

DEEP LEARNING

TEORÍA Y APLICACIONES

Jesús Alfonso
López Sotelo



Jesús Alfonso López Sotelo

Ingeniero electricista, magíster en Automática y doctor en Ingeniería.

Tiene más de veinte años de experiencia en docencia y en desarrollo de proyectos relacionados con Inteligencia Artificial. Sus áreas de interés son las redes neuronales artificiales y el aprendizaje profundo (Deep Learning), los sistemas difusos, la computación evolutiva, la enseñanza de la inteligencia artificial y el impacto que esta tecnología pueda tener en nuestra sociedad.

Es investigador senior del Sistema nacional de ciencia tecnología e innovación en Colombia de Colciencias, es miembro profesional del IEEE (The Institute of Electrical and Electronics Engineers, Inc.) y pertenece a la Sociedad de Inteligencia Computacional de dicho instituto. Actualmente está vinculado a la Universidad Autónoma de Occidente en Cali y pertenece al Grupo de Investigación en Energías, GIEN. Ha publicado diversos artículos, capítulos de libro y libros en las temáticas de redes neuronales artificiales y otras técnicas de inteligencia artificial

		D	E	E	P						
<i>T</i>	<i>E</i>	<i>O</i>	<i>R</i>	<i>Í</i>	<i>A</i>					<i>Y</i>	

JESÚS ALFONSO LÓPEZ SOTELO

				L	E	A	R	N	I	N	G
A	P	L	I	C	A	C	I	O	N	E	S

Alpha**Editorial**

Alpha Editorial

Alfaomega Colombiana S.A.

Calle 62 20-46 esquina, Bogotá

Teléfono (57-1) 746 0102

Fax: (57-1) 210 0122

cliente@alfaomegacolombiana.com

www.alpha-editorial.com

Primera edición: Bogotá, enero del 2021

© Alfaomega Colombiana S.A. / Alpha Editorial

© Jesús Alfonso López Sotelo

Derechos reservados. Esta publicación no puede ser reproducida total ni parcialmente. Ni puede ser registrada por un sistema de recuperación de información, en ninguna forma ni por ningún medio, sea mecánico, fotoquímico, electrónico, magnético, electroóptico, fotocopia o cualquier otro, sin el previo permiso escrito de la editorial.

Edición: Sandra Ardila

Portada: Ana Paula Santander

ISBN 978-958-778-686-6

ISBN 978-958-778-687-3 digital

Impreso en Colombia

Printed and made in Colombia

CONTENIDO

Lista de figuras	IX
Lista de tablas	XIV
Lista de variables	XV
Introducción	XIX

1. Conceptos sobre redes neuronales artificiales y aprendizaje profundo

Introducción	1
Objetivos del capítulo	1
Inteligencia artificial, redes neuronales artificiales y Deep Learning	1
Breve reseña histórica	3
La neurona biológica	6
La neurona artificial	9
Procesamiento matemático en la neurona artificial	10
Red neuronal artificial	12
Arquitecturas de redes neuronales artificiales	13
De acuerdo con la cantidad de capas	14
Redes multicapa superficial o de una capa oculta	14
Redes multicapa profundas	15
De acuerdo con el flujo de la información	15
Redes Feedforward	15
Redes recurrentes	16
El aprendizaje en las redes neuronales artificiales	16
Aprendizaje supervisado	18
Aprendizaje no supervisado	19
Aprendizaje por refuerzo	20
Ejemplo de procesamiento de la información en una red neuronal	20
Aplicaciones	22
Visión por computador	22
Procesamiento de voz	25
Reconocimiento de sonido	25
Procesamiento de texto	26
Otras aplicaciones	26

2. Los principios del aprendizaje profundo: redes monocapa

Introducción	27
Red Neuronal Perceptron	28
Arquitectura y funcionamiento	28
Algoritmo de aprendizaje	31

Red Neuronal Adaline (ADaptative LINear Element)	34
Arquitectura	34
Algoritmo de aprendizaje	35
Limitaciones del Perceptron	37
Proyectos de aplicación	39
Solución de la función lógica AND con un Perceptron	39
Reconocimiento de caracteres usando el Perceptron	42
Filtro adaptativo usando una red Adaline	46
Filtrado de señales de voz	49
Filtro adaptativo usando una red Adaline implementada en Arduino	49
<i>Ejercicios propuestos</i>	54

3. Superando las dificultades con la propagación inversa en las redes

Introducción	59
Arquitectura general de un Perceptron multicapa superficial	60
Entrenamiento de un MLP superficial	61
Nomenclatura para las redes superficiales	61
Algoritmo de entrenamiento	62
Gradiente descendente estocástico y gradiente descendente	62
Gradiente descendente estocástico (GDE)	62
Procesamiento de datos hacia adelante “feedforward”	63
Actualización de pesos para la capa de salida	64
Actualización de los bias para la capa de salida	65
Actualización de pesos para la capa oculta	65
Backpropagation	66
Actualización de los bias para la capa oculta	67
Variaciones del gradiente descendente	67
Algoritmo gradiente descendente con alfa variable	69
Algoritmo gradiente descendente con momentum clásico	69
Algoritmos de segundo orden para redes neuronales MLP	70
Método de Newton aplicado a redes neuronales	72
Levenberg Marquardt	74
Gradiente conjugado	74
Consideraciones de diseño	77
Conjuntos de aprendizaje y de validación	77
Dimensión de la red neuronal	78
Funciones de activación	79
Pre y posprocesamiento de datos	79
Regularización	81
Regularización por parada temprana	81
Regularización por limitación de la magnitud de los pesos L2	83
Proyectos de aplicación	84
Solución del problema de la función XOR con Matlab	84
Aprendizaje de una función seno con Matlab	86
Aprendizaje de una superficie (función de dos variables) Silla de Montar con Matlab	86

Identificación de sistemas usando redes neuronales MLP	89
Diseño del experimento y muestreo de datos	91
Modelo a usar y estimación de parámetros (Entrenamiento de la red)	93
Validación del modelo obtenido con la RNA	95
Aplicación a la clasificación de patrones (el problema del IRIS)	95
Reconocimiento de caracteres mano escritos (conjunto de datos MNIST) con una red superficial	97
Generador de ondas senoidales implementado con una red neuronal MLP en Arduino	102
Emulación de un sistema dinámico implementado con una red neuronal MLP en Arduino	104
<i>Ejercicios propuestos</i>	106

4. Aprendizaje profundo (*Deep Learning*) o las redes de muchas capas

Introducción	117
Problemas para entrenar redes neuronales profundas	118
Desvanecimiento del gradiente	118
Se requiere muchos datos	121
Se requiere una alta capacidad de cómputo	122
¿Cómo solucionar el problema del desvanecimiento del gradiente?	123
Cambio en funciones de activación	123
Cambio en la función de pérdida o de costo	124
Reconocimiento de patrones convencional vs Deep Learning	125
Extracción de características automático en una imagen	126
Algoritmos de entrenamiento para redes profundas	126
Algoritmo gradiente descendente con momentum Nesterov	127
Algoritmo gradiente descendente tipo AdaGrad	127
Algoritmo gradiente descendente tipo RMSProp	128
Algoritmo gradiente descendente tipo AdaDelta	129
Algoritmo gradiente descendente tipo Adam	130
Arquitecturas de Deep Learning	130
Autocodificadores apilados	130
Redes neuronales convolucionales (Convolutional Neural Networks, CNN)	132
Redes recurrentes tipo Long Short Term Memory (LSTM)	132
Redes generativas profundas	134
Autocodificadores Variacionales (VAE)	134
Redes Generadoras Adversarias (GAN)	134
Aprendizaje por Refuerzo Profundo (Deep Reinforcement Learning, DRL)	136
Proyectos de aplicación	139
Aplicación a la clasificación del problema del Iris con una red profunda	139
Reconocimiento de caracteres mano escritos (MNIST data set) con una red profunda	140
Disminución de la dimensionalidad con un autocodificador para el data set MNIST	144
Reconocimiento de caracteres mano escritos (MNIST) con un autocodificador disperso	155
<i>Ejercicios propuestos</i>	159

5. Redes convolucionales

Introducción	163
Breve introducción al procesamiento de imágenes	163
¿Cómo ve un computador?	164
Inspiración biológica de una CNN	167
Funcionamiento de una red convolucional	167
Convolución	167
Zero-Padding	169
Convolución con imágenes a color	170
Función de activación	171
Pooling	172
Capa clasificadora	172
Otros procesos comunes en CNN	172
Arquitectura general de una CNN	177
Dimensionamiento de una red neuronal convolucional	177
Algunas arquitecturas de redes convolucionales representativas	179
LetNet-5	179
AlexNet	180
VGG16 net	180
Resnet	180
Inception	182
De Alexnet a Inception	182
Proyectos de aplicación	184
Uso de una red preentrenada en Matlab	184
Entrenamiento de una CNN para reconocimiento de caracteres mano escritos del data set MNIST	184
Entrenamiento de una CNN para reconocimiento de imágenes Data set CIFAR-10	186
Transfer Learning con Matlab	200
Deep Dream	204
<i>Ejercicios propuestos</i>	213
Bibliografía	217

LISTA DE FIGURAS

Figura 1.1	El aprendizaje profundo como un campo de la inteligencia artificial.	3
Figura 1.2	El aprendizaje automático y aprendizaje profundo.	4
Figura 1.3	Línea de tiempo del desarrollo del aprendizaje profundo (1943–1989).	7
Figura 1.4	Línea de tiempo del desarrollo del aprendizaje profundo (1990–Actualidad).	8
Figura 1.5	La neurona biológica.	9
Figura 1.6	Neurotransmisores en una sinapsis.	9
Figura 1.7	Modelo de neurona artificial.	10
Figura 1.8	Funciones de activación.	11
Figura 1.9	Procesamiento función de activación Softmax.	12
Figura 1.10	Estructura de una red multinivel o multicapa con todas las conexiones hacia adelante.	14
Figura 1.11	Red neuronal artificial monocapa.	15
Figura 1.12	Red neuronal artificial multicapa superficial.	16
Figura 1.13	Red neuronal artificial multicapa profunda.	16
Figura 1.14	Red neuronal recurrente.	17
Figura 1.15	Aprendizaje supervisado.	19
Figura 1.16	Aprendizaje no supervisado.	20
Figura 1.17	Aprendizaje por refuerzo.	21
Figura 1.18	Estructura de conexión entre neuronas.	21
Figura 1.19	Tareas en visión por computador realizadas con redes neuronales profundas.	23
Figura 1.20	Ejemplo de descripción de imágenes: un montón de frutas y verduras en un mesa,	23
Figura 1.21	Traducción simultánea usando como entrada el texto capturado en una imagen.	23
Figura 1.22	Ejemplo Deep Dream. A la izquierda tenemos la imagen original y a la derecha la imagen procesada.	24
Figura 1.23	Ejemplo de procesamiento de una imagen con Neural Style.	24
Figura 2.1	Arquitectura de un Perceptron.	28
Figura 2.2	Perceptron para trabajar puntos en dos dimensiones.	28
Figura 2.3	Semiplanos generados por un Perceptron.	29
Figura 2.4	Representación de la separación que hace un Perceptron en 3D.	29
Figura 2.5	Región de separación lineal de un Perceptron en el plano.	31
Figura 2.6	Región de separación lineal de un Perceptron en el plano con un vector de pesos mal orientado, generando una clasificación errada.	31
Figura 2.7	Región de separación lineal de un Perceptron en el plano con un vector de pesos bien orientado, generando una clasificación adecuada.	32
Figura 2.8	Estructura de una red tipo Adaline.	35
Figura 2.9	Estructura de una red tipo Adaline con múltiples salidas.	35
Figura 2.10	Gradiente descendente.	36
Figura 2.11	Estructura de una red neuronal multicapa.	39
Figura 2.12	Representación gráfica del problema de la XOR.	40
Figura 2.13	Solución del problema de la XOR con dos Perceptrones.	40
Figura 2.14	Solución del problema de la XOR con una red multicapa compuesta de tres Perceptrones convencionales.	41
Figura 2.15	Código para entrenar una red tipo Perceptron para que aprenda la función lógica AND.	41
Figura 2.16	Patrones a clasificar.	43
Figura 2.17	Patrones a clasificar y la recta clasificadora inicial.	43
Figura 2.18	Evolución del error durante el entrenamiento de la red.	43
Figura 2.19	Evolución del error durante el entrenamiento de la red.	44
Figura 2.20	Patrones a clasificar y la recta clasificadora final.	44
Figura 2.21	Red neuronal generada en el Simulink.	44

Figura 2.22	Diagrama para verificar el comportamiento de la red neuronal.	46
Figura 2.23	Representación binaria del número dos.	46
Figura 2.24	Código para entrenar una red tipo Perceptron para que aprenda los números de 0 al 9 codificados en una matriz de 5x3.	47
Figura 2.25	Evolución del error de entrenamiento.	48
Figura 2.26	Evolución del error de entrenamiento.	50
Figura 2.27	Reconocedor de caracteres implementado en Simulink®.	50
Figura 2.28	Esquema de filtrado adaptativo con una red Adaline.	50
Figura 2.29	Esquema de un filtro adaptativo.	51
Figura 2.30	Señal contaminada con ruido.	51
Figura 2.31	Señal filtrada y evolución del error.	51
Figura 2.32	Código para filtrar una señal senoidal usando una red Adaline.	52
Figura 2.33	Esquema para filtrar una señal de voz con una red Adaline.	53
Figura 2.34	Código para filtrar una señal de voz usando una red Adaline.	53
Figura 2.35	Señales en el proceso de filtrado de voz.	54
Figura 2.36	Código para filtrar una señal senoidal usando una red Adaline implementada en Arduino.	54
Figura 2.37	Señales en una etapa inicial del proceso de filtrado.	56
Figura 2.38	Señales en una etapa avanzada del proceso de filtrado.	56
Figura 2.39	Señal cuadrada sin contaminación.	58
Figura 2.40	Señal exponencial sin contaminación.	58
Figura 3.1	Arquitectura general de un MLP.	60
Figura 3.2	Diferencia en el comportamiento de GDE y GD.	63
Figura 3.3	Flujo del error (en líneas punteadas) aplicando backpropagation para estimar el error de las neuronas de la capa oculta.	66
Figura 3.4	Actualización de pesos usando el momentum clásico.	71
Figura 3.5	Oscilaciones alrededor del mínimo producidas por el gradiente descendente.	71
Figura 3.6	Superficie de error para una red con dos conexiones sinápticas.	71
Figura 3.7	Oscilaciones alrededor del mínimo producidas por el gradiente descendente.	72
Figura 3.8	Atenuación de las oscilaciones alrededor del mínimo cuando se usa gradiente descendente con momentum.	72
Figura 3.9	Atenuación de las oscilaciones alrededor del mínimo cuando se usa gradiente descendente con momentum.	72
Figura 3.10	Evolución del gradiente.	75
Figura 3.11	Curvas de nivel de la función de error.	76
Figura 3.12	Conjunto de datos de entrenamiento y validación.	78
Figura 3.13	Normalización de datos para ser usados por una RNA.	80
Figura 3.14	Sobreentrenamiento en una red neuronal artificial.	82
Figura 3.15	Subentrenamiento en una red neuronal artificial.	82
Figura 3.16	Regularización por parada temprana.	83
Figura 3.17	Código para entrenar una red tipo MLP para que aprenda la función lógica XOR.	85
Figura 3.18	Separación lograda por una red MLP.	87
Figura 3.19	Visualización 3D de la separación lograda por una red MLP para el problema de la XOR.	87
Figura 3.20	Estructura de la red a entrenar.	87
Figura 3.21	Código para entrenar una red tipo MLP para el aprendizaje de una función senoidal.	88
Figura 3.22	Situación del aprendizaje de la red.	89
Figura 3.23	Superficie original.	89
Figura 3.24	Estructura de la red a entrenar.	90
Figura 3.25	Superficie aprendida por la red al inicio del aprendizaje (iteraciones = 1).	90
Figura 3.26	Superficie aprendida por la red en una etapa intermedia del aprendizaje (iteraciones = 5).	90
Figura 3.27	Superficie aprendida por la red al fin del aprendizaje.	91
Figura 3.28	Relación entre los patrones de aprendizaje	91

Figura 3.29	Código para entrenar una red tipo MLP para el aprendizaje de una función de dos variable (silla de montar).	92
Figura 3.30	Red neuronal usada para identificar un sistema dinámico o planta.	93
Figura 3.31	Esquema en Simulink del experimento.	93
Figura 3.32	Una posible entrada y su salida obtenida en el experimento.	94
Figura 3.33	Arquitectura de la RNA utilizada para la identificación.	94
Figura 3.34	Código para entrenar una red tipo MLP para el aprendizaje de un sistema dinámico.	95
Figura 3.35	Esquema en Simulink para validar el modelo neuronal.	96
Figura 3.36	Validación del modelo neuronal obtenido.	96
Figura 3.37	Arquitectura de la RNA utilizada para solucionar el problema del iris.	97
Figura 3.38	Código para entrenar una red tipo MLP para resolver el problema	98
Figura 3.39	Visualización de los datos de entrenamiento.	100
Figura 3.40	Visualización de los datos de validación.	100
Figura 3.41	Visualización de la clasificación realizada por la red neuronal.	100
Figura 3.42	Matriz de confusión.	101
Figura 3.43	Ejemplo de algunas imágenes de la base de datos MNIST (Generar imágenes).	102
Figura 3.44	Arquitectura de la RNA MLP superficial utilizada para trabajar con la base de datos MNIST.	102
Figura 3.45	Información de la estructura de datos que cargan en Matlab para trabajar con la base de datos MNIST.	103
Figura 3.46	Código para cargar y organizar la información del conjunto de datos MNIST.	103
Figura 3.47	Visualización de las primeras 36 imágenes de la base de datos MNIST.	104
Figura 3.48	Código para entrenar la red neuronal superficial con el conjunto de datos MNIST.	105
Figura 3.49	Código para validar la red neuronal superficial entrenada con el conjunto de datos MNIST.	106
Figura 3.50	Matriz de confusión.	107
Figura 3.51	Código para extraer los pesos de una red MLP ya entrenada.	108
Figura 3.52	Comandos para obtener los rangos de normalización de las entradas y salidas de una red MLP.	108
Figura 3.53	Código para implementar un generador de ondas senoidales en Arduino usando los pesos de una red MLP ya entrenada.	109
Figura 3.54	Señal de senoidal producida por la red neuronal.	111
Figura 3.55	Código para implementar la emulación de un sistema dinámico en Arduino usando los pesos de una red MLP ya entrenada.	111
Figura 3.56	Señal de salida producida por la red neuronal emulando el sistema dinámico de primer orden.	113
Figura 3.57	Función de dos variables a identificar.	114
Figura 3.58	Código para generar la función peaks en Matlab.	115
Figura 3.59	Representación binaria de un carácter.	116
Figura 4.1	Visualización del desvanecimiento del gradiente en una red MLP.	119
Figura 4.2	Visualización del desvanecimiento del gradiente en una red profunda.	119
Figura 4.3	Visualización del desvanecimiento del gradiente entre dos capas sucesivas en una red MLP.	119
Figura 4.4	Red sencilla para revisar el problema del desvanecimiento del gradiente.	120
Figura 4.5	Gráfica de la derivada de la función sigmoideal.	121
Figura 4.6	Gráfica de la función de activación ReLU.	123
Figura 4.7	Variaciones propuestas para la función de activación tipo ReLU.	124
Figura 4.8	Gráfica de la función de activación ELU.	125
Figura 4.9	Proceso de reconocimiento de patrones, en la parte superior de manera clásica y en la parte inferior con Deep Learning.	126
Figura 4.10	Extracción de características automático.	127
Figura 4.11	Actualización de pesos usando el momentum con variación de Nesterov.	128
Figura 4.12	Autocodificador.	131
Figura 4.13	Red profunda resultante al apilar los diferentes autocodificadores.	131

Figura 4.14	Red convolucional (CNN).	133
Figura 4.15	Unidad LSTM.	133
Figura 4.16	Predicción del siguiente valor de la secuencia.	133
Figura 4.17	Clasificación de la secuencia de entrada.	133
Figura 4.18	Generación de unas secuencias en función de la entrada.	133
Figura 4.19	Predicción de una nueva secuencia dependiendo de la secuencia de entrada.	135
Figura 4.20	Codificación de un rostro realizada por un autocodificador.	135
Figura 4.21	Codificación de un rostro realizada por un autocodificador variacional.	135
Figura 4.22	Arquitectura básica de un VAE.	135
Figura 4.23	Arquitectura GAN.	137
Figura 4.24	Mejora de la resolución de una imagen con una GAN.	137
Figura 4.25	Generación automática de imagen con una GAN.	137
Figura 4.26	Ejemplo de traducción de Imagen a Imagen con una GAN.	138
Figura 4.27	Ejemplo de traducción de Imagen a Imagen con una GAN.	138
Figura 4.28	Ejemplo de una DRL usada para aprender un juego.	139
Figura 4.29	Arquitectura de la red profunda a utilizar para resolver el problema del Iris.	139
Figura 4.30	Código para entrenar una red tipo MLP para resolver el problema de clasificación del Iris con una red profunda.	141
Figura 4.31	Visualización de los datos de entrenamiento usando las tres primeras características.	143
Figura 4.33	Visualización de la clasificación realizada por la red neuronal.	143
Figura 4.32	Visualización de los datos de validación usando las tres primeras características.	143
Figura 4.34	Matriz de confusión.	144
Figura 4.35	Arquitectura de la red profunda a usar.	144
Figura 4.36	Código para cargar y organizar la información del conjunto de datos MNIST.	145
Figura 4.37	Visualización de las primeras 36 imágenes de la base de datos MNIST.	145
Figura 4.38	Código para entrenar la red neuronal profunda con el conjunto de datos MNIST.	146
Figura 4.39	Código para validar la red neuronal profunda entrenada con el conjunto de datos MNIST.	148
Figura 4.40	Matriz de confusión.	149
Figura 4.41	Arquitectura del autocodificador a usar donde se muestra la componente codificadora y decodificadora.	149
Figura 4.42	Código para cargar y organizar la información del conjunto de datos MNIST.	150
Figura 4.43	Código para entrenar un autocodificador con el conjunto de datos MNIST.	151
Figura 4.44	Código para verificar la capacidad de reconstrucción de imágenes de la red entrenada.	152
Figura 4.45	Ejemplos de números reconstruidos por el autoencodificador.	152
Figura 4.46	Estructura de la componente codificadora del autocodificador.	153
Figura 4.47	Código que permite visualizar la información generada en la capa codificadora.	153
Figura 4.48	Representación dígitos aprendidos por el autocodificador en la última capa del codificador (capa latente).	154
Figura 4.49	Estructura de la componente decodificadora del autocodificador.	155
Figura 4.50	Código que permite generar imágenes de caracteres mano escritos con la capa decodificadora.	156
Figura 4.51	Ejemplos de números generados por el decodificador.	157
Figura 4.52	Código para cargar y organizar la información del conjunto de datos MNIST.	157
Figura 4.53	Muestra de las imágenes de entrenamiento.	157
Figura 4.54	Código para entrenar el primer autocodificador.	158
Figura 4.55	Arquitectura del primer autocodificador.	158
Figura 4.56	Código para visualizar los pesos de la capa oculta del primer autocodificador.	158
Figura 4.57	Visualización de los pesos de las primeras 25 neuronas de la capa oculta del primer autocodificador.	159
Figura 4.58	Código para entrenar el segundo autocodificador.	160
Figura 4.59	Arquitectura del segundo autocodificador.	160

Figura 4.60	Código para apilar los autocodificadores ya entrenados y entrenar una capa clasificadora tipo softmax.	160
Figura 4.61	Arquitectura de la red profunda generada al apilar los dos autocodificadores y al adicionarle una capa de clasificación tipo softmax.	160
Figura 4.62	Código para validar la red construida con los autocodificadores apilados.	161
Figura 4.63	Matriz de confusión obtenida con los datos de validación.	161
Figura 4.64	Código para realizar un ajuste fino de la red generada con los autocodificadores apilados y la capa clasificadora.	162
Figura 4.65	Matriz de confusión obtenida con los datos de validación.	162
Figura 5.1	Experimento de Hubel y Wiesel.	164
Figura 5.2	Detección de rostro en una imagen.	165
Figura 5.3	Representación matricial de una imagen a color a RGB.	165
Figura 5.4	Procesamiento de un filtro en una imagen.	166
Figura 5.5	Detección de líneas verticales y horizontales.	166
Figura 5.6	Filtros usados por Viola y Jones para detectar un rostro.	167
Figura 5.7	Representación gráfica de la convolución.	168
Figura 5.8	Cálculo del tamaño de imagen de salida. Win=5, F=3 y S=1.	169
Figura 5.9	Procesamiento de la convolución con una imagen RGB.	170
Figura 5.10	Gráfica de la función de activación ReLU.	171
Figura 5.11	Efecto de aplicar la función de activación ReLU a=Entrada b=Salida.	173
Figura 5.12	Ejemplo del pooling.	173
Figura 5.13	Visualización del efecto pooling sobre la extracción de características.	173
Figura 5.14	Procesamiento de la capa clasificador tipo softmax.	174
Figura 5.15	Visualización del dropout	174
Figura 5.16	Red convolucional (CNN).	175
Figura 5.17	Red neuronal convolucional a dimensionar.	178
Figura 5.18	LetNet-5.	178
Figura 5.19	AlexNet.	181
Figura 5.20	Estructura de la red VGG net.	181
Figura 5.21	Izquierda bloque residual genérico. Derecha bloque residual en una red convolucional	181
Figura 5.22	Bloque de procesamiento Inception.	183
Figura 5.23	Evolución de la precisión (Top-1 %) en Imagenet usando redes convolucionales y sus variaciones.	183
Figura 5.24	Evolución del error (top-5%) de Imagenet. Desde el 2012 las propuestas ganadoras se han basado en CNN.	183
Figura 5.25	Diferentes arquitecturas de CNN considerando su carga computacional y su precisión	185
Figura 5.26	Código para cargar una CNN preentrenada en Matlab y usarla para clasificar una imagen.	185
Figura 5.27	Visualización de la estructura de la CNN (VGG-16).	187
Figura 5.28	Salida de la red con cuatro diferentes imágenes de entrada.	188
Figura 5.29	Estructura de carpetas para el data set MNIST compatible con el objeto ImageDataStore de Matlab.	188
Figura 5.30	Código para crear el data set de MNIST compatible con un ImageDataStore.	188
Figura 5.31	Arquitectura de la CNN usada para resolver el problema del data set MNIST.	191
Figura 5.32	Código para crear una CNN y su entrenamiento con el data set MNIST.	192
Figura 5.33	Ejemplos de algunas de las imágenes usadas para entrenar la red.	195
Figura 5.34	Visualización de la cantidad de patrones por cada una de las categorías para el data set MNIST.	195
Figura 5.35	Visualización de la estructura de la CNN creada.	195
Figura 5.36	Evolución del proceso de entrenamiento de la red convolucional utilizada.	196
Figura 5.37	Matriz de confusión.	196
Figura 5.38	Algunas imágenes del data set CIFAR10.	200
Figura 5.39	Arquitectura de la CNN usada para resolver el problema del data set CIFAR-10.	200
Figura 5.40	Código para crear una CNN y su entrenamiento.	197

Figura 5.41	Ejemplos de algunas de las imágenes usadas para entrenar la red del data set CIFAR-10.	201
Figura 5.42	Visualización de la cantidad de patrones por clase.	201
Figura 5.43	Visualización de la estructura de la CNN creada.	202
Figura 5.44	Evolución del proceso de entrenamiento de la red convolucional utilizada.	202
Figura 5.45	Matriz de confusión.	203
Figura 5.46	Transfer learning donde se reemplaza solo la capa final o clasificadora.	203
Figura 5.47	Código para redimensionar las imágenes del data set original a un tamaño compatible con AlexNet.	205
Figura 5.48	Código para realizar transfer learning con la CNN AlexNet.	206
Figura 5.49	Visualización de la cantidad de patrones por clase.	207
Figura 5.50	Visualización de la estructura de la red AlexNet.	208
Figura 5.51	Salida de la red con algunas de las imágenes de entrada.	209
Figura 5.52	Salida de la red con nueve diferentes imágenes de entrada con la etiqueta generada.	209
Figura 5.53	Visualización del proceso de entrenamiento.	209
Figura 5.54	Matriz de confusión obtenida.	210
Figura 5.55	Diagrama de flujo DeepDream.	211
Figura 5.56	Código para realizar deepdream con la CNN AlexNet.	211
Figura 5.57	Imagen original a la que se le aplicará el Deep Dream.	214
Figura 5.58	Resultado de aplicar Deep Dream en la primera capa convolucional.	214
Figura 5.59	Resultado de aplicar Deep Dream en la cuarta capa convolucional.	214
Figura 5.60	Resultado de aplicar Deep Dream en la capa clasificadora.	215
Figura 5.61	Resultado de aplicar Deep Dream en la capa clasificadora.	215
Figura 5.62	Resultado de aplicar Deep Dream en la capa clasificadora con los canales 'Yorkshire terrier' y 'gold finch' (jilguero).	215

LISTA DE TABLAS

Tabla 2.1	Pasos para el entrenamiento de una red Perceptron.	33
Tabla 2.2	Patrones de entrenamiento.	33
Tabla 2.3	Pasos para el entrenamiento de una red ADALINE.	38
Tabla 2.4	Función Lógica XOR.	39
Tabla 2.5	Función Lógica AND.	40
Tabla 2.6	Patrones correspondientes a los caracteres decimales.	45
Tabla 2.7	Codificación de los patrones de entrenamiento para el reconocimiento de caracteres decimales en MATLAB.	48
Tabla 2.8	Validación de la red usando como entrada el número siete.	48
Tabla 2.9	Validación de la red usando como entrada el número nueve.	48
Tabla 2.10	Patrones a clasificar.	57
Tabla 2.11	Patrones a clasificar.	57
Tabla 3.1	Derivadas de las funciones de activación más representativas.	68
Tabla 3.2	Pasos del algoritmo GDE para entrenar una red MLP superficial.	68
Tabla 3.3	Organización de los patrones de entrenamiento.	96
Tabla 5.1	Comparación arquitecturas de redes convolucionales.	184

LISTA DE VARIABLES

f_i	Activación del filtro convolucional ante <i>i-ésimo</i> patrón del minilote
f_{iNor}	Activación del filtro convolucional ante <i>i-ésimo</i> patrón del minilote normalizado
f_{iBN}	Activación del filtro convolucional ante <i>i-ésimo</i> patrón del minilote luego de aplicar <i>batch normalization</i>
$A\Delta\omega(t)$	Acumulación de las variaciones de peso
$AG(t)$	Acumulación del gradiente
b_j^o	Bias de la capa de oculta
b_k^s	Bias de la capa de salida
-	Bias o umbral de la <i>i-ésima</i> neurona
K	Cantidad de filtros en una capa convolucional
n	Cantidad total de pesos en una red neuronal
x_1, x_2, \dots, x_n	Componentes del vector de entrada
d_1, d_2, \dots, d_M	Componentes del vector de salidas deseadas
y_1, y_2, \dots, y_M	Componentes del vector de salidas producidas por la red
X^P	Conjunto de vectores de entrada. Se representa como una matriz
D^P	Conjunto de vectores de salida. Se representa como una matriz
ϵ	Constante usada para estabilidad numérica (evitar división por cero)
$f'(x)$	Derivada de la función a la que se desea calcular la raíz
f_j^o	Derivada de la función de activación de la <i>j-ésima</i> neurona de la capa oculta
f_k^s	Derivada de la función de activación de la <i>k-ésima</i> neurona de la capa de salida
L'	Derivada de la función de pérdida
σ	Desviación estándar de los datos a normalizar
σ_B	Desviación estándar del minilote
$Neta_j$	Entrada neta de <i>j-ésima</i> neurona
$Neta_{pj}^o$	Entrada neta de la <i>j-ésima</i> neurona de la capa oculta
$Neta_{pk}^s$	Entrada neta de la <i>k-ésima</i> neurona de la capa salida
e_j^o	Error capa oculta
e	Error producido a la salida de una neurona
α	Factor de aprendizaje
ν	Factor de decremento de la razón de aprendizaje
τ	Factor de incremento de la razón de aprendizaje
λ	Factor para ponderar la sumatoria de los pesos al cuadrado
β	Factor que define la influencia del <i>momentum</i>
$f(x)$	Función a la que se desea calcular la raíz
f_j^o	Función de activación de la <i>j-ésima</i> unidad oculta
f_k^s	Función de activación de la <i>j-ésima</i> unidad oculta

LISTA DE VARIABLES

L	Función de pérdida
L_R	Función de pérdida regularizada
L_D	Función de pérdida sin regularizar, solo tiene en cuenta el aprendizaje de la salida deseada
p_i	i -ésimo pixel
H	Matriz Hessiana
W	Matriz o vector de pesos sinápticos
O	Número de neuronas de en la capa oculta
N	Número de neuronas de entrada
M	Número de neuronas en la capa de salida
P	Números de patrones de entrenamiento
p	p -ésimo patrón del conjunto de entrenamiento
φ	Parámetro ajustable del <i>batch normalization</i>
n	Parámetro ajustable del <i>batch normalization</i>
ζ	Parámetro de control en el algoritmo Levenberg-Marquard
κ	Parámetro de la parte exponencial de la función de activación ELU
Pa	Parámetro para definir el <i>zero-padding</i>
w_{ji}^o	Peso de la capa de oculta
w_{kj}^s	Peso de la capa de salida
w_{ji}	Peso sináptico entre la neurona j y la neurona i
μ_B	Promedio del minilote
ρ	Razón de decaimiento
y_j	Salida de j -ésima neurona
Con_Out	Salida de la convolución
y_{pj}^o	Salida de la j -ésima neurona de la capa oculta
y_{pk}	Salida de la k -ésima neurona de la capa de salida
y	Salida de la red neuronal artificial
d	Salida deseada
L''	Segunda derivada de la función de pérdida
S	<i>Stride</i> o corrimiento
L_W	Sumatoria de los pesos al cuadrado
W_i^{nlm}	Tamaño de la imagen de entrada
W_{out}^{lm}	Tamaño de la imagen de salida
W_{pol}	Tamaño de la ventana del <i>polling</i>
F	Tamaño del filtro en las redes convolucionales
m	Tamaño del minilote
δ_{pj}^h	Término de error para la j -ésima neurona de la capa oculta h
δ_{pk}^o	Término de error para la k -ésima neurona de la capa de salida
γ	Valor a determinar en el algoritmo de gradiente conjugado
x_{sn}	Valor a sin normalizar
$x_{raiz}(t)$	Valor actual de la raíz

$w(t)$	Valor actual del peso sináptico
$W(t)$	Valor actual del vector o matriz de pesos
$x_{raiz}(t+1)$	Valor actualizado de la raíz
$w(t+1)$	Valor actualizado del peso sináptico
$W(t+1)$	Valor actualizado del vector o matriz de pesos
$x_{m\acute{a}x}$	Valor máximo de los valores a normalizar
$y_{m\acute{a}x}$	Valor máximo de los valores normalizados = +1
$x_{m\acute{i}n}$	Valor mínimo de los valores a normalizar
$y_{m\acute{i}n}$	Valor mínimo de los valores normalizados = -1
y_{nor}	Valor normalizado
μ	Valor promedio de los datos a normalizar
$\Delta w(t)$	Variación del peso sináptico
$\Delta W(t)$	Variación del vector o matriz de pesos
s	Vector conjugado a r
r	Vector conjugado a s
X	Vector de entrada
$Neta$	Vector de entrada neta a una capa de neuronas artificiales
D	Vector de salida deseada
Y	Vector de salida producida por la red
$h(t)$	Vector dirección de actualización en el gradiente conjugado
G	Vector gradiente

A Andrea, Valerie y Emmanuel.

Mi familia, mis "neuronas",
mi "capa", las "conexiones"
que activan mi vida

PRESENTACIÓN

El aprendizaje profundo o *Deep Learning* es una evolución de las redes neuronales artificiales (RNA). Las RNA constituyen una de las técnicas más relevantes de la inteligencia artificial que trata de emular la manera como trabajan las neuronas del cerebro. Este enfoque se encuentra dentro de la vertiente denominada conexionista, pues se basa en imitar el funcionamiento cerebral por medio de redes formadas por unidades sencillas (neuronas artificiales) interconectadas entre sí. Además, el conocimiento se modifica cambiando la fuerza o el valor de las conexiones existentes entre las diferentes neuronas de la red.

Históricamente, las RNA han tenido varios momentos donde, luego de ser muy utilizadas, pasaron a ser casi que abandonadas. En particular, se han presentado dos periodos de este estilo. El primero comenzó al finalizar la década de los años 60, cuando se hicieron notorias las limitaciones de las primeras RNA como el Perceptron Monocapa. Dichas limitaciones fueron extendidas por algunos investigadores a las redes más complejas de varias capas. De este periodo, las RNA salieron a mediados de los 80, luego de encontrar una solución para entrenar redes multicapa, denominada propagación inversa o *backpropagation*. Infortunadamente, el *backpropagation* funcionaba bien con redes superficiales o de una capa oculta, tratar de entrenar redes neuronales con varias capas ocultas no era viable. Lo anterior produjo el segundo de estos periodos de estancamiento a mediados de los 90. En 2006, se comenzaron a generar resultados que solventarían las dificultades para entrenar redes neuronales de varias capas ocultas, dando origen a lo que se denomina aprendizaje profundo (*Deep Learning*, DL). Actualmente, con el DL se obtienen desempeños comparables a los alcanzados por humanos en tareas de reconocimiento de imágenes y de voz, y superando las otras técnicas de aprendizaje automático en diversas aplicaciones. El DL es usado por los gigantes tecnológicos como Facebook, Google y Microsoft para sus desarrollos, como, por ejemplo, el reconocimiento automático de objetos en una imagen, el etiquetado de imágenes y la traducción simultánea.

Este libro se ha estructurado de tal manera que se realiza un seguimiento por el desarrollo histórico de las RNA, presentando lo principales conceptos asociados a ellas, partiendo desde los modelos más simples o monocapa, hasta los modelos de muchas capas o basados en *Deep Learning*.

Inicialmente (Capítulo 1) se mostrará una breve historia de las redes neuronales artificiales desde sus primeros modelos, hasta el surgimiento del aprendizaje profundo. También se trabajarán los conceptos básicos de las redes neuronales. Estos conceptos serán fundamentales para lo que se presentará en el resto del libro.

Históricamente hablando, las primeras RNA datan de finales de los años 50 y comienzos de los 60, época en que se propusieron las redes monocapa como el Perceptron y el Adaline; estos modelos se trabajarán en el Capítulo 2. El entendimiento de

cómo se entrenan este tipo de RNA es clave para comprender redes neuronales superficiales y profundas, como las que se presentarán en los siguientes capítulos de este libro.

Posteriormente, se trabajarán las redes neuronales multicapa de una capa oculta (Capítulo 3). Estas redes se denominan redes superficiales. Se revisarán sus algoritmos de entrenamiento y un concepto clave en el desarrollo de las redes neuronales artificiales, como es el de la propagación inversa o *backpropagation*. El desarrollo de este concepto permitió que la RNA salieran de su primer periodo de inactividad y produjera a su vez uno de mucho desarrollo.

En el Capítulo 4 se revisarán los conceptos fundamentales de *Deep Learning* y una de sus arquitecturas más representativas, como son los autocodificadores (*autoencoders*). El *Deep Learning* ha permitido que la RNA se convierta hoy en día en la técnica de inteligencia artificial con más aplicación y proyección a mediano plazo. De hecho, la mayoría de aplicaciones que hoy se publicitan como grandes logros en el desarrollo de la inteligencia artificial se basan en esta técnica.

Si el *Deep Learning* fuera una película, las redes convolucionales (*Convolutional Neural Networks*, CNN) serían los protagonistas. Estas redes se verán en el Capítulo 5. Se presentarán las operaciones que realiza este tipo de redes neuronales, su arquitectura y su entrenamiento. Una de las aplicaciones relevantes de las redes convolucionales es la clasificación de imágenes, por lo que se revisará cómo se puede utilizar este tipo de red para este fin.

Este libro, además de presentar los principales conceptos teóricos de las redes neuronales artificiales y del aprendizaje profundo, se centra en mostrar cómo se puede aplicar esta tecnología para solucionar diversos problemas. Para este propósito, al final de cada capítulo se ha preparado una sección especial. En dicha sección se expondrá cómo implementar redes neuronales artificiales en diversas plataformas, como Matlab y Arduino. Lo anterior con el fin de resolver problemáticas tan diversas como el aprendizaje de funciones de una o más variable, el modelado de sistemas dinámicos, el filtrado adaptativo de señales, el reconocimiento de caracteres alfabéticos y numéricos, clasificación de tipos de flores, el reconocimiento de patrones en imágenes, entre otras.

CONCEPTOS SOBRE REDES NEURONALES Y ARTIFICIALES Y DEEP LEARNING

INTRODUCCIÓN

Las redes neuronales artificiales (RNA) son una técnica de inteligencia artificial clasificada dentro del aprendizaje automático o *Machine Learning*. Las RNA siguen una tendencia diferente a los enfoques clásicos de la inteligencia artificial que tratan de modelar la inteligencia humana buscando imitar los procesos de razonamiento que ocurren en nuestro cerebro. El enfoque en que se hallan las RNA se encuentra dentro de la vertiente denominada conexionista, que imita el funcionamiento cerebral por medio de redes formadas por unidades sencillas (neuronas artificiales) interconectadas entre sí. Además, el conocimiento se modifica cambiando la fuerza o el valor de las conexiones existentes entre las diferentes neuronas de la red. El enfoque de las RNA no es reciente, pero en los últimos años ha resurgido el interés en ellas debido a una técnica que permite entrenar redes de muchas capas ocultas denominada aprendizaje profundo o *Deep Learning*. El aprendizaje profundo es la base en la cual se apoya la mayoría de los últimos avances actuales en inteligencia artificial. Para entender el aprendizaje profundo, se hace necesario comenzar con los conceptos iniciales y básicos asociados a las RNA, los cuales serán los temas principales de este capítulo.

Objetivos del capítulo

- Comprender la relación entre inteligencia artificial, aprendizaje automático y aprendizaje profundo.
- Estudiar el desarrollo histórico que dio origen al aprendizaje profundo
- Comprender el modelo de neurona artificial y su relación con el funcionamiento de las neuronas biológicas.
- Conocer las diferentes arquitecturas de redes neuronales y los tipos de aprendizaje que se usan en las mismas.
- Presentar algunas aplicaciones del aprendizaje profundo.

Inteligencia artificial, redes neuronales artificiales y Deep Learning

El término de inteligencia artificial se le atribuye a John McCarthy, quien, en 1956, la definió como “la ciencia e ingenio de hacer máquinas inteligentes, especialmente programas de cómputo inteligentes”.

De una manera coloquial, la podemos definir como el conjunto de técnicas creadas por el ser humano, que tratan de emular la inteligencia biológica; por razones obvias, el santo grial sería generar un algoritmo que emule la inteligencia humana considerada la más compleja entre la de todos los seres vivos.

Dentro de la inteligencia artificial hay diferentes vertientes, entre ellas una de las más usadas es la del aprendizaje automático (*Machine Learning*). El aprendizaje automático son una serie de técnicas que son capaces de “aprender” las relaciones intrínsecas que hay en un conjunto de datos. Lo interesante es que no se les define a priori lo que debe aprender, lo aprende de manera automática a partir de la información que se le suministra. Si tengo una imagen y deseo reconocer si hay un gato o un perro, usando un algoritmo convencional, habría que especificar cómo identificar a cada animal, por ejemplo, un gato tiene las orejas puntiagudas, bigotes, etc. Por otro lado, un algoritmo de aprendizaje automático se le da una serie de imágenes de gatos y perros, y de manera automática se espera que aprenda a reconocer estos animales.

Como un subconjunto de las técnicas de aprendizaje automático están las redes neuronales artificiales que, en esencia, son modelos computacionales que tratan de emular el comportamiento de las neuronas del cerebro por medio de la conexión de neuronas artificiales generalmente organizadas en forma de capas. Existen diferentes tipos de redes neuronales como los mapas autoorganizados, el Perceptron, el Adaline, entre otras.

Dentro las redes neuronales artificiales tenemos las redes que se caracterizan por tener muchas capas o profundas. Este conjunto de redes neuronales artificiales y los mecanismos que hacen posible entrenarlas constituyen el denominado aprendizaje profundo o *Deep Learning*.

En la Figura 1.1 vemos cómo se relacionan los campos de la inteligencia artificial, el aprendizaje automático, las redes neuronales artificiales y el aprendizaje profundo.

Uno de los avances que ha permitido el *Deep Learning* respecto a la técnica de aprendizaje automático tradicional es que, debido a su múltiples capas, una red neuronal profunda permite realizar de manera automática una extracción de características de la información que se le suministra como entrada; esta extracción la hace de manera jerárquica, pasando primero por características de bajo nivel, luego encontrando las de nivel intermedio hasta llegar a las de alto nivel. Tomando como ejemplo una imagen, las características de bajo nivel serían bordes y líneas, las de nivel intermedio serían contornos y las de alto nivel podrían ser formas y objetos.

Lo anterior es contrario a lo que las técnicas de aprendizaje automático tradicional realizan, donde primero se realiza una extracción de características generalmente guiada por un usuario. Con las características ya extraídas se procede a realizar la clasificación. En *Deep Learning*, la red profunda, aparte de realizar la extracción de características, puede incluir una capa clasificadora que nos permitirá generar la salida del proceso realizado por la red neuronal.

En la Figura 1.2 mostramos cómo sería el proceso de clasificar una imagen usando aprendizaje automático y *Deep Learning*. Por ejemplo, se desea saber si en la imagen de entrada hay un perro; en el enfoque tradicional de aprendizaje automático

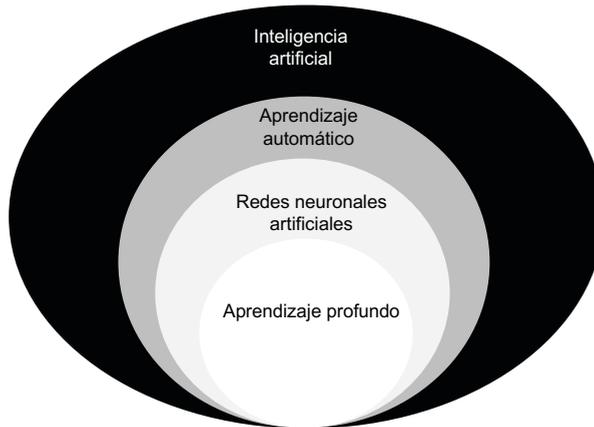


FIGURA 1.1 El aprendizaje profundo como un campo de la inteligencia artificial.

primero se extraen las características de manera manual (para esto hay que resolver la pregunta ¿cómo podemos identificar un perro? Por la forma, por las orejas, etc.) y, con las mismas, se hace una clasificación para determinar si el objeto que hay en la imagen de entrada es un perro o no. Usando *Deep Learning*, a la red profunda se le suministra la imagen de entrada y ella realiza tanto la extracción de características como la clasificación de manera automática, generando como salida nuevamente si hay o no un perro en la imagen de entrada.

Breve reseña histórica

Emular redes neuronales de manera artificial no es un desarrollo reciente. Se hicieron algunos intentos antes del advenimiento de los computadores, pero su verdadero desarrollo tuvo lugar cuando las simulaciones por computador fueron factibles por su capacidad de procesamiento y bajo costo. Luego de un periodo inicial de entusiasmo, las redes neuronales cayeron en una etapa de frustración y desprestigio, ya que el soporte económico era muy limitado, y solo unos pocos investigadores consiguieron logros de algún nivel de importancia. Estos pioneros fueron capaces de desarrollar una tecnología que sobrepasara las limitaciones identificadas en algunas publicaciones de Minsky y Papert en 1969, que sembraron un desencanto y frustración general en la comunidad científica. A este periodo se le puede denominar el primer invierno de las RNA. Luego de superar los inconvenientes para entrenar redes con capas ocultas, llegó un periodo donde se fortaleció el trabajo y la investigación con RNA. Este periodo vino acompañado de un nuevo invierno producido por la incapacidad de solventar los problemas de entrenar RNA con muchas capas ocultas o profundas. Afortunadamente, dichos inconvenientes se solventaron dando origen al denominado aprendizaje profundo, concepto que ha revitalizado el campo de la RNA y han hecho que las mismas sean una de las tecnologías con mejores resultados en aplicaciones de inteligencia artificial.

Acerquémonos a algunas etapas en las que puede resumirse la historia de las redes neuronales artificiales: