REDES NEURONALES EVOLUCIONARIAS
2. SIMPLIFICACIÓN

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

May 30, 2024

Abstract

Investigamos las posibilidades de la Evolución para simplificar una red neuronal hecha por mano humana y posiblemente redundante.

Contents

1 REVISIÓN **4**

2 LA EXPANSIÓN OR **8**

2.1 En retrospectiva 14

3 LA EVOLUCIÓN VIENE EN NUESTRA AYUDA **14**

3.1 Un Algoritmo genético para la optimización de la architectura de una RN 15

3.2 UN EXPERIMENTO 19

4 CONCLUSIÓN **29**

5 RESPUESTAS A LOS EJERCICIOS **30**

6 BIBLIOGRAFÍA **48**

INTRODUCCIÓN

Nuestro objetivo global es investigar las posibilidades de la Evolución para diseñar Redes Neuronales (RNs) y extraer conclusiones que puedan ser relevantes para la evaluación de la Teoría Evolutiva (TE).

En la sección 1, REVISIÓN, repasamos los conceptos importantes relacionados con las redes neuronales (RNs) que se introdujeron en nuestra Introducción (Rodríguez, [1] 2023). Lo que llamamos perfección allí era lo mejor que podía hacer una RN específica con dos perceptrones. Podía clasificar sin desajustes la mitad de los mapas de bits entre un total de 512. Entonces, nuestro perfeccionismo no era más que mediocridad. Mejorar el rendimiento es el primer desafío que se aborda en este artículo.

En la Sección 2, LA EXPANSIÓN OR, ampliamos las capacidades de la RN inicial agregando más perceptrones para abrir más canales de reconocimiento, los que se unen por un perceptrón OR. Siguiendo esta directiva, obtenemos la perfección, una RN que puede reconocer los 512 mapas de bits sin un solo error. Encontramos que la perfección tiene para nosotros un costo exponencial en el número de bits de los mapas de bits en estudio. Esto significa que nuestro enfoque es insostenible para el mundo real.

Se pide ayuda a la evolución en la sección 3. LA EVOLUCIÓN VIENE A AYUDARNOS. De hecho, puede ayudar, pero no lo suficiente como para aplicarlo al mundo real. La razón es que consideramos que las RNs simplificadas también tienen un número de perceptrones que es exponencial en el número de bits de los mapas de bits.

A menos que se indique lo contrario, los programas se ejecutaron en una computadora de mesa con 8 Gigabytes de RAM.

1 REVISIÓN

Nuestro objetivo general es el reconocimiento de patrones con la ayuda de la Evolución.

En nuestra introducción trabajamos con un modelo de juguete en el que las imágenes consistían en píxeles blancos o negros dispuestos en una cuadrícula de 3×3. Definimos dos arquetipos, uno representaba el número 1 y el otro el 0:

```
010      111
010      101
010      111
```

Los mapas de bits correspondientes fueron 010010010 y 111101111. Un mapa de bits es una cadena binaria que representa un signo.

La cuadrícula da lugar a 512 imágenes o mapas de bits diferentes. La distancia Bit a Bit (distancia BbB o BBB) clasificó cada mapa de bits como una deformación de 1 o 0 dependiendo de la distancia más corta a los arquetipos. En la mayoría de los casos, esta clasificación coincide con la del sentido común. Por ejemplo, una deformación de 1 va seguida de una deformación de 0:

```
010      111
000      100
010      111
```

Sea CLASS1 la familia de mapas de bits que se clasifican por la distancia BBB como una deformación de 1, y CLASS0 como las deformaciones de 0.

Para implementar el reconocimiento de patrones, abstrajimos la neurona como una compuerta booleana generalizada tolerante al ruido, que fue simulada por un perceptrón o célula.

1 ***Ejercicio.*** *Explique claramente por qué la lógica de un perceptrón es o no booleana. ¿A qué tipo de lógica obedece?* Respuesta

Unimos dos perceptrones en una Red Neural (RN) con la arquitectura presentada en la figura 1, y demostramos que puede clasificar correctamente 260 de los 512 posibles mapas de bits donde la distancia BBB definió la asignación correcta.

También mostramos en nuestra Introducción que la Evolución puede ajustar algunos parámetros aislados de los perceptrones de nuestra RN para maximizar las coincidencias pero sin un solo desajuste. Este resultado fue válido para sesgos y umbrales, pero fallamos con los vectores de peso. En el último caso, el rendimiento deambulaba preferentemente bajo la mediocridad, un hecho que tomamos como la firma de Evolución. En nuestra introducción, la proporción 260/512 era nuestra definición de perfección. Pero aquí no es más que mediocridad, y nuestro desafío es dejarla para abrazar la búsqueda de la perfección real. Para acometer esta empresa, recordemos la forma de diseñar un perceptrón para reconocer un mapa de bits determinado.

Para diseñar un perceptrón que reconozca un mapa de bits determinado al que llamamos arquetipo, seguimos la siguiente prescripción:

1. El perceptrón debe decidir si un estímulo es lo suficientemente fuerte como para dispararlo. La fuerza del estímulo viene dada por una suma total S, y debe ser más fuerte que un cierto umbral T. Entonces, debemos decidir si $S \geq T$.

2. Para reconocer el mapa de bits del arquetipo, lo transformamos en una expresión algebraica de sumas de la forma $we + b$ con peso w, sesgo b, y e es el valor del bit dado, 1 ó 0. Cuando el bit es $e = 1$, $w = 1$ y $b = 0$. Cuando el bit es $e = 0$, $w = -1$ y $b = 1$.

3. En la suma total $S = \sum w_i e_i + b_i$, el término $\sum b_i$ es el *sesgo inicial*. S debe entrar en comparación con el umbral T que es igual al número de bits.

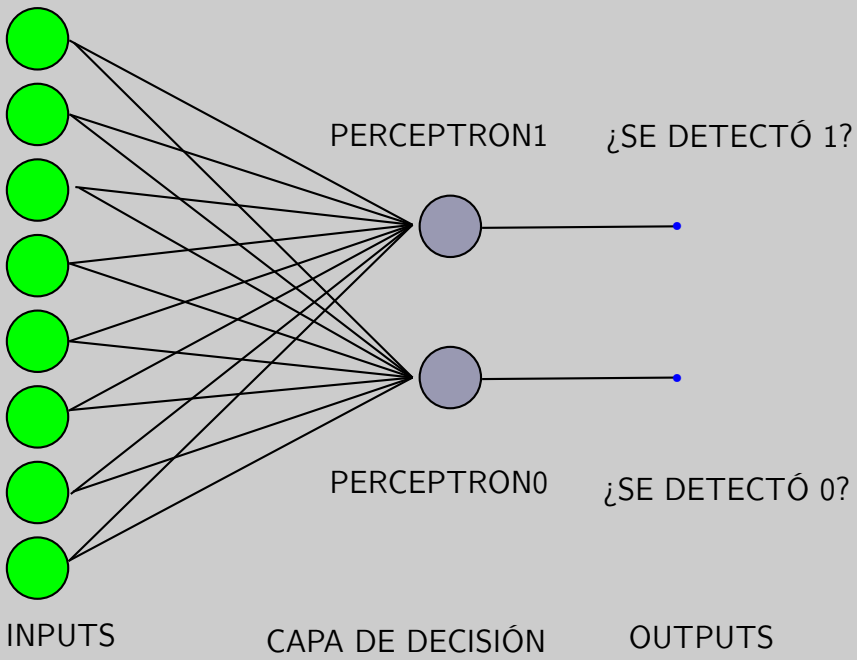


Figure 1: Esta RN fue diseñada por la inteligencia humana. Su tarea es reconocer imágenes distorsionadas de 1 o 0. ¿Cómo funciona? Tiene 9 entradas, una para cada píxel en la cuadrícula binaria 3×3. Las entradas representan un mapa de bits posiblemente distorsionado de las imágenes de 1 o 0. Todas las entradas están conectadas a cada celda o perceptrón de la capa de decisión, que contiene solo 2 celdas o perceptrones. La celda superior debe generar 1 cuando se presenta una imagen de 1 y 0 en caso contrario. La celda inferior debe generar 1 cuando se presenta 0, y 0 en caso contrario. Las imágenes pueden estar un poco distorsionadas por el ruido y/o la variabilidad. La salida general es una cadena, con la salida del Perceptron 1 seguida de la del Perceptron 0. Advertencia: esta salida debe analizarse para ver si contiene una contradicción. Esto sucede cuando ambos perceptrones emiten 1, eso significa que el primer perceptrón ha percibido un 1 y el otro un 0. Si la salida es 10, dice que el primer perceptrón percibió un 1, mientras que el otro no encontró nada. La RN percibió un 1. Si la salida es 01, la entrada era desconocida para el primer perceptrón, mientras que el segundo percibió un 0. La RN percibió un 0. Aquí no hay contradicción. Lo mismo ocurre con la salida 00, que se interpreta como DESCONOCIDA. Por tanto, la lógica de este RN tiene tres valores: 1, 0, DESCONOCIDO.

4. Para extender la tolerancia al ruido o la potencia de reconocimiento en presencia de ruido, se agregó 0.2 al sesgo para cada entrada o bit. ¿Por qué? El poder de reconocimiento significa que el perceptrón es capaz de reconocer imágenes ligeramente distorsionadas. Éstos evocarían respuestas más débiles que el espécimen prístino. Para garantizar que también disparen el perceptrón, se aumenta un poco su intensidad mediante la fuerza bruta. Por lo tanto, se suma un número positivo a_i para cada bit. Finalmente, obtenemos $S = \sum w_i e_i + b_i + a_i$. El término $\sum a_i$ es el sesgo adicional.

5. La salida del perceptrón se lee a partir de la desigualdad $S > T$ que dice que el perceptrón debe reconocer el arquetipo, es decir, la suma total debe ser lo suficientemente fuerte como para activarlo.

2 *Ejemplo.* *Reproduzcamos un resultado de nuestra Intro con este procedimiento. Esta receta se utilizará en todas partes todo el tiempo.*

Supongamos que nuestra tarea es reconocer un 1, cuyo mapa de bits es 010010010. Este es el arquetipo o, más formalmente, el mapa de bits del arquetipo. El perceptrón AND para reconocerlo se construye de la siguiente manera:

- El perceptrón debe decidir si un estímulo es lo suficientemente fuerte como para dispararlo. La fuerza del estímulo viene dada por una suma total S, y debe ser más fuerte que un cierto umbral T. Entonces, debemos decidir si $S \geq T$.

- La suma S es igual a $\sum w_i e_i + b_i$ donde w_i es el peso asociado al estímulo i que se debe multiplicar por e_i, el bit i del mapa de bits, mientras b_i es el sesgo inicial $i - simo$.

- El peso asociado a un 1 en la entrada es 1 y el sesgo es 0. El peso asociado a un 0 es -1 y el sesgo es 1. Por tanto, para el

arquetipo 010010010 obtenemos $S = -e_0 + 1 + e_1 - e_2 + 1 - e_3 + 1 + e_4 - e_5 + 1 - e_6 + 1 + e_7 - e_8 + 1 = -e_0 + e_1 - e_2 - e_3 + e_4 - e_5 - e_6 + e_7 - e_8 + 6$ donde contamos los bits desde 0 hasta 8. Como hay 6 ceros, el sesgo inicial es 6.

- El umbral es el número de bits en el arquetipo o la longitud de cada mapa de bits. Es 9.
- Para mejorar la tolerancia o el poder de reconocimiento, restemos 1.8 del umbral, 0.2 por cada bit, un valor que se eligió tentativamente. Esto equivale a agregar 1.8 al sesgo. Se llama **sesgo adicional**. Sesgo total $B = 6 + 1,8 = 7,8$.
- El ejecutor de la decisión decide si $S = -e_0 + e_1 - e_2 - e_3 + e_4 - e_5 - e_6 + e_7 - e_8 + 7.8 \geq T = 9$.
- Esta desigualdad debe cumplirse para el arquetipo 010010010. ¿Es eso cierto? Reemplazando las variables e_i por el bit correspondiente en el arquetipo, obtenemos: $S = 1 + 1 + 1 + 7.8 = 10.8 \geq T = 9$, lo cual es cierto. El perceptrón reconoce el arquetipo. Notemos que si eliminamos el sesgo adicional, 1.8, la desigualdad dice: $9 \geq 9$, eso significa que cuando no hay tolerancia, el perceptrón reconoce de todos modos el mapa de bits del arquetipo.

3 ***Ejercicio.*** *Demuestre que cuando no hay sesgo adicional, el perceptrón reconoce sólo el arquetipo. Por tanto, el sesgo adicional es el término responsable de la tolerancia a fallos.* *Respuesta*

4 ***Ejercicio.*** *¿Es posible implementar una lógica booleana con perceptrones?* *Respuesta*

2 LA EXPANSIÓN OR

En nuestra introducción, todos los mapas de bits que representan un signo en una cuadrícula 3×3 se clasificaron mediante un método

de distancia como deformaciones de 1 o 0, que eran los arquetipos. La distancia BBB asigna cada mapa de bits al arquetipo más cercano: la CLASS1 es la familia de mapas de bits que se clasifican como deformaciones de 1, mientras que CLASS0 consta de aquellos que cuentan como deformaciones de 0. Diseñamos una RN con dos perceptrones para reconocerlos. Después de la optimización, esta RN pudo clasificar correctamente aproximadamente la mitad de todos los mapas de bits posibles. Por lo tanto, *el desempeño de nuestra RN fue mediocre*.

¿Qué haremos para mejorar la calidad? Si un par de perceptrones fueron capaces de reconocer la mitad de los mapas de bits sin errores de clasificación, ¿qué pasaría si añadiéramos otro par? Por lo tanto, agregaremos otro par de perceptrones AND para reconocer un par de mapas de bits no reconocidos previamente. Así, cada arquetipo tendrá dos perceptrones a su servicio. Investigaremos cómo expandir al máximo su poder de reconocimiento sin errores y después de eso, se les podría unir por un perceptrón OR porque los dos perceptrones colaboran en el reconocimiento. Ver figura 2.

5 ***Ejercicio.*** *¿Planeamos unir los dos perceptrones asociados a cada uno de los arquetipos mediante un perceptrón OR? ¿No sería mejor utilizar una compuerta OR?* *Respuesta*

Debemos estar preparados para agregar muchos perceptrones nuevos para mejorar el poder de reconocimiento de nuestra RN. Dado que todos los perceptrones asociados a cada uno de los dos arquetipos se insertan en un perceptrón OR, denominamos a esta adición la **expansión OR**. Es por eso que necesitamos un método que reciba como entrada un mapa de bits específico y genere un perceptrón AND que lo reconozca. El perceptrón resultante se llama *el perceptrón generado por el mapa de bits dado*. Está incluido en el programa `compareTwoClassifications4.html`.

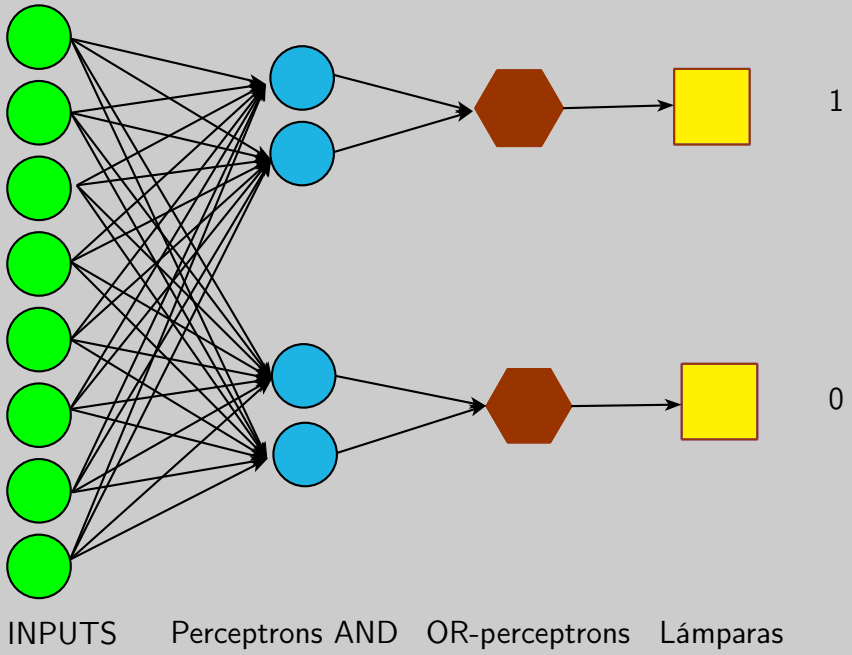


Figure 2: Un perceptrón tiene como objetivo reconocer un mapa de bits que desempeña el papel de arquetipo. Cuando se le proporciona un margen de tolerancia, puede reconocer otros mapas de bits como deformaciones del arquetipo dado. Una RN antigua con dos perceptrones pudo reconocer 260 mapas de bits de 512 y sin ningún error de clasificación. Para mejorar la capacidad de clasificación o la tolerancia a fallos, añadimos otro par de perceptrones. El primer perceptrón nuevo tenía como objetivo reconocer un mapa de bits en CLASS1, las deformaciones del arquetipo de 1. Las dos células estaban unidas por un perceptrón OR. La tarea correspondiente fue resuelta por un segundo perceptrón para el arquetipo de 0. Nos basamos en la distancia BBB para definir qué es distante. Cada arquetipo termina con dos perceptrones a su servicio, que pueden estar unidos por un perceptrón OR. El problema es encontrar la máxima capacidad de reconocimiento sin desajustes ni contradicciones.

6 ***Ejercicio.*** *Ejecute el programa* `compareTwoClassifications4.html`*, que contiene un RN con dos perceptrones que fueron creados automáticamente, y pruébelo por sus resultados: debe reproducir exactamente el mismo resultado del programa* `compareTwoClassifications3.htm` *de nuestra Introducción. Optimice el rendimiento eligiendo entre las opciones del elemento de selección. Originalmente, establecimos el* ***sesgo adicional*** *que mejora el poder de reconocimiento en* $0.2 \times n$*, donde* n *es el número de entradas de cada mapa de bits (todas con la misma longitud). ¿Tiene una propuesta mejor? ¿Espera que sea firme para la expansión OR?* Respuesta

Podemos comenzar nuestra expansión OR agregando dos nuevos perceptrones a la RN con los dos perceptrones que reconocen los 2 arquetipos. Los nuevos deben reconocer dos mapas de bits que el antiguo par de perceptrones no puede. El mapa de bits asociado al primer nuevo perceptrón debe clasificarse como 1 por la distancia BBB, por lo que debe estar en CLASS1, mientras que el del segundo debe clasificarse como 0 y debe estar en CLASS0. El problema es que hay más de doscientos mapas de bits que los dos perceptrones originales no pueden reconocer, mientras que debemos elegir sólo dos mapas de bits. ¿Cuáles? Elijamos dos mapas de bits de la clase de los más distantes. Al ejecutar el programa `compareTwoClassifications4.html` y observar el área de texto CLASIFICACIÓN TRIPLE, decidimos al principio elegir el mapa de bits 101110000 que fue clasificado por la distancia BBB como una deformación de 1. Para las deformaciones de 0, tomamos 101111000. La distancia de ambos mapas de bits a sus respectivos arquetipos originales fue 5.

Estas ideas se implementaron en el programa `addTwoPerceptrons.html`. En este programa, los sesgos de los perceptrones P1 y P0 son los que maximizan las coincidencias sin un solo desajuste. A su vez, los sesgos del nuevo par de perceptrones se pueden cambiar eligiendo una opción en el elemento de selección correspondiente. El perceptrón

que ayuda a P1 es P1h, mientras que P0h está destinado a ayudar a P0.

7 *Ejercicio.* *Ejecute el programa* `addTwoPerceptrons.html` *agregando los dos mapas de bits 101110000 y 101111000 de la clase 5, a una distancia 5 de los arquetipos. El programa genera automáticamente los dos perceptrones correspondientes que reconocen estos mapas de bits. ¿Son de alguna ayuda los dos perceptrones adicionales? Explique sus resultados.* Respuesta

8 *Ejercicio.* *Ejecute el programa* `addTwoPerceptrons.html` *adicionando dos mapas de bits de la clase 4, a una distancia 4 de los arquetipos. Para elegir un par, ayúdate de una lista de los elementos de esta clase que es producida por el programa* `compareTwoClassifications4.` *y publicada en el Área de Texto CLASIFICACIÓN TRIPLE. ¿Son de alguna ayuda los dos perceptrones adicionales?* Respuesta

9 *Ejercicio.* *Ejecute el programa* `addTwoPerceptrons.html` *agregando dos mapas de bits de destino de la clase 3, a una distancia 3 de los arquetipos. ¿Son de alguna ayuda los dos perceptrones adicionales?* Respuesta

Soñábamos con un gran éxito si añadíamos algunos perceptrones. Pero en nuestros intentos anteriores en los que se añadieron dos perceptrones, obtuvimos un máximo de 7 coincidencias adicionales. No hemos logrado producir una solución simple y directa al problema del reconocimiento de patrones. Por tanto, preparémonos para una larga guerra contra la complejidad. El primer paso es mejorar nuestro código JS.

10 *Ejemplo.* *Buscando elegancia y simplicidad, las clases fundamentales de JS, Perceptron y RN, fueron redefinidas en el programa* `addTwoPerceptrons2.html` *. La RN se dividió en dos semicapas que agrupan perceptrones acompañantes que atacan el mismo tipo de tarea, para detectar un 1 o un 0. Se incluyó un perceptrón*

OR por semicapa para unir las salidas de sus perceptrones. Por lo tanto, hemos implementado la arquitectura prometida de la figura 2. Consideramos que la escalabilidad está garantizada -con algunos pequeños ajustes. Esto se utilizará a continuación para implementar la evolución de la arquitectura de una NN determinada.

Como segundo paso para mejorar nuestra maquinaria, ampliemos nuestra NN para incluir no sólo dos perceptrones por semicapa, sino toda una familia o clase de perceptrones, que están asociados a una clase de mapas de bits. Por cada mapa de bits de la familia, se agrega un perceptrón destinado a reconocerlo. Su sesgo adicional puede ser 0 o 1. Esto también se implementó en el programa `addTwoPerceptrons2.html`.

11 *Ejercicio.* *El programa `addTwoPerceptrons2.html` expande nuestro NN agregando los perceptrones asociados con las dos 3-clases (agrega todos los mapas de bits a una distancia 3 del arquetipo correspondiente). Ejecútelo y publique los resultados.* *Respuesta*

12 *Ejercicio.* *¿Qué pasa si agregamos todos los maps de bits de la clase 4 solamente? Pista: para definir la clase, vaya a la línea 1282 del código del programa `add3Class.html`.* *Respuesta*

13 *Ejercicio.* *¿Qué pasa si agregamos la clase 5? Tenga en cuenta que el sesgo adicional, que responde a un reconocimiento mejorado, era un valor aleatorio entre 1 y 0.* *Respuesta*

14 *Ejercicio.* *Agregue algunos números a la siguiente objeción contra el triunfalismo: la adición de clases enteras ayuda, pero hay que pagar un alto costo porque estas clases son realmente enormes. Más concretamente, el objetivo de esta investigación no es mejorar las coincidencias sino mejorar las coincidencias sin desajustes. Por lo tanto, no hay motivo para alegrarse porque, si bien el número de coincidencias aumentó, el número de desajustes también fue enorme.* *Respuesta*

2.1 En retrospectiva

Hemos buscado una respuesta fácil y rápida al problema de conseguir la perfección. Hemos fracasado. Por tanto, es necesario saber si se puede alcanzar la perfección.

15 ***Ejercicio.*** *Utilice los conocimientos adquiridos hasta aquí para demostrar que la perfección es posible para nuestro modelo o problema de juguete. ¿Considera que tu metodología es aplicable a problemas reales?* *Respuesta*

Nuestra conclusión es simple: la perfección es alcanzable pero extremadamente costosa. Por lo tanto, nuestra tarea es simplificar la RN perfecta pero costosa. Para ello, le damos al sesgo adicional de un perceptrón arbitrario un valor de uno, siempre que no cause desajustes. De esa manera, un único perceptrón puede reconocer más de un mapa de bits y los perceptrones correspondientes pueden eliminarse. ¡Pidamos ayuda a la Evolución para implementar esa idea!

3 LA EVOLUCIÓN VIENE EN NUESTRA AYUDA

Ya tenemos una RN que puede reproducir perfectamente la clasificación de un grupo de mapas de bits dictada por una función de distancia. La RN tiene 254 perceptrones, todos los cuales tienen un sesgo adicional igual a 0 excepto aquellos dos que reconocen los arquetipos, que son privilegiados. Por eso, cada perceptrón sólo puede reconocer un mapa de bits, precisamente el que se utilizó para generarlo. Podemos darle al sesgo adicional el valor 1 y ver si se reconocen algunos otros mapas de bits pero sin discrepancias. Si ese es el caso, se pueden eliminar los perceptrones correspondientes. Este procedimiento de verificación es lo que la Evolución debe maximizar.

Observemos que, aparte del sesgo adicional, todo en la RN está congelado. Entonces, la Evolución hace dos tareas: elegir un subconjunto de perceptrones y jugar con su sesgo adicional, 1 o 0, para

ver si se mantiene la perfección mientras la RN se vuelve más simple. Así, la Evolución se refiere en primera instancia a la arquitectura de la RN, más exactamente, a su composición de perceptrones incluidos. ¿Por qué esperamos que esto funcione?

Esperamos que muestras más pequeñas de perceptrones funcionen mejor porque tienen menos oportunidades de crear contradicciones. Sin embargo, la condición límite de cero desajustes parece ser muy estricta. Esto debe aclararse mediante experimentación directa porque carecemos de una visión tan poderosa para predecir cuál será el resultado.

3.1 Un Algoritmo genético para la optimización de la architectura de una RN

Nuestra tarea es encontrar el NN más simple que discrimine perfectamente los mapas de bits como pertenecientes a cualquiera de dos clases, una deformación del arquetipo de 1 o 0. La clasificación de cada mapa de bits está dictada por la distancia BBB. Los NN constan de dos semicapas de perceptrones que reconocen los arquetipos antes mencionados, más un conjunto de perceptrones de ayuda que reconocen un mapa de bits específico de clase 4 o 5. Estas clases constan de aquellos mapas de bits cuya distancia al arquetipo más cercano es 4 o 5 y que naturalmente se dividen en dos subclases, una cuyos miembros cuentan como deformación de 1 y la otra como de 0. Por eso tenemos dos semicapas.

Se comprende fácilmente la codificación de la arquitectura y la posibilidad de su evolución. Imaginemos que tenemos una lista de mapas de bits. Una arquitectura se codifica en una primera prueba como una cadena binaria de presencia (1) o ausencia (0) de cada mapa de bits. Dicha cadena define una arquitectura porque cada mapa de bits genera un perceptrón que lo reconoce y la cadena determina qué perceptrones ingresan a la NN.

Ejemplo. Si una subclase de mapas de bits tiene 3 miembros,

la cadena 101 significa que el NN tiene dos perceptrones, uno para reconocer el primer mapa de bits y otro para reconocer el tercero. Ningún perceptrón está destinado directamente a reconocer el segundo mapa de bits, posiblemente porque es reconocido por uno de los otros dos.

Por lo tanto, la evolución de las arquitecturas de RNs se rastrea claramente por la evolución de la cadena binaria correspondiente que las codifica mediante la correspondencia de presencia-ausencia. Lo que veremos es un proceso evolutivo de cadenas binarias, y las RNs aparecen sólo en el proceso de evaluación de su aptitud. Sin embargo, la simplificación es posible gracias a un sesgo adicional distinto de cero, gracias al cual un perceptrón puede reconocer más que su mapa de bits generador. Pero esto todavía no está codificado en las cadenas binarias.

16 ***Ejercicio.*** *Proponer e implementar una solución para el siguiente problema: para codificar presencia-ausencia, necesitamos una cadena binaria. Pero, además, debemos gestionar el sesgo adicional que toman los valores 0 o 1. ¿Qué hacer?* *Respuesta*

La evolución se simula mediante un algoritmo genético. La población de cromosomas en evolución consta de cadenas terciarias que codifican la arquitectura de las RNs junto con el sesgo adicional de cada perceptrón, excepto aquellos que reconocen los arquetipos, que son privilegiados.

El algoritmo consta de los siguientes procedimientos:

1. Inicialización: Nuestro algoritmo comienza con una RN que reproduce perfectamente la distancia BBB sobre todos los mapas de bits. Por eso, todos los cromosomas de la población al principio son cadenas de unos, 111...1, es decir, excepto aquellos perceptrones que reconocen los arquetipos, todos los perceptrones están presentes, y sus sesgos son iguales a 0 y, por tanto, cada perceptrón puede reconocer como máximo su mapa de

bits generador. La RN posiblemente sea redundante. A continuación, la población está sujeta a un ciclo evolutivo recurrente que se espera que la simplifique manteniendo la perfección. El ciclo consiste en

2. Evaluación de la puntuación o aptitud de cada cromosoma. Se define de la siguiente manera: Cada cromosoma, una cadena terciaria, define la arquitectura de una RN mediante el código: [0 = ausencia; 1 = presente con sesgo adicional 0; 2= presente con sesgo adicional 1]. Los mapas de bits son los de las clases 4+5. Se mide el desempeño de la RN y se le otorga una puntuación dictada por el número de aciertos menos un castigo por sus errores de clasificación. El número de ceros en la cadena terciaria se suma a la puntuación porque representa la simplicidad lograda que resulta de eliminar algunos perceptrones. Cada RN se prueba en todos los mapas de bits y se otorga la puntuación correspondiente, la cual determina directamente su idoneidad.

   La función de aptitud es

```
score =
-50 * (512 - numberOfMatches) [chastise for  imperfect
- 5 * numberOfMismatches  [chastise for mistakes]
+ 0.1 * numberOfDeletions [reward for simplification]
+ 30000 [constant to make the result positive]}.
```

   donde el número de eliminaciones es el número de perceptrones que se eliminaron de la RN original. Es exactamente el número de ceros del cromosoma o cadena terciaria dada. La puntuación aumenta cuando aumenta el número de coincidencias y eliminaciones, y disminuye cuando aumenta el número de discrepancias.

3. Se clona al más apto para generar una nueva población con el mismo número de individuos.

4. La población pasa por mutación: se realizan algunos cambios en los clones. La mutación comienza con 111...1 y explora paso a paso lo que sucede cuando se introducen ceros y doces.

5. La población sufre recombinación o intercambio de información.

6. El ciclo termina después de un número determinado de generaciones.

Este algoritmo fue implementado en el programa `ENN2Simplify.html`. El programa selecciona las clases 4+5 para ampliar la RN y mejorar su rendimiento. El programa comenzó desde la perfección, es decir, con una RN que reconoce y clasifica correctamente los 512 mapas de bits.

17 ***Ejercicio.*** *Asegúrese de que el programa `ENN2Simplify.html` sea correcto, es decir, que efectivamente simule la Evolución. Para ello, active las banderas de impresión del programa para que imprima el efecto de mutación, recombinación y selección.* *Respuesta*

Los principales resultados exploratorios del programa `ENN2Simplify.html` son:

1. La evolución no puede mantener la perfección porque el número de aciertos disminuye hasta alcanzar una meseta turbulenta que representa un punto de equilibrio + fluctuaciones.

2. El número de coincidencias tiende a estabilizarse alrededor del 70-80% del valor óptimo inicial. Entonces, parece existir un punto de equilibrio.

3. El número de discrepancias crece de 0 a más de 100.

4. El número de eliminaciones aumenta de 0 a casi 100.

5. El número de coincidencias parece ajustarse a una función decreciente exponencial. El número de desajustes y de eliminaciones

parece ajustarse a funciones crecientes exponenciales. Todas las funciones están sujetas a fluctuaciones.

El programa se puede seguir a través de la consola que debe activarse antes de iniciar el programa.

Es necesario poner a prueba estas observaciones. Entonces, hagamos un experimento para verificar o rechazar la existencia de un punto de equilibrio en el número de coincidencias y evidenciar el papel de la mutación y la selección. La existencia de un valor de equilibrio del número de coincidencias es sorprendente porque el comportamiento de los perceptrones es caótico: su respuesta a un estímulo justo por debajo del umbral es totalmente diferente a la de un estímulo justo por encima de él.

3.2 UN EXPERIMENTO

Para arrojar luz sobre la validez de los resultados exploratorios que se acaban de formular, llevamos a cabo un experimento adecuado, cuyas características son:

El número de coincidencias se registró para un tamaño de población de 10, durante 3000 generaciones para diversos valores de la tasa de mutación y cuando la selección se activaba y desactivaba. Para desactivar la selección, todas las arquitecturas recibieron una puntuación igual a cero. En este caso, el programa crea variabilidad con mutación, recombinación y fuertes efectos de muestreo. Cuando se activa la selección, las puntuaciones de las arquitecturas son por méritos, como se expresó anteriormente. Para saber si existe un valor de equilibrio, se corrió el programa a partir de dos condiciones iniciales: desde perfección total cuando el número de coincidencias es máximo, 512, y desde imperfección total cuando el número de coincidencias es mínimo, 0. En el primer caso, todos los perceptrones de la RN están activos y tienen un sesgo adicional igual a 0. El número de coincidencias en este caso debe ser 512. En el segundo, ningún perceptrón está activo y por lo tanto no se reconoce ningún mapa

de bits, el número de coincidencias debe ser 0. Si existe un valor de equilibrio, ambas dinámicas deben converger al valor de equilibrio en algún punto intermedio. El programa genera el valor máximo de coincidencias para cada generación.

Los resultados siguen. Ver página siguiente.

EXPERIMENTOS SIN SELECTION

1. Mutation rate = 0.001.

```
From below
Generation numberOfMatches
 0       0
300     101
600     248
900     259
1200 260
1500 342
1800 332
2100 330
2400 331
2700 334
3000 338


From above
0        512
300     434
600     388
900      377
1200 363
1500 270
1800 281
2100 347
2400 340
2700 269
3000 278
```

2. Mutation rate = 0.01.

```
From below
Generation numberOfMatches
 0      0
300     273
600     269
900     340
1200 201
1500 335
1800 266
2100 269
2400 208
2700 205
3000 262


From above
 0      512 intel
300     275
600     271
900     275
1200 276
1500 350
1800 259
2100 343
2400 338
2700 205
3000 270
```

3. Mutation rate = 0.1.

```
 From below
 Generation numberOfMatches
 0       0
300      262
600      337
900      329
1200 342
1500 269
1800 342
2100 335
2400 266
2700 331
3000 337

From above
 0       512
300      274
600      330
900      331
1200 260
1500 271
1800 273
2100 265
2400 209
2700 336
3000 270
```

4. Mutation rate = 0.5

```
 From below
 Generation numberOfMatches
 0      0
300     278
600     267
900     338
1200 271
1500 338
1800 336
2100 335
2400 197
2700 336
3000 274


From above
0       512
300     262
600     335
900     345
1200 334
1500 337
1800 278
2100 334
2400 271
2700 267
3000 335
```

EXPERIMENTOS CON SELECTION

5. Mutation rate = 0.001.

```
From below
 Generation numberOfMatches
 0      0
300     358
600     368
900     369
1200 378
1500 386
1800 394
2100 401
2400 405
2700 407
3000 403

From above
0       512
300 491
600 479
900 473
1200 468
1500 456
1800 459
2100 450
2400 443
2700 432
3000 434
```

6. Mutation rate = 0.01.

```
 From below
 Generation numberOfMatches
 0      0
300     392
600     408
900     399
1200 412
1500 401
1800 398
2100 392
2400 406
2700 397
3000 401

From above
0      512
300     401
600     403
900     401
1200 385
1500 400
1800 404
2100 404
2400 393
2700 396
3000 409
```

7. Mutation rate = 0.1.

```
From below
Generation numberOfMatches
 0      0
300     357
600     348
900     342
1200 357
1500 349
1800 346
2100 357
2400 351
2700 350
3000 360

From above
0      512
300     353
600     354
900     351
1200 357
1500 346
1800 352
2100 350
2400 358
2700 361
3000 360
```

8. Mutation rate = 0.5.

```
 From below
 Generation numberOfMatches
 0       0
300      345
600      343
900      331
1200 333
1500 339
1800 332
2100 344
2400 339
2700 326
3000 346

 From above
0       512
300      342
600      336
900      334
1200 338
1500 345
1800 349
2100 341
2400 339
2700 336
3000 337
```

18 ***Ejercicio.*** *Utilice una hoja de cálculo para dibujar los datos presentados y sacar sus conclusiones con respecto a la posible existencia de un valor de equilibrio y el papel de la selección y de la tasa de mutación sobre él. Dé sentido a los resultados.* Respuesta

19 ***Ejercicio.*** *Implementar una salida a la siguiente acusación: si la Evolución no puede mantener la perfección trabajando en simplificación, es porque está mal programada. Por lo tanto, se debe cambiar la definición de la función de puntuación.* Respuesta

Nuestro análisis hasta este punto contiene posiblemente una contradicción. Por un lado, decimos que la tasa de mutación está desconectada del nivel del punto de equilibrio. Por otro lado, observamos que la velocidad de aproximación al valor de equilibrio es menor cuanto menor es la tasa de mutación. ¿Qué podría pasar cuando la tasa de mutación es muy baja?

20 ***Ejercicio.*** *Utilizando la puntuación recién definida en la respuesta al ejercicio anterior, pruebe la idea de que una tasa de mutación más baja producirá un resultado menos imperfecto. Empiece con* $0,0005$*. ¿Estaría de acuerdo en que para tasas de mutación muy bajas, la perfección es la regla?* Respuesta

Los resultados del último ejercicio muestran que **La evolución puede crear y mantener la perfección para el problema de simplificar una RN** pues se eliminaron 70 perceptrones de 254.

21 ***Ejercicio.*** *¿Considera que las buenas noticias sobre el poder simplificador de la Evolución sobre las RNs se aplican al mundo real?* Respuesta

22 ***Ejercicio.*** *Haga todo lo posible para organizar todos los resultados según la Vista Científica Oficial sobre la Evolución.* Respuesta

23 ***Ejercicio.*** *¿Considera que nuestros resultados refutan o apoyan la opinión científica oficial sobre la evolución?* Respuesta

4 CONCLUSIÓN

Construimos con nuestra inteligencia humana una RN con 254 perceptrones que pueden reconocer correctamente los 512 mapas de bits

posibles de una familia determinada que fueron clasificados por una distancia como deformación de una imagen del número 1 o de 0. Le dimos a la Evolución la tarea de simplificar la RN manteniendo la perfección, es decir, sin desajustes ni incógnitas. La evolución puede simplificar fácilmente la RN si la imperfección es bienvenida: logramos una simplificación de 70 perceptrones junto con muchos desajustes e incógnitas. Pero para lograr la perfección, 512 coincidencias, sin que se produjeran discrepancias ni incógnitas, necesitábamos reducir la tasa de mutación. Un valor de 0.0005 por carácter por generación fue suficiente para nosotros. Este es un resultado pobre para aquellas personas que quieren sostener la TE porque apenas resuelve un detalle de un problema extremadamente complejo. En cambio, se ajusta a las necesidades de aquellos que quieren falsificar la TE porque vemos que la Evolución es una herramienta ordinaria que quema tiempo y recursos y por tanto toneladas de imperfecciones es lo que debe quedar detrás de la perfección cuando se alcanza. Sin embargo, esto no es una falsificación porque carece de predicciones respecto a la suerte de los fósiles. Por eso hemos recurrido a la abstracción, como se mostró en trabajos anteriores.

5 RESPUESTAS A LOS EJERCICIOS

1, página 4. Un perceptrón se activa cuando el estímulo supera un umbral determinado. El disparo se interpreta aquí como una afirmación de que se ha encontrado un patrón. Pero si un estímulo no provoca el disparo, el perceptrón permanece como si no hubiera ningún estímulo. Por tanto, el perceptrón lo ignora por dos posibles causas. Primera: el estímulo no representa el patrón objetivo. Segundo: la insuficiente capacidad discriminante del perceptrón.

Consideramos que cuando el perceptrón se dispara representa un SÍ. Pero si no dispara, está diciendo que el estímulo es DESCONOCIDO. Por lo tanto, el comportamiento general tiene dos valores, SÍ y DESCONOCIDO, pero es no booleano porque de su silencio no

podemos concluir que el estímulo no corresponde al patrón objetivo.

3, página 8. Supongamos que un perceptrón está diseñado para reconocer el arquetipo de 1 cuyo mapa de bits es 010010010. La desigualdad correspondiente es $S = -e_0 + e_1 - e_2 - e_3 + e_4 - e_5 - e_6 + e_7 - e_8 + 6 \geq T = 9$. Si ingresamos el arquetipo, obtenemos $S = 0+1+0+0+1+0+0+1+0+7.8 = 10.8 \geq T = 9$ que dice que se reconoce el arquetipo. Otro mapa de bits tendría al menos una mutación. Consideremos, por ejemplo, 110010010, que representa $S = -1+1-0-0+1-0-0+1-0+6 = 8 < 9$, lo que significa que este mapa de bits no se reconoce como 1. Consideremos ahora 000010010, que representa: $S = -0+0-0-0+1-0-0+1-0+6 = 8 < 9$ por lo que no se reconoce como una deformación de 1. Y y así sucesivamente: una mutación en el arquetipo de 0 en 1, cambia un 0 por un -1 en la suma, mientras que un cambio de un 1 en 0, cambia un 1 por un 0 en la suma. Por con siguiente, la nueva suma es menor que la del arquetipo. Concluimos que sin sesgos adicionales, el perceptrón reconoce sólo el arquetipo. Por tanto, el sesgo adicional es el término responsable de la tolerancia del perceptrón.

4, página 8. Supongamos que tenemos que discriminar entre dos patrones. Ideamos una NR con dos lámparas terminales, la primera brilla cuando se detecta el primer patrón, la segunda cuando se manifiesta el segundo patrón. En una lógica booleana, si el primer patrón no se detecta, el segundo sí debe detectarse. Y viceversa. Como veremos, tal certeza es tremendamente esquiva porque es fácil cometer errores de clasificación, en los que ambas lámparas terminales se encienden, es decir, ambas afirman haber encontrado su patrón correspondiente. Desde la perspectiva booleana, la habitual para un sector importante de nuestras vidas, esto es una contradicción. O cuando se enciende la lámpara equivocada mientras la correcta permanece apagada. Esto es un desajuste y también una contradicción con la clasificación correcta. Por lo tanto, uno debe contentarse con

una lógica booleana aproximada pero tratando de disminuir la incertidumbre y las contradicciones tanto como sea posible. Esto implica que estamos ante un problema de optimización, un tema que le da la bienvenida a los Algoritmos Genéticos, que son simulaciones de la Evolución, que hemos explorado y seguiremos haciéndolo.

5, página 9. Sí, sería más sencillo unir los perceptrones mediante una compuerta OR. Pero el objetivo importante para nosotros es facilitar la próxima expansión recursiva, y esto será más sencillo con perceptrones en lugar de compuertas. Planeamos agregar un par de perceptrones para ampliar las capacidades de reconocimiento de la RN, pero no estamos seguros de que esto sea suficiente para alcanzar la perfección. Entonces, debemos estar listos para agregar más pares, y es de esa manera que obtenemos una expansión OR. Un perceptrón OR será el indicado para hacer frente a esta situación.

6, página 9. Se encontró que los valores óptimos de los sesgos eran $9{,}8$ para el perceptrón P1 y $4{,}8$ para P0. El número de coincidencias fue 260 y las discrepancias 0 de 512 posibles mapas de bits. Para ajustar estos valores, establecemos el sesgo adicional en $0{,}4 \times n$, donde n es el número de entradas de cada mapa de bits. Para hacer esto, utilizamos la definición de sesgo total: es el número de ceros en el mapa de bits, o de -1's en el vector de pesos, más el sesgo adicional para mejorar el poder de reconocimiento. Ahora bien, ¿podemos utilizar esta fórmula indiscriminadamente?

No. Esta fórmula, que siempre puede ser cuestionada, podría ser aceptable para perceptrones aislados. Pero en la expansión OR tenemos dos módulos, cada uno con un perceptrón principal, dirigido a uno de los arquetipos, y unos auxiliares que deben reconocer mapas de bits adicionales pero sin contradecir la distancia BBB. Ése es un problema totalmente nuevo. Por lo tanto, se debe investigar el valor óptimo. Debemos permitir que el perceptrón acepte aquellos mapas de bits que difieren en un lugar de su objetivo. Por lo tanto, el sesgo

adicional podría establecerse uniformemente en 1. Por la misma razón, el sesgo podría crecer una unidad por paso cuando se busca su optimización cuando está en juego la cuestión de agregar o no un nuevo perceptrón.

7, página 12. El mejor rendimiento para los perceptrones adicionales P1h y P0h con mapas de bits 101110000 y 101111000 respectivamente fue de 262 coincidencias con 0 discrepancias. Esto se logró seleccionando el sesgo de P1h como 5 y el de P0h como 4. Esto significa una ganancia de 2 coincidencias entre más de 250 que faltan. El resultado es insignificante. ¿Por qué es tan pequeño?

La distancia entre estos dos mapas de bits, 101110000 y 101111000, es 1, pero según la distancia BBB, el primero representa un 1, mientras que el segundo un 0. Esto significa que los respectivos perceptrones AND que reconocen a cualquiera de ellos no pueden tener margen de tolerancia cuando se exige 0 errores. Por lo tanto, los nuevos perceptrones reconocen los nuevos mapas de bits de destino y nada más. ¿Es eso cierto? Esta es una prueba para toda nuestra programación. ¿Es eso cierto?

Sí. Es verdad. Al observar los resultados que aparecen en el área de texto CLASIFICACIÓN QUÍNTUPLE, se puede ver que, con una excepción, todas las coincidencias pertenecen a las clases 0, 1, 2 y 3, que ya fueron aceptadas por el NN original con solo dos perceptrones. La excepción correspondió a 101110000, el objetivo auxiliar 1. Lo mismo ocurre con el segundo objetivo auxiliar 101111000: el sesgo total óptimo es el sesgo inicial. Pero ¿por qué sólo ellos? Para saber eso, notamos que el sesgo total óptimo del primer perceptrón era 5, que es justo el sesgo correspondiente a 101110000, que tiene 5 ceros, sin margen para un reconocimiento mejorado. ¿Qué pasa si aumentamos el sesgo? Al observar los resultados con sesgos totales 6 y 4, encontramos en el área de texto antes mencionada que el NN clasifica correctamente 6 nuevos mapas de bits que se clasifican por la distancia BBB como deformaciones de 1. No hay discrepancias

para este caso. Sin embargo, el perceptrón P1 clasificó 4 mapas de bits que son deformaciones de cero como 1 y el perceptrón P0 como 0. Vemos que el mejor desempeño no tiene tolerancia.

8, página 12. Con los mapas de bits de destino 000000001 y 000001111 de la clase 4, se obtuvieron 262 coincidencias con 0 errores seleccionando el sesgo de P1h como 8 y el de P0h como 5. Si los sesgos son 9 y 5, el número de coincidencias es 267 sin discrepancias. Esto no es un error anunciado porque la distancia entre estos dos mapas de bits es 3. El código se modificó alrededor de la línea 280.

9, página 12. Con los mapas de bits objetivo 000000000 y 000101111 de clase 3, solo se obtuvieron 260 coincidencias con 0 errores seleccionando el sesgo de P1h como 9 y el de P0h como 4. Pero si los sesgos son 9 y 5, el número de coincidencias limpias es 266. Esto no es un fallo anunciado porque la distancia entre estos dos mapas de bits es 5.

11, página 13. Elegimos la clase 3 y la expansión correspondiente se consigna en el programa `add3Class.html`. Del programa `addTwoPerceptrons2.html` aprendimos que el sesgo adicional podría ser 0 o 1. Así, asignamos 0 o 1, elegido al azar, como sesgo adicional de cada perceptrón de la red. Después de algunas pruebas, el número de coincidencias fue muy cercano a 360, el de discordancias 67 y el de incógnitas 85 para un total de 512, que es el número de mapas de bits posibles. Por lo tanto, nos olvidaremos de toda la clase 3.

12, página 13. Si añadimos la clase 4, el número de coincidencias fue muy cercano a 376, el de desajustes 133 y las incógnitas 3 para un total de 512. Las incógnitas fueron sacrificadas a favor de coincidencias y desajustes, pero el gran favorecido fue el último.

13, página 13. El programa `add3class.html` fue modificado en la línea 1282 para seleccionar la clase 5, compuesta por aquellos mapas de bits que se encuentran a una distancia de 5 de los arquetipos correspondientes. Esta es la salida:

```
 The chosen class is the = 5-class.
 Number of bitmaps in \textsc{class1} = 35
 Number of bitmaps in \textsc{class0} = 35

runNN(myNNClass)
 numberOfMatches 380, 74.21875%
 numberOfMismatches 68, 13.28125%
 numberOfUnknowns 64, 12.5%
 Total 512
 Score = 40
```

Declaramos que una asignación aleatoria de 1 al adicionalBias no funciona porque puede aumentar las coincidencias pero a un alto precio en desajustes.

14, página 13. El número de mapas de bits en la clase 3 es $84+84=168$ mientras que las coincidencias obtenidas fueron casi cien. El número de mapas de bits de la clase 4 es $91+91=182$ mientras que las coincidencias obtenidas fueron un poco más de cien. Si sumamos la quinta clase, obtenemos unas 120 coincidencias, pero estas clases tienen un tamaño de $35+35=70$. Entonces la objeción es correcta.

15, página 14. El programa `perfection.html` muestra que es posible hacer 512 dianas sin desajustes. Esto es lo que hicimos:

Construimos una RN con dos semicapas. La primera estuvo dirigida a reconocer el arquetipo de 1 y sus deformaciones, mientras que el segundo los de 0 y sus deformaciones. Cada semicapa constaba de un perceptrón para reconocer el arquetipo correspondiente

y sus deformaciones a las distancias 1, 2 y 3. Para cada mapa de bits a la distancia 4 o 5 de cada uno de los dos arquetipos, se añadió un perceptrón auxiliar para reconocerlo. Cada mapa de bits tenía un perceptrón consagrado con un sesgo adicional igual a cero, lo que implicaba que la relación de reconocimiento era uno a uno en las clases 4 y 5. El número total de perceptrones fue 254, que es aproximadamente la mitad de los 512 mapas de bits que se reclasificarán. Por lo tanto, no es posible utilizar este método para lograr la perfección en aplicaciones reales porque el número de perceptrones necesarios crece exponencialmente con el número de bits de los mapas de bits.

16, página 16. Resolvimos el problema con el sesgo adicional de la siguiente manera: no usamos una cadena binaria para indicar la presencia-ausencia de un mapa de bits. En su lugar, utilizamos una cadena terciaria cuyas entradas pueden tomar los valores 0, 1 o 2. La opción 0 se interpreta como la ausencia del mapa de bits y del perceptrón correspondiente. El número 1 significa que el mapa de bits correspondiente está presente y que el sesgo adicional del perceptrón asociado es 0. El número 2 significa que el mapa de bits correspondiente está presente y que el sesgo adicional del perceptrón asociado es 1. Ejemplo: si tenemos un cromosoma de la forma 021, significa que tenemos una familia de mapas de bits con tres miembros. De ellos, el primero no se tomó, mientras que el segundo se tomó y se le otorgó un sesgo adicional de 1 para permitir el reconocimiento de más que el mapa de bits generador. A su vez, se tomó al tercer miembro y se le dio un sesgo adicional de 0.

17, página 18. En el programa `ENN2Simplify.html`, el número de generaciones a ejecutar se puso igual a 5, línea 351, y el tamaño de la población, línea 357, igual a 10. Las banderas de impresión están debajo de la línea 369. La mutación se hizo visible con el cambio `$PRINT_MUTATION_IND$=true`. No se le vieron fallos. La impresión del efecto de recombinación se ac-

tivó con `$PRINT_RECIND$=true`. Fue muy fácil comprobar que su trabajo es correcto. Los métodos que ejecutan la selección son diversos y complejos. Los métodos principales fueron `función evaluaFitnessPop(myNN, CLASS1, CLASS0)` en la línea 1719 y `función evaluaFitnessChro(myNN, cromosoma, CLASS1, CLASS0 )` en la línea 1662. No se realizó ninguna corrección adicional. En conclusión, el autor no detectó ningún error.

18, página 28. Con la ayuda de gráficos pero sin análisis estadístico, interpretamos estos resultados de la siguiente manera:

- La Evolución es lo suficientemente potente como para crear funciones de hasta 0,8 de perfección en menos de 3000 generaciones con una población de 10 individuos.
- El número de coincidencias tiene un valor de equilibrio alrededor del cual fluctúa el proceso.
- El valor de equilibrio es indeterminado pero está acotado por arriba. De hecho, el programa informa el número máximo de coincidencias de cada generación. Su número oscila entre 250 y 450. Por lo tanto, el valor medio está limitado por 450. Esto demuestra que La Evolución es incapaz de alcanzar o mantener la perfección, que está representada por 512 coincidencias.
- Aparte de raras excepciones, las eliminaciones van acompañadas de un gran número de incógnitas y discrepancias (no reportadas en las tablas anteriores).
- El valor promedio de coincidencias sin selección es casi 300, pero cuando se activa la selección, es 400. Vemos que el valor de equilibrio es muy sensible a la selección.
- El valor de la tasa de mutación no afecta el valor de equilibrio pero afecta la velocidad de aproximación a él, es decir, la velocidad de aproximación es menor para tasas de mutación bajas.

- Los caminos que conducen a la imperfección son muy comunes.

Racionalizamos nuestras inferencias de la siguiente manera:

1. El comportamiento de los perceptrones es caótico: la respuesta a un estímulo justo por debajo del umbral es bastante distinta de la de otro un poco superior al umbral. Nuestras observaciones dicen que el caos es visible a través de las fluctuaciones resultantes, que danzan alrededor de un valor de equilibrio.

2. La mutación y la recombinación crean y destruyen continuamente coincidencias. La cantidad de creaciones debidas a mutación es proporcional al número de no coincidencias y la de destrucción al de coincidencias. Tenemos un proceso de creación-aniquilación. El equilibrio se alcanza cuando la creación equilibra la aniquilación. El valor de equilibrio se ve desdibujado por las fluctuaciones debidas a la naturaleza caótica del comportamiento de los perceptrones y a los efectos de muestreo que deben disminuir cuando crece el tamaño de la población.

3. El valor de equilibrio es estable. No es como una aguja que gira sobre su punta, sino como una bola que fluctúa alrededor del fondo de un tazón. Esto significa que el valor de equilibrio absorbe tanto las condiciones iniciales como las fluctuaciones que pueden aparecer por un efecto combinado de selección y mutaciones afortunadas. La estabilidad del valor de equilibrio implica que la evolución queda bloqueada en la imperfección.

19, página 29. Hemos venido trabajando por el puntaje definido por:

```
score =
-50 * (512 - numberOfMatches) [chastise for  imperfection
- 5 * numberOfMismatches  [chastise for mistakes]
```

```
+ 0.1 * numberOfDeletions [reward for simplification]
+ 30000 [constant to make the result positive]}.
```

Pongamos a prueba ahora la siguiente definición:

```
 score = number of matches
         - 5 * number of mismatches
         + 0.1*number of deletions
```

El programa se ejecutó durante 10000 generaciones comenzando con la perfección total y con una tasa de mutación igual a 0.001 y un tamaño de población de 10.

```
Generation  Number of matches
0       512
1000 511
2000 509
3000 510
4000 510
5000 512
6000 512
7000 511
8000 508
9000 510
10000 511
mismaches were always zero,
unknowns were less than 4,
deletions nearly 60.
```

A partir de estos resultados, concluimos que la Evolución de esta RN es muy sensible a la forma de selección porque no se observó una estabilidad tan alta de cuasi perfección con la forma anterior de puntuación. Además, la Evolución no puede mantener la perfección para el valor de mutación de 0.001, pero la casi perfección es la regla.

20, página 29. Pusimos a prueba una tasa mutacional igual a 0.0005 sobre una población de 10 individuos. Los resultados son:

1. Experiments without selection

```
From below
MUT_RATE = 0.0005
Generation matches
0    0
1000 158
2000 181
3000 199
4000 273
5000 267
6000 268
7000 262
8000 266
9000 263
10000 329
11000 330
12000 339
13000 334
14000 342
15000 197
16000 200
17000 258
18000 277
19000 282
20000 282
```

From above
without Selection
MUT_RATE = 0.0005
Generation matches
0 512
1000 401
2000 287
3000 355
4000 345
5000 343
6000 335
7000 334
8000 201
9000 204
10000 194
11000 271
12000 268
13000 263
14000 333
15000 339
16000 341
17000 337
18000 344
19000 343
20000 266

2. Experiments with selection

```
From below
MUT_RATE = 0.0005
Generation matches
0    0
1000 351
2000 400
3000 503
4000 511
5000 511
6000 512
7000 512
8000 512
9000 512
10000 512
11000 512
12000 512
13000 512
14000 512
15000 512
16000 512
17000 512
18000 512
19000 512
20000 512
Number of deletions 70
Generation = 20000
numberOfDeletions = 70
numbOfMatches = 512
numbOfUnknowns = 0
numbOfMismatches = 0
score = 519
```

```
From above
MUT_RATE = 0.0005
Generation matches
0       512
1000 512
2000 512
3000 512
4000 512
5000 512
6000 509
7000 512
8000 512
9000 512
10000 512
11000 512
12000 512
13000 512
14000 512
15000 512
16000 512
17000 512
18000 512
18000 512
19000 512
20000 512
Generation = 20000
numberOfDeletions = 70
numbOfMatches = 512
numbOfUnknowns = 0
numbOfMismatches = 0
score = 519
```

Con la ayuda de gráficos, se hizo evidente que existe una gran diferencia entre las salidas sin selección y con selección. Cuando no hay selección, el valor de equilibrio del número máximo de coincidencias es de alrededor de 265. Cuando opera la selección, es de 512, es decir, la perfección y las fluctuaciones son bastante pocas.

21, página 29. Creemos que nuestra metodología no se aplica a la vida real porque la cantidad de mapas de bits es una función exponencial de la cantidad de bits y nuestro poder de simplificación es solo una fracción de ese costo. Por lo tanto, el número de perceptrones en los NN simplificados sigue siendo exponencial, es decir, nuestro enfoque es irrealizable para aplicaciones reales. Nuestro veredicto se basa en nuestra intuición y no en las matemáticas. Vale la pena investigar esta advertencia porque no sabemos cómo escala la fracción de simplificación con respecto a la longitud de los cromosomas.

22, página 29. Podemos acomodar nuestros resultados sobre la Evolución de las RN y del cerebro según el muy simple credo científico de la Evolución: La Evolución Natural comenzó con propuestas simples que surgieron por mera casualidad y que se fueron perfeccionando con el transcurso de las generaciones. Existe una tendencia natural hacia la perfección que fue posible gracias a una disminución de la tasa de mutación que siguió a un refinamiento en la fidelidad de la replicación y de los mecanismos de reparación.

23, página 29. Según nuestros resultados hasta aquí, la Evolución explica una pequeña fracción de todo el problema (la optimización moderada de la arquitectura de una RN) pero su núcleo (el origen de la arquitectura y sus componentes) permanece intacto. Por lo tanto, es vergonzoso afirmar algo a favor de la TE (Teoría Evolutiva). Pero una luz de esperanza apareció en el último experimento,

en el que se dio un valor muy bajo a la tasa de mutación. Es por eso que nos sentimos obligados a volver a ejecutar un programa de nuestra Introducción en el que la Evolución no pudo emular la perfección lograda por una RN creada por el hombre con 2 perceptrones.

El programa `EvolNN4.html` tenía como objetivo que la Evolución optimizara los valores de peso de los dos perceptrones de la RN, un elemento fundamental de los perceptrones que, en nuestra Introducción, no pudimos especificar con el uso de Evolución. Lo ejecutamos utilizando una tasa de mutación más baja, de 0.0001, y una tasa de recombinación de 0.1. El programa se ejecutó en un portátil con 16 GB de RAM y lo detuvimos después de 14 horas de trabajo. ¿Qué obtuvimos?

Destaquemos algunos resultados:

1. Se observaba una tendencia siempre ascendente por segmentos, pero de todos modos, el comportamiento caótico de los perceptrones hacía su juego y los trazos se interrumpían de vez en cuando sin alcanzar la perfección.

2. Cuanto más perfeccionada es la RN, más largo es el tiempo de espera para un nuevo estado más elevado. Quizás sea interesante observar los números que muestran las generaciones en las que se produjo una mejora:

Genertn	score	matches	mismatches	cost
0	44	64	4	100
719	64	104	8	72000
737	72	72	0	73800
904	112	112	0	90500
40222	152	152	0	4022300
61972	212	212	0	6197300
156367	212	212	0	15636800

El costo es la cantidad de fósiles que quedan hasta la generación determinada. Si se repite el experimento, los números pueden

cambiar, pero el patrón permanece. Si la tasa de mutación aumenta, los segmentos en los que la evolución siempre aumenta se acortan, pero la recuperación comienza de inmediato.

3. La perfección no crece de manera constante, sino que crece a saltos, digamos, de 152 coincidencias pasa a 212 en una generación, pero después de un tiempo de espera muy largo. Esto es compatible con la afirmación de que la evolución natural fue responsable de los saltos observados en la morfología de las conchas de los caracoles (Wikipedia, [3] 2024). Pero en nuestro caso, las mutaciones en los valores de peso no fueron macro, como podemos ver aquí:

Genertn	Weight values of P1	Weight values of P0
0	0,0,-1,-1,0,0, 0, 0,-1	1,1,0,1,-1,0,1,1,-1
719	-1,0, 0, 0,1,0,-1,-1,-1	1,0,0,0, 1,1,1,0, 1
737	-1,0, 0,-1,1,0,-1,-1,-1	1,0,0,0, 1,1,1,0, 1
904	-1,0, 0,-1,1,0,-1, 0,-1	1,0,0,0, 1,1,1,0, 1
40222	-1,0, 0,-1,1,0,-1, 0,-1	1,0,0,1, 1,1,1,0, 1
61972	-1,0, 0,-1,1,0,-1, 0,-1	1,0,1,1, 1,1,1,0, 1
156367	-1,0, 0,-1,1,0,-1, 0,-1	1,0,1,1, 1,1,1,0, 1

En 156367 generaciones con una población de 100 habitantes, el mayor número de coincidencias logrado fue 212 de 512 con cero desajustes.

Gracias a este y muchos otros experimentos que hemos realizado, reconocemos el poder creativo de la Evolución. Ahora bien, ¿ Tiene la Evolución una firma que deja en sus productos? Sí, y con mucha firmeza: junto a la propuesta de soluciones a los problemas planteados, la Evolución deja abundantes testimonios de sus muchas, muchas pruebas realizadas antes de detenerse.

Nuestros resultados se acumulan naturalmente en torno a nuestra acusación habitual contra la Teoría Evolutiva: la evolución no es mágica, no puede resolver los problemas de inmediato. Más bien, es una

herramienta ordinaria que emplea tiempo y recursos para proponer soluciones aproximadas a los problemas planteados. Así, nos preguntamos una y otra vez: ¿dónde están en el registro fósil las huellas de las demasiadas pruebas de la Evolución, vestidas de imperfecciones de todo tipo, antes de alcanzar la perfección que veo en la mayoría de los niños? Dado que el número de intentos necesarios para alcanzar el siguiente nivel de calidad crece desproporcionadamente, consideramos que no podemos afirmar que los efectos del muestreo puedan ser responsables de la inobservabilidad de la Evolución de la Perfección.

Sin embargo, nuestros resultados aislados son discutibles. Esta verdad junto con el hecho de que la Evolución es un algoritmo nos han impulsado a abstraer el problema de la Evolución de la Perfección a lo largo de la teoría general de la complejidad de los algoritmos (Rodríguez [2] 2021). La idea es que la Evolución requeriría muchas eternidades para desarrollar la vida tal como existe, y por lo tanto en el universo actual -con una vida extremadamente corta- todo lo hecho por la Evolución debe llevar su marca de imperfección. Surge así una predicción obligatoria: la imperfección será visible y abundará en mi propio cuerpo. Esto es falso. Soy un hombre muy viejo y, sin embargo, la cuasi-perfección de mi cuerpo me maravilla.

La Teoría Evolutiva es falsa.

Pero dado que la evolución es una cruda realidad, debe provocar mutaciones que deterioren el genoma. Esto plantea el problema de cómo vivir junto con la Evolución cuando se exige perfección. Consideramos que esto se hace mediante un diseño tolerante a fallos, como lo es nuestra biología, en el que soluciones preparadas previamente resuelven fallos específicos. Por ejemplo, la coagulación es necesaria para controlar las hemorragias, pero ella también puede ocurrir dentro de nuestros vasos y esto provocaría coágulos de sangre. Tolerancia a fallos significa en este caso que tenemos una maquinaria que disuelve los coágulos internos. Esta maquinaria está codificada en nuestros genomas. Por eso, un hematoma provocado por un golpe desaparece

en una o dos semanas.

6 BIBLIOGRAFÍA

References

[1] José Rodríguez (2023)
Evolutionary Neural Networks 1. Introduction.
https://ejristos.com/Evol%20Neural%20Networks/EvolNNintroSpanish.pdf 3

[2] José Rodríguez (2021)
Falsification of the Evolutionary with flipp.
https://ejristos.com/GPfalsifiesEvo/FalsifTEconFlipp.pdf 47

[3] Wikipedia contributors. (2024).
Punctuated equilibrium. *In Wikipedia, The Free Encyclopedia. Retrieved 13:53, April 2, 2024,.*
https://es.wikipedia.org/wiki/Equilibrio_puntuado
https://en.wikipedia.org/wiki/Punctuated_equilibrium 46

Estas conexiones a la red estaban activas el 30/V/2024