REDES NEURONALES EVOLUTIVAS
4. GRADUALISMO

José del Carmen Rodríguez Santamaría
https://www.ejristos.com
Bogotá, Colombia

May 20, 2025

Abstract

Investigamos las posibilidades de la Evolución en la construcción de redes neuronales que reconocen imágenes. En artículos anteriores, trabajamos con Redes Neuronales diseñadas por humanos que eran circuitos booleanos con tolerancia al ruido. La tarea de la Evolución consistió en especificar algunos parámetros. Nuestros resultados fueron muy modestos, un comportamiento posiblemente debido al carácter saltacionista y discontinuo de estos circuitos. En este artículo, aceptamos el reto de diseñar y estudiar el gradualismo, bajo el cual se responde con respuestas cercanas a estímulos cercanos.

Contents

1 MARCO TEÓRICO 3

1.1 Gradualismo vs Saltacionismo 4

2 NUESTRO PROPÓSITO 8

3 IMPLEMENTACIÓN DEL GRADUALISMO 9

4 CONCLUSIÓN 19

5 Respuestas a los ejercicios 20

6 LA BIBLIOGRAFÍA 27

1 MARCO TEÓRICO

Nuestro objetivo general es utilizar la Evolución para diseñar Redes Neuronales (RN) que clasifiquen imágenes. Trabajamos con modelos de juguete para obtener información y permitir realizar cálculos en computadoras comunes. Nuestras imágenes consisten en patrones dibujados en una cuadrícula de 3 x 3, cuyos píxeles son blancos o negros, y que, por lo tanto, pueden describirse mediante cadenas binarias de 9 bits. Estas se denominan **mapas de bits**.

Usamos la distancia BBB para medir distancias entre mapas de bits, la cual cuenta el número de sitios en que difieren. Por ejemplo, la distancia BBB entre 000 y 010 es 1, y 101 está más cerca del primer patrón que del segundo. Según el criterio del vecino más cercano, cada mapa de bits es una deformación bien de una imágen de 1 o bien de 0.

Nuestros perceptrones son compuertas booleanas generalizadas tolerantes al ruido y la variabilidad. En nuestra *Intro*, demostramos que una RN con dos perceptrones reconoce la mitad de los 512 patrones posibles. La Evolución ajustó conjuntos separados de diversos parámetros de esa RN. (Rodríguez, [8] 2024a). Las compuertas booleanas generalizadas son generadoras de saltos y discontinuidades, ya que dos entradas casi iguales pueden estar separadas por el umbral y, por lo tanto, las respuestas son extremadamente diferentes: en un caso, el perceptrón se activa, pero en el otro no.

Como con dos perceptrones pudimos acertar la mitad de los casos sin discordancias, aspiramos a alcanzar la perfección añadiendo algunos pares de perceptrones. Descubrimos que la perfección, clasificar correctamente los 512 mapas de bits, es alcanzable, pero con un costo de unos 250 perceptrones. Con todo, la Evolución tiene un pequeño poder para simplificar la RN (Rodríguez, [9] 2024b).

También aplicamos la antigua estrategia informática de "dividir y vencer", que divide un problema complejo en subproblemas, cuyas soluciones se combinan para generar una solución al problema original (geeksforgeeks, [4] 2024). Así, dividimos la cuadrícula de 3 x 3 en 3 filas. Esto provoca que cada mapa de bits se divida en tres submapas de bits que deben ser reconocidos por circuitos separados, los cuales se unen para producir una clasificación final de mapas de bits completos. Sin embargo, fallamos: la estrategia de "dividir y vencer" produjo peores resultados que el procedimiento inicial (Rodríguez, [10] 2024c).

1 ***Ejercicio.*** *Al estudiar un modelo de juguete, concluimos que las RNs con un gran número de entradas dirigidas a patrones grandes son más económicas y tienen mayor capacidad de reconocimiento que los conglomerados de redes neuronales pequeñas que apuntan a subpatrones. Esto se propuso como explicación de por qué las neuronas se obsesionan con un gran número de entradas o dendritas (Rodríguez, [10] lc, 2024c). ¿Considera que esta conclusión es firme y no debe revisarse en ningún caso?* Respuesta

Entremos ahora a discutir el tema del presente trabajo.

1.1 Gradualismo vs Saltacionismo

Todo ser humano sabe que un cambio con pocos detalles es más fácil que otro con un gran número de especificaciones que funcionan en su totalidad. La Evolución está cargada de la misma idea. En este sentido, la visión de Darwin merece ser citada. Veamos, en primer lugar, cómo intenta reducir las tareas complejas a una serie de pequeñas subtareas —una estrategia que en informática se denomina «divide y vencerás»—, y que, según nuestros propios resultados (Rodríguez, «rodriguez24c» lc, 2024c), no es del todo correcta:

Suponer que el ojo, con todos sus inimitables mecanismos para ajustar el enfoque a diferentes distancias, admitir diferentes cantidades de luz y corregir la aberración esférica y cromática, pudiera haberse formado por selección natural, me parece, lo confieso abiertamente, absurdo en grado sumo. Sin embargo, la razón me dice que si se puede demostrar la existencia de numerosas gradaciones desde un ojo perfecto y complejo hasta uno muy imperfecto y simple, siendo cada grado útil para su poseedor; si, además, el ojo varía mínimamente y las variaciones son hereditarias, lo cual es ciertamente el caso; y si alguna variación o modificación en el órgano es útil para un animal en condiciones de vida cambiantes, entonces la dificultad de creer que un ojo perfecto y complejo pudiera formarse por selección natural, aunque insuperable para nuestra imaginación, difícilmente puede considerarse real. (Darwin, 1857, pág. 186).

¿Qué podría ocurrir si la estrategia mencionada no funciona? Reconoce que esto sería fatal para su teoría, pero considera que ese no es el caso en biología:

Si se pudiera demostrar la existencia de un órgano complejo que no pudiera haberse formado mediante numerosas, sucesivas y sutiles modificaciones, mi teoría fracasaría por completo. Pero no he podido encontrar ningún caso similar. Sin duda, existen muchos órganos cuyos grados de transición desconocemos, sobre todo si consideramos especies muy aisladas, en torno a las cuales, según mi teoría, ha habido mucha extinción. O, si consideramos un órgano común a todos los miembros de una clase extensa, pues en este último caso el órgano debió formarse por primera vez en un período extremadamente remoto, desde el cual se han desarrollado todos los miembros de la clase; y para descubrir los primeros grados de transición por los que ha pasado el órgano, tendríamos que buscar formas ancestrales muy antiguas, extintas hace mucho tiempo. (Darwin, [2] 1857, pág. 189).

Observemos que la Evolución, al ser un relojero ciego e insensato (compárese con Dawkins, [3] 1986), está, en la visión de Darwin, cargada de una idea humana sobre la complejidad. ¿Es correcto?

Sí, absolutamente. Podemos justificar esta opinión de la siguiente manera:

Nosotros, así como la Evolución, ejecutamos tareas algorítmicamente, siguiendo instrucciones verbales en tándem: haz esto, luego aquello, luego esto otro,... Oficialmente, *el genoma es software y la Evolución es un desarrollador de software*. Gracias a la biología molecular, esta idea se asienta sobre bases sólidas. La Teoría Evolutiva (TE) ahora dice: *la Evolución es el desarrollador de software responsable de la aparición de todas las especies* (Rodríguez, [6], 2010). Adoptando esta postura, cada rama de un árbol evolutivo es una lista de modificaciones esperadas en tándem que la Evolución presumiblemente ejecutó al pasar de una especie a otra hasta la última, que está representada por una hoja.

La idea de que *el Genoma es software* y de *la Evolución como desarrolladora de software* permite idear una falsación directa de la Teoría Evolutiva si tan solo aceptamos el criterio establecido por Darwin:

La mayoría de los niños nacen perfectos, incluso en poblaciones tradicionalmente pequeñas. Esto también aplica a todas las poblaciones de seres vivos. De ahí la siguiente implicación: un grado de perfección tan alto como el que observamos en casi todos los seres vivos se ha conservado a pesar de las mutaciones, porque la información genética correspondiente se compone de paquetes gigantescos de instrucciones inmodificables. La consecuencia de una sola mutación es el colapso de funciones esenciales. En los mamíferos, se produce un aborto. Este tipo de fallo fatal también es típico en los

sistemas de software desarrollados por seres humanos. En biología, el núcleo genético inmodificable es responsable del concepto de especie. Junto a este núcleo inmodificable, existe un amplio margen de cambio, y por eso somos tan diferentes. Además, también existe un amplio margen de cambio en tandem, por lo que existen numerosas «pruebas» de la TE; por ejemplo, la Evolución de los virus de la COVID-19 muestra que algunas partes de su genoma son plásticas (Amoutzias [1] 2022).

Para pasar de la retórica a la ciencia, debemos realizar experimentos que conduzcan a predicciones para luego compararlas con las observaciones. Los experimentos informáticos con Evolución se denominan *Algoritmos Genéticos*, y cuando la Evolución se utiliza para diseñar software, tenemos *Programación Genética*.

Los dos resultados principales de nuestros estudios en Programación Genética son:

- La Evolución puede sintetizar software.
- Siempre se produce basura en volúmenes inmensos que aumentan cuando crece tanto la complejidad de la tarea como del lenguaje utilizado. (Rodriguez, [11] 2021d, [12] 2022e).

¿Qué es la basura en el desarrollo de software? Es software que no produce nada o bien productos defectuosos. En nuestros programas, nuestras simulaciones de la Evolución, la basura se reporta como el costo en individuos para llegar a la generación actual, especialmente en la que se llega a la perfección. Por lo tanto, **tenemos una predicción obligatoria**: *Si la vida en la Tierra es el resultado de la Evolución, entonces el registro fósil, así como nuestro propio cuerpo, deben estar lleno de imperfecciones, malformaciones y disfunciones que deben ser visibles desde el nacimiento.* **Esto es contrario a la observación**. Por ejemplo, tengo 71 años y, a pesar de todos mis problemas, me asombra la gran perfección de mi

mente y mis sentidos. Por eso, la TE es para mí obviamente falsa (compare con Rodríguez [7], 2021a). Se invita al lector a emitir el juicio correspondiente sobre su propio cuerpo y facultades.

En resumen: La TE es falsa, pero la Evolución es real. Y la Evolución aplicada es tan real como queramos. En particular, hemos demostrado que la Evolución puede ayudar en el diseño de RNs. Hasta ahora, solo hemos demostrado que la Evolución puede encontrar valores adecuados para conjuntos separados de parámetros de una RN de juguete.

Nuestras RNs han sido circuitos en cascada de compuertas booleanas tolerantes a fallos, un tipo específico de perceptrón o células neuronales artificiales. Hemos trabajado con circuitos de una o más capas que reconocen patrones en presencia de ruido y variabilidad. A pesar de haber trabajado con posiblemente el modelo de juguete más simple, con frecuencia nos hemos quedado cortos de memoria y paciencia. ¿Por qué? Suponemos que eso se debe a que los cambios en la entrada o en los parámetros pueden crear saltos en la salida y causar cambios desproporcionados en la aptitud. Nuestra Evolución es saltacionista. Es por eso que la Evolución no puede acumular pequeñas mejoras. Es fácil inferir que para reunir pequeñas mejoras, debemos recurrir al gradualismo, un tipo de Evolución en el que pequeños cambios en la entrada producen pequeños cambios en la salida y la aptitud. El gradualismo también requiere bajas tasas de mutación y recombinación, un tamaño de población suficientemente grande y una función de aptitud sin demasiados picos.

2 NUESTRO PROPÓSITO

Nos adherimos a la creencia generalizada de que la Evolución avanza rápidamente cuando los cambios permitidos en el genoma producen cambios suaves en la función de aptitud. Si no se implementa el grad-

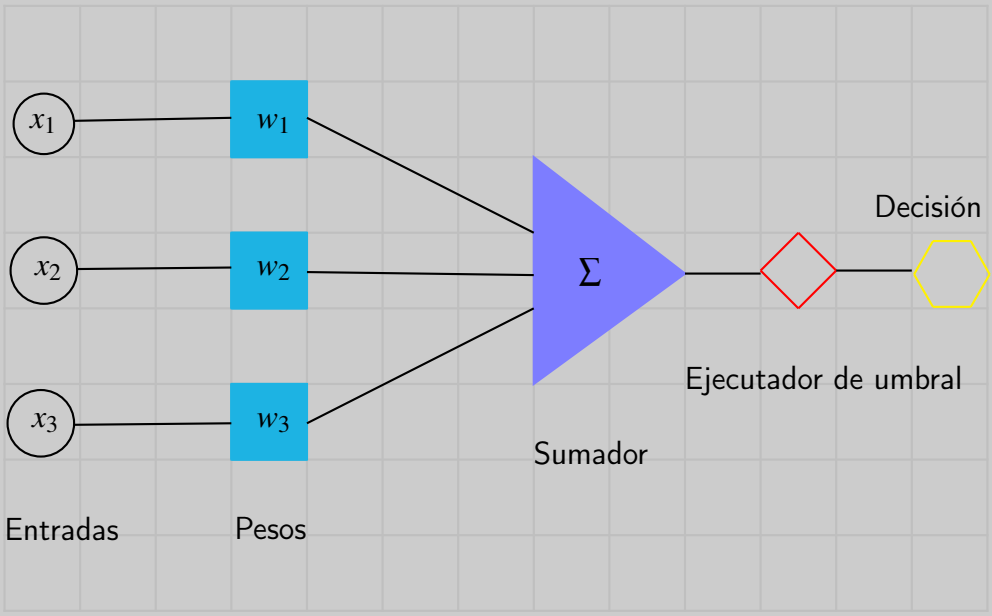
ualismo, la Evolución se frenaría. ¿Por qué? Bajo el gradualismo, los cambios son pequeños y resultan de pocos detalles, mientras que en el saltacionismo, la aptitud permanece en éxtasis durante demasiadas generaciones. Sin embargo, si el tiempo es suficiente, puede ocurrir un gran cambio favorable con numerosas especificaciones que funcionen en su totalidad. Sin embargo, el lapso de tiempo requerido puede ser mayor que la edad del universo. Por lo tanto, **nuestro propósito en este artículo es diseñar el gradualismo para la especificación de los parámetros de una RN y estudiar sus posibilidades**.

3 IMPLEMENTACIÓN DEL GRADUALISMO

Nuestra tarea ahora es detectar cualquier rastro de saltos en cualquier parte de nuestros perceptrones e implementar cambios suaves en todas partes. De nuestra *Intro*, sabemos:

Un perceptrón puede interpretarse naturalmente como una máquina que toma decisiones. Véase la figura 1. Según nuestra implementación hasta aquí, recibe entradas binarias, x_i, que indican si existe un factor determinado, 0 si no existe, 1 si existe. Cada entrada se multiplica por un número que representa su importancia, w_i. Los resultados de las multiplicaciones se suman para obtener $S = \sum w_i x_i$. Para obtener estabilidad frente al ruido y la variabilidad, añadimos un valor positivo, b. Obtenemos $S = \sum w_i x_i + b$. Si la suma S supera un umbral T, la decisión es afirmativa; de lo contrario, es negativa.

En resumen, obtenemos $S = \sum w_i x_i + b$ como entrada para el ejecutor de umbral, que compara el valor de S con el umbral T: si $S \geq T$, la decisión es afirmativa; de lo contrario, es negativa. En la literatura, el valor de desplazamiento b se denomina **sesgo**.



j

Figure 1: Esto es un perceptrón. Es una abstracción de una neurona que puede interpretarse naturalmente como una máquina que toma decisiones. Supongamos que necesitamos decidir si vamos a continuar leyendo o no un artículo. Tenemos como entrada al perceptrón la presencia o ausencia de algunos factores, x_i, digamos, LA LECTURA ES APARENTEMENTE MUY IMPORTANTE PARA CONSTRUIR UNA COSMOVISIÓN, y ES INTERESANTE. Los factores tienen diferentes pesos w_i para influir en la decisión. Entonces, las entradas se multiplican por sus pesos y sus valores multiplicados se suman. Para mejorar la estabilidad del perceptrón frente al ruido y la variabilidad, se añade una constante b a la suma, para obtener $S = \sum w_i x_i + b$. Si la suma S supera un cierto umbral T, la decisión es afirmativa; de lo contrario, es negativa. Por lo tanto, la decisión es afirmativa si $S \geq T$, de lo contrario, es negativa. Debemos atacar las siguientes características para implementar el gradualismo: 1) los valores de los factores x_i son dicotómicos, 1 = presente de lo contrario 0 = ausente, 2) los pesos son impuestos por compuertas booleanas, 1 de lo contrario 0. 3) El sesgo es un número entero, 4) La decisión $S \geq T$ es SÍ de lo contrario NO. 5) Conjeturamos que el umbral es posiblemente la fuente más importante de saltos.

Los saltos son sinónimos de discontinuidades y presumiblemente responsables del fracaso posterior de la Evolución:

- La presencia o ausencia de un estímulo es dicotómica: existir o no existir. Esto significa que los píxeles de nuestros gráficos son blancos o negros.
- Para simular compuertas booleanas, los pesos son 1, de lo contrario 0.
- El sesgo es un valor entero.
- Para reproducir compuertas booleanas, se puede asignar un sesgo adicional a cada factor. Puede tomar valores de 0 o -1.
- Hay un umbral que responde de forma bastante diferente a entradas casi iguales.
- Una decisión puede ser SÍ o NO y preferimos interpretarla como SÍ o DESCONOCIDO.

Así pues, nuestros perceptrones son estupendos creadores de saltos.

Remedios para probar:

- Hemos trabajado con mapas de bits, la representación digital de un dibujo, cuyos píxeles son blancos o negros. Ahora, podríamos probar con valores de gris. Los píxeles de dicho gráfico se representan con números en el intervalo [0,1] para representar diversos tonos de gris, desde 0 para el blanco hasta 1 para el negro.
- En lugar de números enteros, los pesos toman valores reales, es decir, en el intervalo $[-\infty,\infty]$. Por razones prácticas y para empezar, reemplazaremos toda la línea real por el intervalo $[-99,99]$ y trabajaremos con 2 cifras significativas. De esa manera, un número se verá como, digamos, -13.24. El signo

se codificará como el primer dígito, si es par, el signo es más, si es impar, el signo es menos. Por ejemplo, la cadena 12345 representa el número -23.45 pero la cadena 82345 representa el número 23.45. El punto decimal no se codificará porque siempre estará en el mismo sitio. Esto significa que los factores no son simplemente aceptados o rechazados, sino que pueden tener pesos relativos que pueden interpretarse, por ejemplo, desde el rechazo hasta el afecto con cada grado de variación.

- El sesgo toma valores reales en lugar de números enteros.
- No se incluyen sesgos adicionales, aquí son redundantes.
- Tenemos un gran problema con el umbral que provoca que la RN responda de forma muy diferente a 4.9 que a 5, ya que la RN solo puede responder SÍ o NO. Sospechamos que esta es la principal causa del caos que destruye el cambio favorable acumulado y convierte la Evolución en una frustración debido a la sobremultiplicación de los recursos demandados, incluido el tiempo de CPU. Nuestra solución es la regla TRIZ: *la mejor solución es ninguna solución*. Por lo tanto, investigaremos qué sucede si eliminamos los umbrales y decidimos que una *célula o perceptrón se activa como respuesta a una entrada dada cuando su salida es la mayor en comparación con todos los perceptrones o células en la misma capa, y también definimos que las demás células no importan, por lo que sus respuestas se toman como* 0 *o DESCONOCIDAS*. Con esta regla, las células no pueden contradecirse entre sí. Esta regla se inspiró en la función ReLu de la teoría ordinaria de las RNs (Nielsen [5] 2018).

La forma recién anunciada de Gradualismo se implementa en el programa `EvolNNGradualism.html`.

2 *Ejercicio.* *Ejecute el programa* `EvolNNGradualism.html` *para comprobar la conclusión del Autor: esta forma de gradualismo*

resuelve todos los problemas con la ayuda de tan solo dos perceptrones. Permite que la Evolución alcance la perfección (512 coincidencias de 512 opciones), sin desajustes, generalmente antes de 2000 generaciones con una población de 100, y que la mantenga indefinidamente. Además, los recursos utilizados se mantienen dentro de los límites establecidos. Respuesta

Hemos trabajado con modelos saltacionistas y hemos descubierto que el coste de ir más allá de las soluciones mediocres es intolerablemente alto. Y aquí, de repente, surge el gradualismo y exhibe rápidamente una perfección que, además, se mantiene eternamente.

3 ***Ejemplo.*** *La afirmación de que el gradualismo es milagroso es seria. ¿Cómo puede asegurarse de que su software sea correcto y de que esta pretensión no sea una falsa ilusión?*

Trabajamos con un modelo de juguete, un problema sencillo que puede estudiarse con una computadora común. Los resultados demuestran claramente que la Evolución no es mágica, sino real. Se observa cómo gana aptitud en pasos relativamente pequeños, que pueden ocurrir tras largos tiempos de espera, hasta alcanzar la perfección, algo que con demasiada dificultad podíamos vislumbrar en el anterior entorno saltacionista. Consideramos que este comportamiento es tan especial que apostamos a que el software es correcto y simplemente cumple lo que promete.

En cualquier caso, consideramos que la corrección y depuración de software son tareas esenciales que deben ser accesibles para todos. Por lo tanto, se pueden ver varias variables booleanas después de la línea 291 en el programa `EvolNNGradualism.html` que permiten elegir métodos o procedimientos específicos para imprimir sus resultados. De esta manera, se puede ver el resultado de muchos métodos y comprobarlo.

La implementación del gradualismo es lo más importante para este artículo. Por ello, se diseñó una prueba específica, cuyo resultado

puede consultarse en un área de texto independiente, TEST. La implementación de esta prueba aparece en el programa `EvolNNGradualism2.html`.

4 *Desafío.* *Resuelva de una vez por todas la siguiente queja: la implementación actual del gradualismo requiere un programa completo para comprobar su corrección. ¿Es eso transparencia? No, en absoluto. Haga algo mejor.*

5 *Ejemplo.* *Nuestro Gradualismo es incompleto porque necesita que le digan cuál es el valor máximo entre un par de valores reales. Esto se hizo con unas pocas líneas de código TypeScript, que corresponde a recibir ayuda de una máquina de Turing. Para completar nuestro Gradualismo, vamos a mostrar que una RN puede encontrar el valor máximo entre dos números reales. De esa manera nuestra RN se convierte en Gradualista plena.*

Supongamos que tenemos dos números, x e y. Para saber si x es el máximo entre los dos, se resta $x-y$. Si el resultado es positivo, x es el valor máximo. Si es negativo, x no es el valor máximo. Esto se puede reescribir como $x-y \geq 0$ que es la determinación directa de un perceptron cuyos pesos son 1 y -1, su sesgo es 0, y el umbral es 0. Vamos a denotar este perceptrón como el subtractor $X-Y$.

Una RN que encuentra el valor máximo entre dos números, x y y, se construye de la siguiente manera: hay dos entradas, x e y, y dos luces de salida, X e Y. La primera se enciende cuando x es el máximo,la segunda cuando y es el máximo. Entre las entradas y la salida tenemos dos perceptrones, el primero es $X-Y$, el segundo es $Y-X$. Cada subtractor tiene dos pines de entrada y uno de salida. Las conexiones se muestran en la figura 2.

6 *Desafío.* *Diseñe una RN que encuentre el máximo entre tres números reales. Generalice a n números.*

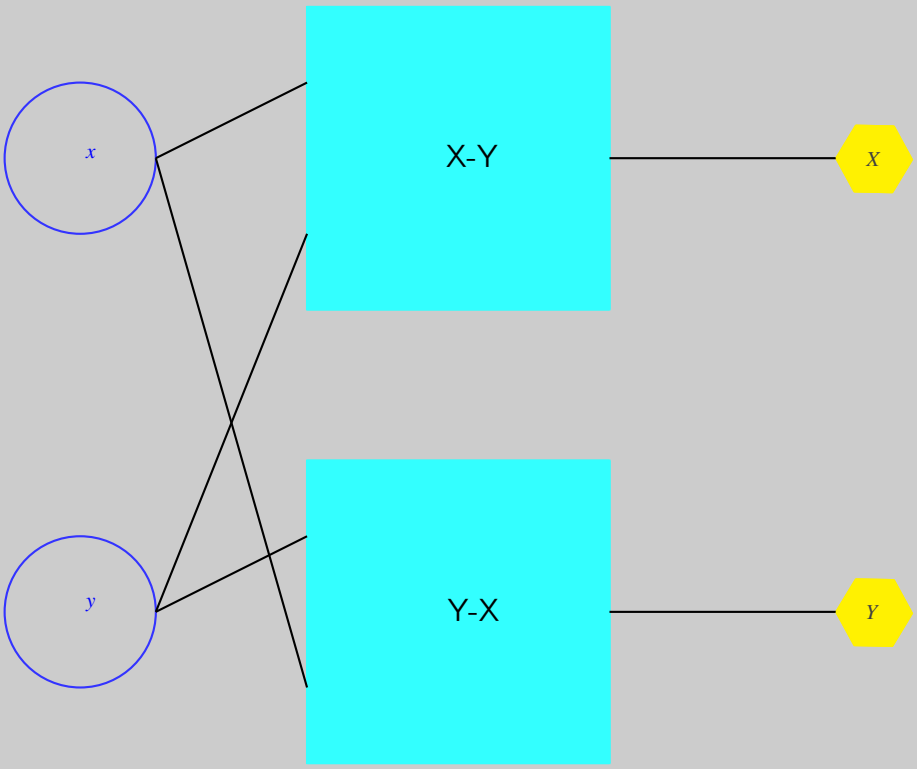


Figure 2: Esta RN encuentra el máximo entre dos números x y y. La RN tiene dos perceptrones. El perceptrón superior debe dispararse cuando x es el máximo, el inferior cuando y es el máximo. Los dos perceptrones son de la misma naturaleza, sustraen el número que llega al pin inferior del que llega al pin superior y comparan el resultado con el umbral 0. Las conexiones son tales que el perceptrón superior ejecuta $x-y$ mientras que el inferior $y-x$ y compara el resultado con el umbral 0. Por lo tanto, los perceptrones superiores se disparan cuando la resta es positiva porque x es el mayor, así que $x-y \geq 0$, mientras que el inferior se dispara cuando y es el máximo. Cuando las entradas son iguales, los dos perceptrones se disparan.

7 Ejemplo. *Los ingredientes de nuestra implementación del gradualismo son: convertir todos los parámetros en variables reales y quemar el umbral. Hemos sostenido que la quema del umbral es el factor más importante que impulsa el gradualismo. Para comprobar esta afirmación, construimos una RN con parámetros reales, pero con un umbral que fijamos en cero. A continuación, la estudiamos para ver qué sucede. Nuestra solución se ha empaquetado en el programa EvolNNGradualism3.html. Expliquemos por qué podemos fijar el umbral en cero:*

Para cada perceptrón, calculamos la suma S dada por $S = \sum w_i x_i + b$. Si la suma S supera un umbral T, si $S \geq T$, la decisión es afirmativa; en caso contrario, es negativa. Por lo tanto, se trata de la desigualdad:

$\sum w_i x_i + b \geq T$

donde b y T deben ser especificados por el Proceso Evolutivo. Esto es equivalente a

$\sum w_i x_i + b - T \geq 0$

Dado que b y T son incógnitas que no están relacionadas con los otros parámetros, podemos cambiar el nombre de la incógnita $b - T$ como b, para obtener

$\sum w_i x_i + b \geq 0$

Es por ello que hemos fijado nuestro umbral en 0.

¿Qué obtuvimos?

El programa `EvolNNGradualism3.html` se ejecutó durante 20000 generaciones con una población de 100. JavaScript se sobrecargó con la tarea de impresión debido a la necesidad de reportar los resultados durante 20000 generaciones. El resultado del experimento fue simple: la Evolución no logró encontrar ni una sola coincidencia, ni un solo resultado positivo. Esto nos sorprendió y nos planteó la duda: ¿es correcto el programa?

Consideramos que este programa es correcto porque es una ligera variación del programa `EvolNNGradualism.html` que muestra cómo el gradualismo funciona con gran éxito cuando no hay umbral. Pero para ampliar la ventana de observación, hicimos algunos cambios: la tasa de mutación se movió de 0,0001 a 0,1. Fracaso. La tasa de mutación se movió a 0,3. Fracaso. El nuevo programa se ejecutó durante 100000 generaciones. Fracaso. La codificación se cambió de 12,34 a 123,4. Fracaso. Estos cambios se encadenaron. Por último, ejecutamos el programa durante 100000 generaciones con una tasa de mutación de 0,0001 y un tamaño de población de 100. La tasa de mutación se eligió tan baja porque, según resultados previos, es necesaria una tasa de mutación baja para alcanzar y mantener la perfección. Fracaso.

Interpretamos este fallo recurrente como una prueba de que el umbral es la causa de todos los problemas.

8 ***Ejercicio.*** *El programa EvolNNGradualism3.html. no pudo encontrar una valoración satisfactoria de los parámetros de una RN con parámetros reales. ¿Por qué? Vemos una disyuntiva: Esto se debe a que no existe una valoración satisfactoria o a que su probabilidad es demasiado baja. En consecuencia, demuestre que la RN del programa EvolNNGradualism3.html, que tiene dos perceptrones de parámetros reales y un umbral, no puede satisfacerse de ninguna manera; es decir, no existe ninguna valoración de parámetros con al menos una coincidencia. De lo contrario, defina un conjunto de valores de sus parámetros tal que la RN encuentre al menos una coincidencia; es decir, que pueda clasificar correctamente al menos un mapa de bits, pero sin discordancias.* *Answer*

9 *Ejercicio.* *Hemos dicho que el principal obstáculo al gradualismo es el umbral. Por lo tanto, hemos menospreciado la codificación de los parámetros como números reales. Ponga a prueba esta creencia negativa.* Respuesta

10 *Ejercicio.* *Resuelva la siguiente queja, que se asemeja más a un ataque: El gradualismo implica la acumulación de pequeñas mejoras en la aptitud debido a pequeños cambios en el genoma. El gradualismo parece garantizar que la Evolución alcanzará un óptimo local con bastante rapidez. Sin embargo, aquí, la cadena* `01234` *representa el número real* 12.34*, mientras que la cadena* `11234` *representa* -12.34*. Por lo tanto, pequeños cambios en los cromosomas causan grandes cambios en el valor de los números reales asociados, lo que puede causar un cambio drástico en la aptitud o la función de puntuación cuando los perceptrones tienen un umbral. En consecuencia, la Evolución no funciona cuando se incluye el umbral.* Respuesta

11 *Ejercicio.* *Demuestre claramente que nuestro Gradualismo no tiene nada que ver con la Evolución Natural ni con la Teoría Evolutiva del Origen de las Especies.* Respuesta

12 *Ejercicio.* *Justifique o refute el siguiente razonamiento:* Todos los experimentos, tanto los nuestros como los de otros autores, abordan la Evolución dirigida, en la que existe un objetivo. Dado que estos no tienen nada que ver con la Evolución natural, la ciencia no tiene nada que decir respecto a la refutación de la Teoría Evolutiva. En resumen, la Teoría Evolutiva es infalsable. Respuesta

13 *Graduación.* *Hemos aprendido que la codificación con el formato* 12.34 *contribuye al gradualismo, pero no demasiado. Pensamos al principio que las valoraciones satisfacibles son escasas. Sin embargo, descubrimos que la codificación* 123.45 *era mucho mejor,*

ya que, a partir de una valoración satisfacible, el gradualismo puede lograr una mejora significativa. De estos resultados, inferimos que las valoraciones satisfacibles son numerosas, pero no se encontraron debido a la falta de expresividad de la codificación con el formato 12.34*. Demuéstrelo; es decir, encuentre una codificación que permita que la Evolución alcance rápidamente la perfección.*

4 CONCLUSIÓN

Una Red Neuronal Gradualista de juguete funciona tal como Darwin lo esperaba: la Evolución acumula pequeños avances hasta alcanzar una perfección estable, poco a poco, pero quizás tras largos tiempos de espera. Además, hemos demostrado la afirmación intuitiva de que el umbral de los perceptrones es la principal causa de problemas para la Evolución. De hecho, el umbral genera caos porque provoca que el sistema responda de forma muy diferente a entradas casi iguales. Advertencia: la Evolución dirigida tiene un objetivo y una recompensa definidos con precisión, pero la Evolución natural no tiene ningún propósito, objetivo ni recompensa predefinida. En consecuencia, los resultados de las simulaciones no se pueden extrapolar a la naturaleza. Sin embargo, la Evolución sin objetivos es más lenta y produce más basura que la dirigida, así que tenemos razón siempre que decimos que nuestras simulaciones refutan la Teoría Evolutiva debido a los inmensos volúmenes de basura esperados, que, al referirse al desarrollo de software, se representan por la ausencia de resultados o por productos defectuosos, es decir, extinciones, malformaciones y disfunciones, tanto en el registro fósil como en nuestros propios cuerpos.

5 Respuestas a los ejercicios

1, página 4. La conclusión más contundente a la que hemos llegado hasta ahora es que las RN son entidades complejas. Por lo tanto, toda extrapolación debe justificarse cuidadosamente. En cuanto a nuestra implementación de la estrategia de "divide y vencerás" sobre posiblemente el modelo de juguete más simple, la complejidad nos impide generalizar. Esto es una clara invitación a la investigación.

2, página 12. El programa `EvolNNGradualism.html` se ejecutó en una PC con 16 GB de RAM. El programa necesitó unos veinte minutos para ejecutarse durante 20000 generaciones con los siguientes parámetros:

```
TAMAÑO POP = 100,
MUT_RATE = 0.0001, //por carácter
REC_RATE = 0.1,
```

La tarea era optimizar los parámetros de la RN, pesos y sesgos de cada uno de los dos perceptrones, para lograr el número máximo de coincidencias sin desajustes.

Seguimos el comportamiento del mejor de cada generación durante una simulación. Su puntuación crece a saltos bastante pequeños. Podemos distinguir dos etapas. En la primera, las mejoras son comunes y rápidas. En la segunda, en la generación 236, aparece un gran intervalo de tiempo de espera.

La perfección, en lo que respecta al mejor de cada generación, se alcanzó en la generación 1593 y se mantuvo hasta la generación 20000, cuando el programa se detuvo.

La siguiente tabla muestra algunos detalles:

```
Generation score   matches mismatches       cost
     0     -364       366        146        100
   236      320       480         32      23700
  1568      386       491         21     156900
  1586      392       492         20     158700
  1593      512       512          0     159400
 20000      512       512          0    2000100
```

Si se repite el experimento, se podría alcanzar la perfección mucho más rápido, y en otros casos, más lento. La perfección se conserva.

8, página 17. Demostremos que la arquitectura utilizada admite una valoración con cientos de coincidencias. Esta valoración se deriva de los resultados publicados en la Introducción. En ese caso, usamos números enteros, pero aquí usamos números reales. No hay conflicto, ya que los números enteros también son números reales. Esto demuestra que existen valoraciones que producen coincidencias, pero son tan escasas que no se garantiza encontrar una por mera aleatoriedad, incluso añadiendo mutación y recombinación.

Según nuestra introducción, el perceptrón P1 tiene los siguientes parámetros

```
La  RN tiene dos perceptrones, Perceptrón1, P1,
y Perceptron0, P0.

P1, tenía los siguientes parámetros:

entradas (el mapa de bits de 1) = [0, 1, 0, 0, 1, 0, 0, 1
pesos = [-1, 1, -1, -1, 1,-1,-1, 1,-1];
umbral = 9;
sesgo = 9.8; //
```

En el gradualismo considerado aquí, nuestros cromosomas se codifican en el siguiente orden: 8 pesos y sesgo. Cada parámetro se

codifica como una subcadena de 5 dígitos. El primer entero codifica el signo, + si el dígito es par, -1 si es impar. Los otros cuatro enteros representan el valor del parámetro, que puede tener dos formatos: 12,34 o 123,4. Esta elección se puede realizar en el método `descipherSubstring(s)` en la línea 1250 del programa `EvolNNGradualism4.html`. Se estableció un umbral en cero.

Los pesos de P1 son:

```
-1 1 -1 -1 1 -1 -1 1 -1
10100001001010010100001001010010100001001010
```

El sesgo de este programa se calcula restando el umbral del sesgo reportado en la Introducción, ya que el umbral es absorbido por el sesgo. Es 9,8 - 9 = 0,8. Por lo tanto, la cadena asociada es `00080`.

Concatenando esta cadena corta con la larga, obtenemos el primer semicromosoma:

```
1010000100101001010000100101001010000100101000010010100
```

El otro semicromosoma se encuentra de la siguiente manera. A continuación, se presenta la información reportada sobre P0:

```
La  RN tiene dos perceptrones, Perceptrón1 y Perceptrón0

Perceptron0, P0, tenía los siguientes parámetros:
entradas (el mapa de bits de 0) = [1, 1, 1, 1, 0, 1, 1, 1
pesos = [1, 1, 1, 1, -1, 1, 1, 1, 1];
umbral = 9;
Sesgo total = 4,8.
```

Los pesos de P0 y sus cadenas asociadas son:

```
1 1 1 1 -1 1 1 1 1
00100 00100 00100 00100 10100 00100 00100 00100 00100
```

El sesgo reportado es de 4,8 y el umbral de 9; por lo tanto, el sesgo actualizado es 4,8-9 = -4,2, codificado como `10420`. Concatenando todo, obtenemos el subcromosoma para P0:

`0010000100001000010010100001000010000100001000010010420`

Los dos subcromosomas asociados con P1 y P0 se unieron y la cadena final se clonó para usarse como población inicial de cromosomas en nuestra simulación de gradualismo. Esto se realizó en el programa `EvolNNGradualism4.html`, en el método `createPop()`, líneas 1088-1093.

Para que el programa `EvolNNGradualism4.html` sea correcto, debe cumplir una predicción: los parámetros dados deben hacer que la RN alcance 260 coincidencias con 0 desajustes de 512 instancias. Este programa debe reproducir exactamente estos resultados; de lo contrario, es incorrecto. Además, este programa es una ligera variación del programa `EvolNNGradualism3.html`. Por lo tanto, si este es incorrecto, el otro también lo es. Y si este es correcto, el otro también. Recuerde que `EvolNNGradualism3.html` reveló un fracaso de nuestro gradualismo para superar el caos total creado por el umbral.

Resultado: el programa cumple la expectativa: reprodujo 260 coincidencias con 0 desajustes. Por ello, consideramos que estos programas son correctos, lo que refuerza nuestra creencia de que el umbral es la causa de todos los problemas, es decir, la incapacidad de Evolución para encontrar una valoración con un gran número de coincidencias con 0 desajustes.

9, página 18. Ya hemos dicho por qué creemos que el programa `EvolNNGradualism4.html.` es correcto: este programa incluye un umbral en 0, y reproduce exactamente la respuesta esperada que se dio en la codificación de los parámetros cuyos valores se encontraron en nuestra Intro. La RN resultante produce 260 coincidencias bajo 0 desajustes de un total de 512 opciones. Esto significa que hay 512 - 260 = 252 incógnitas. Ese programa se ejecutó con

una población de 2 individuos durante 0 generaciones, es decir, solo se calculó el comportamiento inicial. Y se cumplieron las expectativas. Para evaluar el efecto de la codificación de parámetros como números reales, ejecutamos el mismo programa durante muchas generaciones para ver qué sucede: ¿El gradualismo mejora la calificación de nuestra Intro? Para facilitar la ejecución por parte del lector, el cambio aparece en el programa `EvolNNGradualism5.html`.

¿Qué resultados positivos obtuvimos?

El programa se ejecutó con una población de 100000, una tasa de mutación de 0,0001, una tasa de recombinación de 0,1 y un número de generaciones de 100. En la generación 50, el programa informó lo siguiente:

```
Generación = 50 bestSeq = 10100001001010010100001901010001
vPesosP1 = -1,1,-1,-1,1.9000000000000001,-1,-1,0,-1
sesgoP1 = 0.8
vPesosP0 = 1,1,1,1,-1,1,1,1,1
sesgoP0 = -4.2
puntuación = 274
número de coincidencias = 274
número de desajustes = 0
Costo = 510000
```

Esto representa una mejora de 260 a 274 coincidencias con 0 desajustes. Esto indica que la codificación de valores como variables reales es algo rentable, y por lo tanto, no es despreciable. En cualquier caso, la liberación del umbral es el ingrediente más importante del gradualismo, ya que permitió pasar de 260 coincidencias a 512, lo cual es la perfección.

10, página 18. Proponemos la siguiente solución al problema del signo para la Evolución de una RN de juguete cuando se incluye un umbral en 0. Primero, se utiliza la codificación sin signo 123.45,

que funciona para números de 0 a 999.99. Para recuperar el signo, se resta 500 del número decodificado. Por ejemplo, la cadena 12345 codifica para el número real 123.45, que luego se desplaza al número negativo $123.45 - 500 = -376.55$. Esto se realizó en el programa `EvolNNGradualism6.html`.

¿Qué obtuvimos?

Ejecutamos el programa durante 100 000 generaciones con una población de 100 y una tasa de mutación de 0,0001. Esto se realizó durante la noche, ya que una CPU dedicada necesitaba 44 segundos para ejecutar 100 generaciones y, por lo tanto, más de 12 horas para ejecutar 100000.

Nuestro gradualismo en los parámetros y la codificación se reflejó en el crecimiento de la puntuación, que combina una recompensa por las coincidencias y un castigo por las discrepancias. El programa reportó la puntuación máxima y las coincidencias y discrepancias correspondientes. El número máximo de coincidencias alcanzó 432 para la generación 53200 y se mantuvo así hasta la generación 100001. Durante el proceso evolutivo, el número de discrepancias se mantuvo bajo, pero al final resultó ser cero. El número de coincidencias aumentó a saltos, que fueron escasos pero grandes al principio, por ejemplo $0 \rightarrow 56 \rightarrow 140 \rightarrow 152 \rightarrow 276$ en 33600 generaciones, pero luego se desaceleró en magnitud, ejemplo $375 \rightarrow 380 \rightarrow 381 \rightarrow 384$ durante aproximadamente 8000 generaciones.

El comportamiento actual con nuestra RN de juguete se describe simplemente de la siguiente manera: *En el gradualismo diseñado, pequeñas mutaciones producen cambios en las variables de salida, que son relativamente grandes pero escasos al inicio del proceso evolutivo para luego desacelerarse cerca de un óptimo local.* ***La Evolución de la perfección es claramente visible.*** *Si la tasa de mutación es baja, el óptimo local se conserva durante un largo período de tiempo.* La única razón por la que aquí hablamos de un óptimo local y no global es que desconocemos la identidad del óptimo global. Sin umbral, es 512. Pero con un umbral como este, desconocemos cuál

es el número máximo posible de coincidencias sin un solo desajuste.

11, página 18. Estamos desarrollando Evolución Dirigida Aplicada; es decir, tenemos un objetivo y reunimos todo lo necesario para alcanzarlo. La diferencia con la Evolución Natural es que no tiene objetivo. ¿Cuál es el problema que surge? En que no hay suficientes recursos para producir algo útil. Imaginemos un barco que se coloca en un lugar aleatorio del océano y se deja a la deriva por el viento y las olas. ¿Alguien espera que llegue a puerto? Esa persona tiene demasiada fe. Creemos que la Evolución sin objetivo es posible a costa de desperdiciar recursos casi infinitos.

12, page 18. Según nuestras simulaciones, no se puede responsabilizar a la Evolución Dirigida de la aparición de la vida y las especies en la Tierra porque el tiempo permitido es insuficiente. Este veredicto va acompañado de esta otra observación: la Evolución Dirigida produce demasiada basura. De esto deducimos inmediatamente que la Teoría Evolutiva es falsa porque la Evolución Natural no tiene objetivo y, por lo tanto, es más lenta y produce más basura que la Evolución dirigida. Si bien esta afirmación es intuitivamente aceptable, demostrarla explícitamente es un desafío magnífico. Mientras tanto, aclaremos que en relación con el desarrollo de software por parte de la Evolución Natural, producir inmensos volúmenes de basura implica que, aparte de tener extinciones comunes, el registro fósil, junto con nuestros cuerpos, debe estar lleno de malformaciones y disfunciones.

6 LA BIBLIOGRAFÍA

References

[1] Amoutzias GD, Nikolaidis M, Tryfonopoulou E, Chlichlia K, Markoulatos P, Oliver SG. The Remarkable Evolucionary Plasticity of Coronaviruses by Mutation and Recombination: Insights for the COVID-19 Pandemic and the Future Evolucionary Paths of SARS-CoV-2. Viruses. 2022; 14(1):78. https://doi.org/10.3390/v14010078 7

[2] CHARLES DARWIN, M.A. 1859
THE ORIGIN OF SPECIES BY MEANS OF NATURAL SELECTION, OR THE PRESERVATION OF FAVOURED RACES IN THE STRUGGLE FOR LIFE. LONDON: JOHN MURRAY, ALBEMARLE STREET.
http://darwin-online.org.uk/content/frameset?itemID=F
5

[3] Richard Dawkins 1986
The Blind Watchmaker: Why the Evidence of Evolución Reveals a Universe without Design. Norton & Company, Inc
https://terebess.hu/keletkultinfo/The_Blind_Watchmaker
6

[4] Divide and Conquer Algorithm (2024)
https://www.geeksforgeeks.org/divide-and-conquer/
4

[5] Michael Nielsen (2018)
Neural Networks and Deep Learning .Determination Press 2015.
http://neuralnetworksanddeeplearning.com/

Code samples for "Neural Networks and Deep Learning"
https://github.com/mnielsen/neural-networks-and-deep-l
12

[6] José Rodríguez (2010)
The genome is software and evolution is a software developer .
http://arxiv.org/abs/1003.0575 6

[7] José Rodríguez (2021a)
Tomemos en serio a Darwin.
https://ejristos.com/dl/DarwinEnSerio.pdf
8

[8] José Rodríguez (2024a)
Redes neuronales evolutivas 1. Introducción.
https://www.ejristos.com/Evol%20Neural%20Networks/EvolNNintroSpanish.p
3

[9] José Rodríguez (2024b)
Redes neuronales evolutivas 2. Simplificación.
https://ejristos.com/Evol%20Neural%20Networks/EvolNN2Spanish.pdf
3

[10] José Rodríguez (2024c)
Redes neuronales evolutivas 3. Divide y vencerás.
https://www.ejristos.com/Evol%20Neural%20Networks/EvolNN3DivConquerSpa
4

[11] José Rodríguez (2021d)
Falsificación de la Teoría Evolutiva con Flipp.
https://ejristos.com/GPfalsifiesEvo/FalsifTEconFlipp.pdf
7

[12] José Rodríguez (2022e)
Falsificación de la Teoría Evolutiva con MicroLisp.
https://ejristos.com/GPfalsifiesEvo/FalsifTEconMicroLisp.pdf
7

Estos enlaces estaban activos el 15/05/2025