Tzolkin Garduño Alvarado, Feliú Sagols Troncoso y Gunnar Wolf

Las neuronas artificiales y su papel central en la inteligencia artificial

Una de las grandes tendencias de los dos últimos años, con merecida razón, ha sido la inteligencia artificial. Dos de los conceptos fundamentales sobre los cuales están creados casi todos los sistemas a los que hoy damos este nombre son las neuronas artificiales (más específicamente, los denominados *perceptrón*) y las redes neuronales (los *perceptrones multicapa*). Nuestra motivación al escribir este artículo es revisar estos conceptos, explicándolos en su papel de modelos básicos en el desarrollo de la inteligencia artificial. Sin entrar en detalles técnicos, también consideramos cómo el comportamiento "inteligente" que observamos como resultado de estas estructuras está basado en procesos de optimización combinatoria.

Introducción

Este artículo ofrece una introducción al perceptrón como estructura básica de la inteligencia artificial (que en adelante denotaremos con IA). El público principal al que está dirigido es a estudiantes de carreras relacionadas con matemáticas o computación. Aunque para comprender algunos conceptos se recomienda tener conocimientos de cálculo, álgebra lineal y probabilidad, buscamos que la mayor parte del contenido fuera accesible aun para estudiantes de bachillerato. Presentamos los diversos temas que abordamos primero en lenguaje coloquial e intuitivo, y después con un cierto rigor matemático. Al final del texto aparecen referencias para quienes deseen profundizar en algunos de los temas.

Iniciaremos presentando el *perceptrón* como fundamento de muchas tecnologías fundamentales de la IA. Pero, como veremos, éste tiene límites claros en su funcionamiento. Una vez que abordemos estos límites, explicaremos las particularidades del *perceptrón multicapa*, también conocido como *red neuronal* debido a su relevancia en el desarrollo de aplicaciones de clasificación más complejas.

La IA abarca áreas de gran actualidad, como la ciencia de datos, el aprendizaje automático y el procesamiento del lenguaje natural, entre otras. Resaltamos la

importancia de entender la matemática subyacente, buscando no usarla ciegamente o como mero formulismo. Las técnicas en IA emulan funciones biológicas y naturales, aunque no son capaces de pensar ni son, bajo ningún concepto, inteligentes. En esencia, la IA es una metodología que nos sirve para resolver problemas de optimización combinatoria. Estos problemas buscan reducir el espacio de búsqueda cuando el número de opciones es abrumadoramente grande. Considérese, por ejemplo, un juego de ajedrez: profundizamos más buscando las posibles jugadas derivadas de decisiones que nos signifiquen una ventaja medible inmediatamente (como prever lo que seguirá a capturar la reina del contrincante), al mismo tiempo que descartamos acciones que nos resulten claramente desventajosas (poner a nuestra reina en una posición vulnerable sin obtener ventaja alguna por ello), ahorrándonos considerar todas las derivaciones de dichas acciones.

Comenzando por el principio: el perceptrón sencillo

Santiago Ramón y Cajal, precursor de la neurociencia, postuló en el siglo XIX la "teoría de la neurona", indicando que las neuronas son unidades discretas (esto es, una gran cantidad de estructuras con aproximadamente el mismo funcionamiento en cada una de ellas), basada en conexiones sinápticas. Posteriormente, el modelo de Hodgkin-Huxley en la década de 1950 detalló cómo las señales eléctricas viajan por las neuronas, lo que les valió ser reconocidos con el premio Nobel de medicina en 1963. McCulloch y Pitts adoptaron las ideas de Ramón y Cajal en 1943, adecuándolas de conformidad con el modelo de Hodgkin-Huxley, y modelaron la primera *neurona artificial* en 1957.

Una neurona artificial, a la cual la literatura especializada denomina *perceptrón*, representa un modelo simplificado de neuronas cerebrales. No se derivó directamente de estudios neurológicos, sino de nociones básicas sobre neuronas y necesidades de computación.

Los perceptrones adoptan la idea de recibir múltiples entradas, procesarlas por cada una de sus terminales de entrada (*dendritas*) y generar una salida (Rosenblatt, 1958). Sin embargo, estas representaciones son simplificadas en comparación con las neuronas biológicas, como se muestra en la **Figura 1**.

Los primeros modelos artificiales surgieron en los años cuarenta y cincuenta, con representaciones binarias de neuronas (McCulloch y Pitts, 1943). El perceptrón mejoró esta idea, introduciendo ***pesos*** en las entradas, lo que permite variar el impacto que

Pesos
Factor multiplicativo que se da a la salida de cada perceptrón, modulando el impacto que tiene en el resultado global.

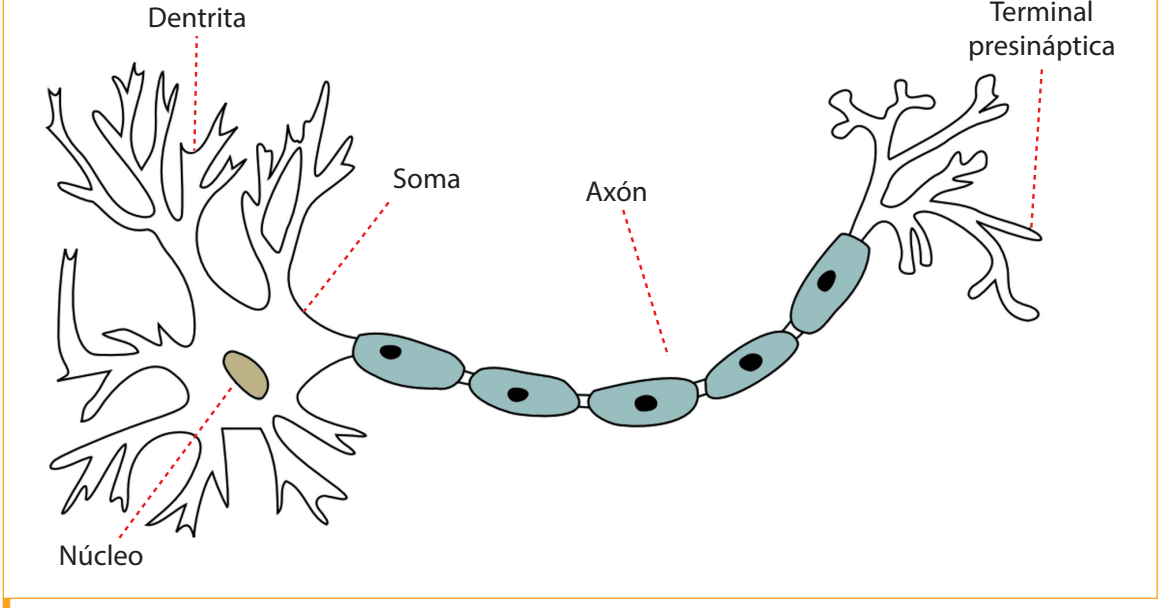

Figura 1. Una neurona biológica.

tiene cada una de ellas en el resultado obtenido. Como veremos en breve, el perceptrón, como fue planteado originalmente, tenía restricciones, como la de no poder resolver problemas que no fueran linealmente separables.

En la **Figura 2** se presenta el modelo básico de neurona artificial. Según Rosenblatt, una neurona artificial toma entradas, las pondera con pesos y genera una salida considerando un sesgo y una función de activación:

$$y = \sigma\left(\sum_{i=1}^{n} w_i x_i + b\right)$$

Un avance significativo en la evolución de las neuronas artificiales surgió con el desarrollo del algoritmo de *retropropagación* en los años ochenta (Rumelhart, Hinton y Williams, 1986). El método de retropropagación se basa en el modelo previamente descrito, donde las entradas se ponderan, suman y aplican a una función de activación, permitiendo entrenar redes multicapa y superar las limitaciones del perceptrón.

Más allá de obtener un 0 o un 1: las funciones de activación

Los primeros perceptrones entregaban como respuesta únicamente un valor binario, y la función de activación era simplemente un umbral; si la suma de todas las entradas ponderadas por los pesos resultaba mayor al umbral, el perceptrón se *encendía* (1); y en caso contrario, permanecía *apagado* (0). Sin embargo, el poder entregar una respuesta sobre un *rango* (o incluso un *continuo*) de valores permite modelar mucho mejor las soluciones a problemas de clasificación. Es por esto que comenzaron a utilizarse diversas funciones de activación. La **Figura 3** presenta algunos ejemplos de funciones frecuentemente utilizadas.

Las funciones de activación en un perceptrón (o cualquier otra red neuronal) juegan un papel crucial en la capacidad del modelo para representar y resolver problemas no lineales. Dichos problemas pueden implicar aquellos donde las coordenadas que generan salidas de *prendido* y *apagado* puedan ser explicadas con una división sobre el plano con una línea recta. Algunas razones para elegir funciones de activación son:

1. **Introducen no linealidad:** un perceptrón sin función de activación es lineal. Las funciones de activación permiten representar relaciones más complejas.
2. **Limitan la transformación:** algunas, como la sigmoide, limitan los valores de activación (puede apreciarse en la **Figura 3(a)**, en que la entrada sobre x puede ser arbitrariamente grande, pero la salida se

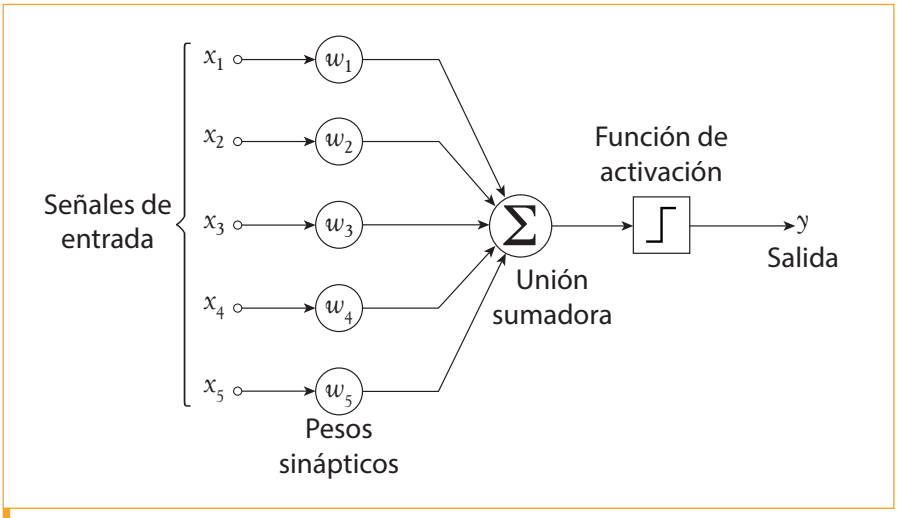

Figura 2. Modelo básico de un neurona artificial.

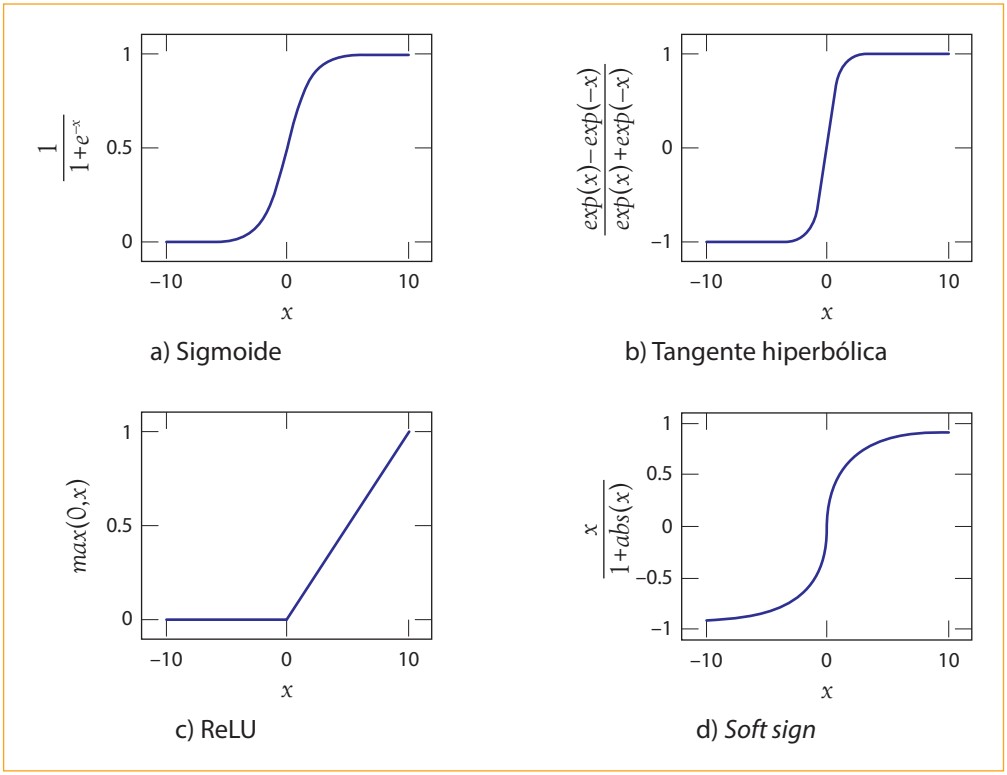

Figura 3. Algunas funciones de activación.

limita a valores entre 0 y 1), mejorando la estabilidad numérica.

3. **Son diferenciables:** esto permite usar un "descenso de gradiente" para ajustar los pesos de la red basándose en los errores entre los resultados esperados y los obtenidos.
4. **Mitigan sesgos:** funciones como ReLU previenen el *desvanecimiento del gradiente* en redes profundas.

En resumen, estas funciones son vitales para mejorar la capacidad de las redes neuronales. Por ejemplo, la función ReLU es popular por su simplicidad y por su capacidad para evitar que en algún momento el gradiente de la función de error se haga cero. Este problema es lo que se conoce como *desvanecimiento del gradiente*.

¿Y de dónde vienen los pesos? El entrenamiento del perceptrón

Para que un perceptrón entregue resultados que sean de utilidad, debe ser *entrenado* para entregar resultados útiles sobre un conjunto de datos de entrada. Para esto, puede utilizarse un proceso de *entrenamiento supervisado*. En este proceso, el perceptrón se entrena con ejemplos que indican las entradas (que representaremos con vectores x_i) y las salidas esperadas o etiquetas para el conjunto de datos ejemplo (que representaremos con números reales $\hat{y}_i$). Basado en Rosenblatt (1958) y Minsky y Papert (1969), el entrenamiento es iterativo y se resume en:

1. Establecer pesos y umbral inicialmente a valores pequeños.
2. Para cada objeto en el conjunto de entrenamiento, se calcula la salida de la neurona. Si es incorrecta, se ajustan los pesos y el umbral según la regla del perceptrón.
3. Los ajustes de pesos (w_i) y umbral siguen la fórmula:

$$w_i = w_i + \Delta w_i,$$

donde

$$\Delta w_i = \eta(y_i - \hat{y}_i)x_i.$$

η representa a la velocidad de aprende, o *learning rate* en inglés. Esta corresponde a la velocidad de ajuste del modelo. y_i es la salida del perceptrón con x_i como entrada y $\hat{y}_i$ es el valor esperado, o el valor correcto.

4. El proceso continúa hasta que la distancia entre la salida obtenida y la esperada, o el *error*, es mínimo (esto es, que la red neuronal clasifica, asignando alguna de las etiquetas o valores preestablecidos, de la forma esperada al conjunto de entrenamiento), o se alcanza cierto número preestablecido de iteraciones.

Si un conjunto de datos es *linealmente separable*, el perceptrón *converge* (esto es, que puede garantizarse que tras cierto número de rondas de entrenamiento se llega a un entrenamiento aceptable), lo que se conoce como el Teorema de Convergencia del Perceptrón: según Novikoff (1962), si existe un hiperplano correcto, el perceptrón lo encontrará; hablamos de *hiperplanos* porque los datos de entrenamiento se representan como simples puntos en un espacio de *alta dimensionalidad*. Sin embargo, muchos conjuntos reales no son linealmente separables, lo que limita la aplicabilidad del perceptrón.

Afinando el entrenamiento: función de pérdida y regularización

El entrenamiento del perceptrón busca minimizar la función de pérdida definida como:

$$L = \sum_{i=1}^{n} \max\left(0, -y_i \cdot f(x_i)\right)$$

donde x_i y y_i son la observación y la etiqueta, respectivamente, y $f(x_i)$ es la salida del perceptrón. Lo que hace esta función es contar el número de fallas del perceptrón en su tarea de clasificación. La meta de dicho entrenamiento es encontrar pesos que reduzcan esta pérdida. Si su mínimo es cero, el perceptrón clasifica todas las entradas correctamente (Bishop, 2006). Este tipo de problemas son el objeto de estudio de la optimización combinatoria y es lo que en esencia dota de "inteligencia aparente" al proceso.

En todas las aplicaciones de IA, el encontrar una solución "inteligente" se reduce a resolver un problema de optimización parecido. Es relevante recalcar que se trata de optimización incluso en aplicaciones de IA donde se obtiene como salida un texto o una imagen: el resultado generado es el que mejor se ajusta a los datos de entrenamiento; esto es, se optimiza para lograr un resultado congruente con el cuerpo de datos empleado para el entrenamiento.

Para evitar el "sobreajuste", donde el perceptrón clasifica bien las observaciones de entrenamiento, pero no las nuevas (podría entenderse como que una neurona *deja de aprender*), se introduce la *regularización*. Una técnica común es la regularización L_2 o *decaimiento del peso*, que añade un término de penalización basado en el tamaño de los pesos:

$$L_{\text{reg}} = L + \lambda \ \|w\|^2$$

Aquí, $\|w\|^2$ es la suma de los cuadrados de los pesos y λ regula la penalización. Esta regularización tiende a seleccionar modelos con pesos menores, reduciendo el riesgo de sobreajuste.

Tejiendo una red: el perceptrón multicapa

El perceptrón multicapa (MLP, por sus siglas en inglés) es una extensión del perceptrón simple con múltiples capas de neuronas conectadas entre sí (Bishop, 2006). En un MLP, los datos de entrada se dividen y dan a diferentes perceptrones (una capa) las salidas de esta capa, alimentan a otros múltiples (una segunda capa) y se arman estructuras de perceptrones. Estas capas incluyen entrada, salida y al menos una capa oculta. A diferencia del perceptrón simple, el MLP puede modelar relaciones no lineales, gracias a las funciones de activación no lineales en las capas ocultas.

El aprendizaje del MLP se realiza mediante *retropropagación*. Primero, se introduce un ejemplo de entrenamiento y se propaga para obtener una salida. Luego se compara la salida con la etiqueta verdadera

(ya conocida) para calcular la pérdida. Finalmente, el error se retro-propaga y se actualizan los pesos y sesgos, buscando minimizar la pérdida. Este proceso se repite múltiples veces hasta obtener la precisión deseada (Haykin, 2009).

Conclusiones

El perceptrón es la base de muchas redes neuronales actuales. Su extensión, el MLP, puede manejar datos no lineales. Gracias a las funciones de pérdida y regularización, el perceptrón ha evolucionado hacia estructuras avanzadas, como son las **redes recurrentes y convolucionales**. Los desafíos en estas arquitecturas a menudo se abordan mediante la comprensión del perceptrón básico.

Los perceptrones son meramente *bloques de construcción*, herramientas muy sencillas si se toman uno por uno. Sin embargo, al presentarse como redes neuronales multicapa se han convertido en el fundamento de buena parte de las técnicas de IA, y todo indica que seguirán dominando esta disciplina durante el futuro previsible.

En México se han utilizado los perceptrones multicapa en tiendas departamentales para clasificar a los clientes de acuerdo con el tipo de productos de su preferencia y canalizar de manera dirigida la publicidad que se les manda. En el Servicio de Administración Tributaria se han utilizado para localizar defraudadores fiscales y también se han aplicado para clasificar mensajes en la red X de acuerdo con su temática y sentimiento (si son mensajes positivos, negativos o neutros). Éstos son ejemplos de cuya existencia el lector seguramente ya se ha percatado, pero indudablemente se pueden localizar muchísimas más aplicaciones en contextos diversos.

Tzolkin Garduño Alvarado
University of Technology of Troyes, Francia.
tzolkin.garduno_alvarado@utt.fr

Feliú Sagols Troncoso
Departamento de Matemáticas, Cinvestav-Instituto Politécnico Nacional.
fsagols@math.cinvestav.edu.mx

Gunnar Wolf
Instituto de Investigaciones Económicas y Facultad de Ingeniería, Universidad Nacional Autónoma de México.
gwolf@gwolf.org

Redes recurrentes
Redes neuronales (o de perceptrones) que tienen un factor de memoria, mejorando la capacidad de predecir los siguientes elementos en una secuencia de datos.

Redes convolucionales
Redes de perceptrones multicapa estructuradas de forma que pueden aplicarse a matrices bidimensionales, especialmente útiles para el reconocimiento de patrones en imágenes.

Referencias específicas

Bishop, C. M. (2006), *Pattern Recognition and Machine Learning*, Cambridge, Springer.
Haykin, S. (2009), *Neural Networks and Learning Machines*, EUA, Pearson Education.
McCulloch, W. S. y W. Pitts (1943), "A Logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics* 5(4):115-133.
Minsky, M. y S. A. Papert (1969), *Perceptrons: An Introduction to Computational Geometry*, EUA, MIT Press.
Novikoff, A. B. (1962), "On the Convergence Proofs on Perceptrons", *Proceedings of the Symposium on the Mathematical Theory of Automata*, 12:615-622.
Rosenblatt, F. (1958), "The Perceptron: A Probabilistic Model for Information Storage and Organization in the Brain", *Psychological Review* 65(6): 386.
Rumelhart, D. E., G. E. Hinton y R. J. Williams (1986), "Learning Representations by Back-Propagating Errors", *Nature*, 323:533-536.