de RNA.

Entre las múltiples herramientas para desarrollar RNA que existen, se ha observado que tradicionalmente se implementa el Neural Network Toolbox (Demuth y Beale, 2004) integrado a MATLAB/Simulink, que pertenece a MathWorks. Sin embargo, en los últimos años, librerías, utilidades o paquetes creados para usar con Python como TensorFlow (Abadi *et al.*, 2016), PyTorch (Team PyTorch, 2020), scikit-learn (Pedregosa *et al.*, 2011), entre otras iniciativas como Keras (Chollet,

2015) y paquetes enfocados a las operaciones matemáticas como NumPy (Oliphant, 2006), han abarcado casi en su totalidad las diversas ramas de la IA y la inteligencia computacional en general.

La ventaja que tiene disponer de todas estas librerías es que constantemente son depuradas por empresas o comunidades de software y hardware libre. De esta forma, un desarrollador puede tener acceso al código de estos paquetes (como en el caso de los mencionados para usar con Python). No obstante, este ejercicio le puede resultar muy abrumador al lector cuando se trata de hacer un simple cambio debido a la complejidad y robustez de estos paquetes. La inexperiencia al enfrentarse a tal tarea podría terminar en fracaso y con malos resultados en su implementación.

Adicional a lo anterior, si bien dichas herramientas pueden resultar muy útiles al implementar las RNA, se considera que, a su vez, no dejan observar de manera detallada el funcionamiento de las redes. Es así como esta obra busca guiar al lector para que implemente herramientas computacionales que le permitan apropiarse de los conceptos básicos de las RNA y, de igual forma, desarrollar destrezas para programarlas, con el objetivo de que en un futuro pueda proponer sus propios algoritmos. Cabe aclarar, sin embargo, que con esto no se desea reemplazar las herramientas ya establecidas; solo se pretende fomentar la curiosidad y el desarrollo de herramientas propias que le permitan al lector un conocimiento más profundo en el área.

Por supuesto que, así como existen diversas herramientas para el diseño e implementación de RNA, hay un sinnúmero de publicaciones concernientes al tema específico que aquí se desea tratar. Libros como *Neural Network Design* (Hagan, Demuth, Beale y De Jesús, 2014), *Neural Networks: A Comprehensive Foundation* (Haykin, 1999) y *Artificial Intelligence: A Modern Approach* (Russell y Norvig, 2010) pueden proporcionar una excelente fundamentación teórica, práctica y complementaria puesto que fueron precisamente los que sirvieron como referentes en la construcción de esta obra.

Dado lo anterior, los autores de este nuevo trabajo han decidido que finalmente este documento sea puesto al servicio de la comunidad académica, ya que la información consignada tiene una estructura con la que cualquier estudiante o aprendiz se puede familiarizar muy fácilmente. De modo tal que el lector no solo se encontrará con los conceptos más simples de la IA, como lo es el funcionamiento de una neurona no biológica, aún considerada como la fuente principal de inspiración de este nuevo campo de conocimiento, sino también con conceptos más complejos relacionados con los procesos de aprendizaje de una red neuronal. Además, se presentan los métodos de entrenamiento que requieren algunos otros tipos de redes para poder funcionar correctamente en los componentes electrónicos, en el caso de que se desee implementarlos, o en los simuladores computacionales, si el propósito solo es observar su respuesta.

Esa combinación casi instantánea entre el componente teórico y el componente práctico es lo que quizás diferencia la estructura de este ejemplar con la de las demás obras mencionadas con anterioridad. En el primer capítulo, y a partir de esta breve introducción, se puede encontrar una pequeña guía, útil y muy detallada, que pone en contexto y advierte de las herramientas que se utilizarán. En el capítulo dos se presentan la historia de las redes neuronales y otros conceptos propios de la IA. En el tercer capítulo se describe a grandes rasgos el modelo natural de una neurona biológica, y en el capítulo cuatro se exponen los conceptos básicos ampliamente fundamentados y fuertemente acompañados de ejemplos prácticos de las RNA. Finalmente, en el capítulo cinco se abordan los procesos de entrenamiento y los tipos de entrenamientos más usados a lo largo del desarrollo de las RNA, además de la descripción de códigos de ayuda para resolver problemas matemáticos que se suelen presentar en el momento de diseñar las redes.

Es así como a lo largo de esta obra el lector se encontrará poco a poco sumergido en conceptos más complejos que los anteriores, acompañados sutilmente de ejemplos prácticos con los cuales afianzará magistralmente la teoría y nutrirá sus habilidades técnicas. Además, los capítulos cuatro y cinco poseen cada uno un apartado de ejercicios en su parte final, en su mayoría propuestos por los mismos autores con la intención de seguir proporcionando material de trabajo y de desarrollo. De manera similar, en los apéndices de

códigos en Bash Shell y RNAlib se encontrarán herramientas diseñadas específicamente para complementar el estudio de los códigos en C. A continuación, se detallarán los pormenores requeridos para obtener el mayor provecho de este manuscrito.

Convenciones

El lenguaje de programación principal con el cual se desarrollarán los algoritmos de este libro será C, y como compilador se hará uso de gcc (GCC, 2020). Sin embargo, en ocasiones se implementarán el lenguaje Bash Shell (Cooper, 2014) y el paquete computacional Gnuplot (Williams *et al.*, 2019) para graficar. Todo el desarrollo está pensado para trabajar en un sistema operativo basado en GNU/Linux.

Para efectos prácticos, se elaboraron dos archivos, denominados RNAlib.h y RNAlib.c, los cuales cuentan con muchas funciones para el desarrollo de los diferentes ejercicios expresados en esta obra. Estos archivos, que se encuentran descritos en el apéndice B, deben estar en la misma carpeta en donde se realizará la compilación, y para utilizarlos es necesario copiar o enlazar desde otra carpeta. Para hacer el enlace, se deben ejecutar las siguientes instrucciones:

```
ln -s ~/<Path de la carpeta>/
RNAlib.h
ln -s ~/<Path de la carpeta>/
RNAlib.c
```

Una vez presentes estos archivos o el vínculo de ellos, y tan pronto se encuentre desarrollado el código por ejecutar, se debe compilar de la siguiente manera en la terminal:

```
gcc -Wall -o ejecutar code.c
RNAlib.c -lm
```

En la instrucción anterior `gcc` es el compilador; la bandera `-Wall` muestra todos los *warning* (alertas o advertencias) que podrían desencadenar errores; `-lm` llama a las librerías matemáticas y, finalmente, con la bandera `-o` se toman todos los archivos y se traducen a un archivo binario ejecutable. Para obtener mayor información sobre estas banderas y otras que no se incluyen en este libro puede consultar GCC (2020).

Una vez compilado, se debe ejecutar de la siguiente manera:

```
./ejecutar
```

Tanto el archivo binario como el código fuente pueden tener cualquier nombre (respetando las reglas de C y gcc). Sin embargo, si se hace uso del archivo de cabecera RNAlib.h, tanto este como el RNAlib.c deben conservar su nombre.

RNAlib cuenta con un repositorio en GitLab:

```
https://gitlab.com/diegorestepo/
rnalib/-/tree/master/librerias
```

Este repositorio, en el que estarán alojadas las versiones más actuales, es de libre acceso, de manera que todos los que lo deseen pueden aportar a este proyecto. Adicionalmente, se ha preparado una máquina virtual con Xubuntu 14.04 de 32 bits en formato OVA, la cual corre sobre VirtualBox 6.1 o superior (para evitar errores se debe verificar que la herramienta de virtualización del computador anfitrión esté habilitada). Esta máquina virtual está dividida en tres archivos de formato zip denominados:

```
ElCaminoALasRedesNeuronales.zip.001
ElCaminoALasRedesNeuronales.zip.002
ElCaminoALasRedesNeuronales.zip.003
```

Estos archivos pueden ser descargados desde el siguiente enlace:

```
https://universidadmag-my.
sharepoint.com/:f:/g/personal/
diegorestrepoal_unimagdalena_edu_
co/ErmxqVz7PyZKunfhkuoujAUBmXY-
fX22hSKIcQHuYTKYAw?e=gi3unS
```

Una vez descargados, solo es necesario descomprimir el primero para obtener el archivo OVA, que luego se puede importar en el VirtualBox. Es preciso anotar en este caso que, para que esta máquina virtual corra adecuadamente, el computador anfitrión debe poseer más de 2 GB de memoria RAM.

Después de importar la máquina virtual, el lector podrá acceder a ella por medio del usuario `rna` con la contraseña `camino123`. En el escritorio de la máquina virtual encontrará una carpeta llamada `RNA`, y dentro de ella dos subcarpetas: `Ejemplos` y `librerías`, las cuales contienen todo lo necesario para seguir esta obra.

Bash Shell

Bash Shell (Bourne-Again Shell) es un lenguaje de programación interpretado que se convirtió en un estándar para las secuencias de comandos de los sistemas UNIX y derivados o inspirados en este, como GNU/Linux. Principalmente, Bash se emplea para administrar el sistema operativo desde la terminal o un emulador de terminal, y para automatizar tareas y procesos por medio de *scripts*.

En este documento se realizan *scripts* en Bash para automatizar la compilación de algunos códigos en C. Esto implica llamar dependencias o utilidades, modificar parámetros de dichos códigos o automatizar la generación de gráficas. Algunos de los comandos más utilizados en este libro se observan en la tabla 1.

Tabla 1. Comandos de Bash más usados

Comando	Descripción
cat	Concatena, crea y muestra el contenido de los archivos
cd	Cambio de directorio
clear	Limpia la terminal, borra el texto impreso
cp	Copia archivo
chmod	Permite cambiar los atributos de un archivo o directorio
echo	Imprime en la terminal
less	Muestra el contenido de un archivo
ln	Crea un *link* de referencia a un archivo preexistente
ls	Lista el contenido del directorio
mkdir	Crea un directorio
mv	Mover un archivo
pwd	Muestra el directorio actual

Fuente: elaboración propia.

En la tabla 1 solo se presentan los comandos que competen a esta obra; sin embargo, existen muchos más que pueden ser consultados en Cooper (2014). Adicionalmente, Bash cuenta también con estructuras de control como el *if* y estructuras iterativas como el *for*, que se presentan en el transcurso de este documento.

Para realizar un *script* en Bash se debe iniciar con #!/bin/bash, lo cual llama al intérprete (normalmente se utiliza la extensión .sh en este tipo de archivos, pero no es de cumplimiento obligatorio). A continuación, se presenta un ejemplo de un *script* en Bash Shell que lleva por nombre *bienvenida.sh* y es la introducción a los lectores en este lenguaje. En este encontramos la llamada al intérprete, la instrucción para limpiar la terminal, varios mensajes al usuario, la muestra del directorio actual de trabajo, y la creación de directorios y archivos de texto.

```
#!/bin/bash

clear

echo
echo "Bienvenido"
echo

echo
echo "Usted se encuentra trabajando
en el directorio:"
pwd

echo
echo "Creando nuevo directorio"
mkdir nuevo_directorio
```