



UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

SELECCIÓN DE CARACTERÍSTICAS Y
CLASIFICACIÓN DE MICROARREGLOS MEDIANTE
ALGORITMOS BIOINSPIRADOS Y EL USO DEL
NEURÓN GENERALIZADO

TESIS

QUE PARA OPTAR POR EL GRADO DE:
MAESTRA EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

PRESENTA

ROMERO MONTIEL FLOR ALEJANDRA

DIRECTOR DEL TRABAJO:
DRA. KATYA RODRÍGUEZ VÁZQUEZ
POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

México, CDMX.

SEPTIEMBRE, 2018



Universidad Nacional
Autónoma de México

Dirección General de Bibliotecas de la UNAM

Biblioteca Central



UNAM – Dirección General de Bibliotecas
Tesis Digitales
Restricciones de uso

DERECHOS RESERVADOS ©
PROHIBIDA SU REPRODUCCIÓN TOTAL O PARCIAL

Todo el material contenido en esta tesis esta protegido por la Ley Federal del Derecho de Autor (LFDA) de los Estados Unidos Mexicanos (México).

El uso de imágenes, fragmentos de videos, y demás material que sea objeto de protección de los derechos de autor, será exclusivamente para fines educativos e informativos y deberá citar la fuente donde la obtuvo mencionando el autor o autores. Cualquier uso distinto como el lucro, reproducción, edición o modificación, será perseguido y sancionado por el respectivo titular de los Derechos de Autor.

DEDICATORIA

A mi familia por todo su apoyo y por la confianza depositada en mi.
A mi madre por estar a mi lado aún cuando la situación es difícil.
A mi padre, la estrella que me saluda desde el cielo.
A mi esposo por todo el amor y comprensión.
A los profesores por compartir su conocimiento, sus sabios consejos y su visión del mundo.
A CONACYT por el apoyo económico que me brinda.

Índice general

1. Introducción	1
2. Microarreglos de ADN	5
2.0.1. Metodología del microarreglo de ADN	6
2.0.2. Cuantificación de microarreglos	7
2.0.3. Estado del arte	8
3. Algoritmos bioinspirados	9
3.1. Algoritmos basados en cúmulos de partículas	9
3.1.1. Descripción del algoritmo PSO	10
3.1.2. Tipos de algoritmos de PSO	12
3.2. Algoritmos genéticos	14
3.2.1. Terminología de los algoritmos genéticos	15
3.2.2. Operadores genéticos	17
3.3. Evolución diferencial	18
3.3.1. Variantes de DE	19
3.3.2. Parámetros del algoritmo	21
4. Neurón generalizado	23
4.1. Redes neuronales artificiales	23
4.1.1. Neurón generalizado	24
5. Metodología	27
5.1. Reducción de dimensionalidad	27
5.2. Clasificación	29
5.3. GA y PSO	31
5.4. GA y AG	31
5.5. GA y DE	32

6. Resultados	33
6.1. GA y PSO	33
6.2. GA y GA	37
6.3. GA y DE	41
6.4. Resumen de los resultados	45
7. Conclusiones	47
Siglas	51
Bibliografía	53

Capítulo 1

Introducción

Los microarreglos de ácido desoxirribonucleico (ADN) son utilizados para la cuantificación masiva de la expresión de genes. Este análisis permite diagnosticar enfermedades, identificar diferentes tumores, seleccionar para un paciente específico el mejor tratamiento para resistir una enfermedad [24]. El microarreglo de ADN tiene una enorme cantidad de genes para analizar (del orden de los miles). Por lo que las bases de datos de microarreglos tienen muchas características (genes) y un pequeño número de muestras; esto es una desventaja cuando se realiza una tarea de clasificación y reconocimiento con los métodos actuales [8].

Las redes neuronales artificiales (ANN por sus siglas en inglés) son modelos computacionales usados para clasificación y otras tareas [23]. El papel de las ANN en las aplicaciones ha crecido gradualmente y como resultado, redes dotadas con la capacidad de deducir por sí mismas, la mejor manera de representar los datos, sin necesidad de haber sido programada explícitamente para ello. La red neuronal convencional que es usada generalmente se puede dividir en tres etapas o partes: entrada, oculta y salida. El problema básico al que se enfrenta en problemas altamente complejos es el gran tiempo de entrenamiento debido al número de neuronas y el número de capas en la etapa oculta. Se ha mostrado que un neurón generalizado, es capaz de sobreponerse a estas complicaciones [29, 38].

La red neuronal generalizada o neurón generalizado (GN) es otro tipo de ANN que fue desarrollado con el objetivo de reducir el diseño de una red neuronal, pues no tiene muchas conexiones, con un buen rendimiento en comparación con el perceptrón multicapa (MPL), además de ser fácil de implementar [9]. Algunas de las ventajas del GN son que requiere menos neuronas, es decir menos pesos y el tiempo de entrenamiento para la red es reducido. Por lo que esta estructura requiere menos memoria y requisitos de hardware, lo que la hace atractiva para aplicaciones baratas y rápidas. Algunos trabajos han aplicado GN para resolver

funciones de aproximación, para calcular estimaciones de densidad, predicción y problemas de clasificación [43] [28]. Por otra parte la reducción de dimensionalidad se convierte en crucial para hacer no solo manejable los datos sino también para poder obtener información relevante en el análisis de los mismos. En este trabajo se analiza un problema de clasificación de microarreglos de ADN. En los trabajos encontrados en la literatura se puede notar que comparten una característica: los autores buscan reducir la dimensionalidad de los genes porque muchos de ellos son irrelevantes y la consecuencia de conservarlos es costosa, aumenta el tiempo de cómputo, la alta complejidad y el bajo rendimiento en la clasificación o predicción de una enfermedad.

Las preguntas que se buscan responder en este trabajo son: ¿Es posible reducir el número de características (genes) en las bases de datos de microarreglos conservando la información relevante? ¿Con las características seleccionadas (genes) es posible realizar una buena clasificación?

La metodología propuesta consiste en reducir primero la dimensionalidad de los genes usando un algoritmo genético, pues en [?] muestra buen desempeño al realizar esta tarea. Luego clasificar los datos con un GN, comparando los resultados obtenidos con una ANN, donde algunos parámetros son determinados mediante tres algoritmos bioinspirados distintos: algoritmo basado en cúmulo de partículas, algoritmo genético y evolución diferencial. Dichos algoritmos son metaheurísticas, estrategias de alto nivel que usan diferentes métodos para explorar el espacio de búsqueda [32].

De las diferentes descripciones de metaheurísticas que se encuentran en la literatura se pueden destacar ciertas propiedades fundamentales que caracterizan a este tipo de métodos:

- Las metaheurísticas son estrategias generales que “guían” el proceso de búsqueda.
- El objetivo es una exploración del espacio de búsqueda eficiente para encontrar soluciones (casi) óptimas .
- Las metaheurísticas hacen uso de conocimiento del problema que se trata de resolver en forma de heurísticos específicos que son controlados de manera estructurada por una estrategia de más alto nivel.

Los detalles e implementaciones, así como los resultados obtenidos de cada algoritmo serán descritos en los capítulos siguientes.

El objetivo en este trabajo es generar un algoritmo capaz de clasificar correctamente muestras de un microarreglo para una enfermedad en particular, combinando un algoritmo bioinspirado y un neurón generalizado.

Una contribución destacable en este trabajo es la forma en que se seleccionan los genes relevantes en cada caso, pues generalmente primero se reduce la dimensión del problema y luego se aplica un algoritmo para clasificar, en este caso se toman en cuenta los resultados del clasificador para decidir qué genes conformarán la base de datos.

Capítulo 2

Microarreglos de ADN

Los microarreglos de ADN surgen de la necesidad de analizar la cantidad de información procedente de los grandes proyectos de secuenciación de genomas [2, 17]. Estas herramientas de estudio de la información genética vienen empleándose en dos aplicaciones fundamentales, la secuenciación o genotipado del ADN y el análisis de la expresión génica.

La determinación de la variación en las secuencias de ADN humano mediante microarreglos permite, entre otras cosas, identificar la huella genética de la predisposición a padecer diferentes enfermedades. Por otro lado, el empleo de microarreglos de ADN en estudios de la expresión génica está permitiendo la identificación y validación de manera mucho más rápida de genes como potenciales dianas terapéuticas ¹.

Un microarreglo de ADN (también denominado ADN *chip*, *oligonucleotide* ADN *chip* o *gene chip*) consiste en múltiples fragmentos de ADN complementario (cada uno de los cuales representa un gen diferente) adheridos a un soporte físico concreto (vidrio, plástico, silicón, etc.) y agrupados de manera ordenada -filas y columnas- según su función (receptores, hormonas, citocinas, etc.). Actualmente incluyen en su uso entre 9000 y 40000 fragmentos de cADN (genes) por cm^2 . Por lo tanto, disponen virtualmente de la expresión de todo el genoma en estudio [7, 1].

Un microarreglo de ADN sirve para determinar la expresión genética completa de un tejido en un momento determinado. Esta “*foto genética transversal*” de un tejido concreto se denomina “*transcriptoma*”. El transcriptoma, al contrario que el genoma (conjunto de todos los genes existentes en una célula), cambia continuamente en respuesta a cambios en las condiciones microambientales celulares o tisulares (temperatura, pH, hormonas, etc.). Por lo tanto, la interpretación del

¹Una diana terapéutica o blanco molecular puede definirse como el lugar del organismo donde un fármaco ejerce su acción

transcriptoma objeto de estudio requiere necesariamente de su comparación con el de un tejido control. El microarreglo de ADN permite esta comparación.

2.0.1. Metodología del microarreglo de ADN

El procedimiento para comparar el transcriptoma del tejido o cultivo celular objeto de estudio con el transcriptoma control es relativamente simple. En primer lugar, se debe aislar en ARNm de ambos tejidos y, a partir de cada uno de ellos, obtener sus correspondientes ADNc. Estas moléculas de ADNc deben marcarse con un compuesto fluorescente, que será diferente en los tejidos objeto de estudio y el control.

Lectura de microarreglos

Se utiliza un láser para excitar las moléculas fluorescentes unidas al ADNc y poder obtener la imagen en cada una de las muestras contenidas en el microarreglo. Este procedimiento se hace para cada uno de los fluoróforos obteniéndose dos imágenes, una para el fluoróforo Cy3 (rojo) y una para el fluoróforo Cy5 (verde) como puede verse en la imagen 2.1. Una vez obtenidas estas imágenes, pueden ser combinadas para obtener un aspecto visual de microarreglo. En la imagen 2.2 se muestra un ejemplo, en el que se pueden observar puntos amarillos, verdes, rojos y un sin número de tonalidades entre el verde y el rojo. Todos aquellos puntos que se ven rojos o tonalidades anaranjadas, pueden ser interpretados como genes que aumentaron su expresión. Los puntos verdes o tonalidades entre el amarillo y el verde serán aquellos genes que disminuyeron su expresión. Y finalmente los puntos amarillos representan a los genes que en ambas condiciones se expresan de forma igual.

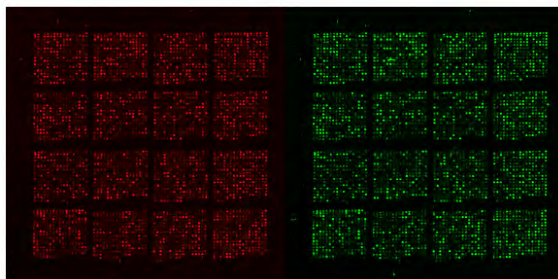


Figura 2.1: Imagen de fluorescencia para un microarreglo hibridizado con una sonda marcada con dUTP-Cy3 (rojo) y una sonda marcada con dUTP-Cy5 (verde). Tomada de [41].

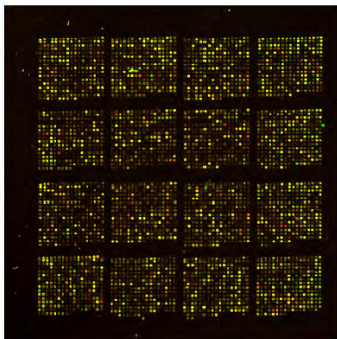


Figura 2.2: Imagen combinada de las imágenes de la imagen 2.1. Tomada de [41].

2.0.2. Cuantificación de microarreglos

Para la obtención de los valores cuantitativos de cada una de las señales de fluorescencia contenidas en el microarreglo, se requiere del análisis de las imágenes. En primer lugar se debe considerar la aplicación de un filtro para depurar la imagen de pequeñas imperfecciones o señales no deseadas que pudieran encontrarse entre las muestras. Posteriormente se genera una retícula en la que se definen las áreas que se van a cuantificar. Definida la retícula, se determina una zona de exclusión y el área para calcular la señal de fondo de la figura 2.3 -recuadro de la imagen 2.3-. Definidos estos parámetros se calcula la densidad de los pixeles en cada área definida, dando como resultado los valores de densidad, fondo y señal para cada una de las muestras en el microarreglo.

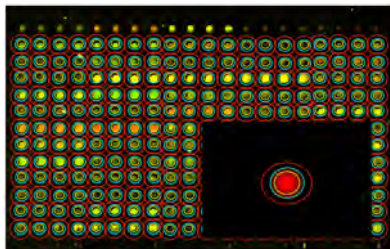


Figura 2.3: Cuantificación de la señal de un microarreglo, definición y ajuste de la retícula. En el recuadro, dentro del círculo amarillo, está la zona para obtener la densidad real, entre el círculo amarillo y el azul está la zona no considerada y entre el círculo azul y el rojo la zona para determinar la señal de fondo. Tomada de [41].

Los valores de señal se obtienen de la resta numérica de la densidad menos el fondo de cada uno de los genes estudiados. Para la interpretación de estos resultados se trata de discriminar entre los miles de datos que se obtienen, aquellos que son significativos y que representan un cambio real.

2.0.3. Estado del arte

En la literatura se han encontrado diversos trabajos donde utilizan distintas técnicas basadas en aprendizaje supervisado para la clasificación de microarreglos. En [11] diseñan un marco de técnicas basadas en aprendizaje automático para discriminar entre leucemia linfoblástica aguda y leucemia mieloide aguda, comparando los resultados obtenidos al contrastar una ANN, una Máquina de Soporte Vectorial (SVM), Naive Bayes, Regresión Logística, kvecinos cercanos (KNN) y Árboles de Clasificación, reportando una precisión de clasificación significativa de 98 % usando ANN.

También se han encontrado trabajos donde se realiza selección de genes, como en [31] donde se desarrolla un estudio comparativo entre selección secuencial hacia adelante (SFS) y algoritmos genéticos (GA) con el objetivo de identificar un grupo de genes con alto valor predictivo usando seis técnicas estándar de aprendizaje de máquinas para realizar la tarea de predecir cáncer; análisis discriminante lineal (LDA), SVM, Naive Bayes, Red Neuronal Constructiva CMANTEC, KNN y el MPL. Los mejores resultados se obtuvieron al usar GA para la selección de genes y MLP, LDA y SVM, reportando en algunos casos precisión de hasta el 100 %.

En [25] se propone un modelo híbrido de dos fases para la clasificación del cáncer, que integra la selección de características basadas en la correlación con PSO binario mejorado (iBPSO). Este modelo selecciona un conjunto de genes de baja dimensionalidad usando un clasificador Naive Bayes con validación cruzada. Los subconjuntos de genes tienen tamaño muy pequeño (hasta 1.5 %).

Los algoritmos PSO, GA y evolución diferencial (DE) son descritos en el capítulo 3, siguiendo con los pormenores del neurón generalizado en el capítulo 4, la metodología propuesta y los resultados son detallados en los capítulos 5 y 6 respectivamente, para finalizar las conclusiones en el capítulo 7.

Capítulo 3

Algoritmos bioinspirados

Los algoritmos bioinspirados son aquellos que están basados en el comportamiento y la forma de actuar de ciertos sistemas biológicos, su relevancia es debido a que son capaces de minimizar el tiempo de computación de ciertos problemas matemáticos.

3.1. Algoritmos basados en cúmulos de partículas

Un Algoritmo basado en cúmulos de partículas o *Particle Swarm Optimization* (PSO) es una técnica metaheurística basada en poblaciones e inspirada en el comportamiento social del vuelo de las bandadas de aves o el movimiento de los bancos de peces [36]. PSO fue originalmente desarrollado por el sociólogo James Kennedy y por el ingeniero electrónico Russell Eberhart en 1995 [12], basándose en un enfoque conocido como la “*metáfora social*”, que describe a este algoritmo y que se puede resumir de la siguiente forma: los individuos que conviven en una sociedad tienen una opinión que es parte de un “*conjunto de creencias*” (el espacio de búsqueda) compartido por todos los posibles individuos. Cada individuo puede modificar su propia opinión basándose en tres factores:

- Su conocimiento sobre el entorno (su valor de *fitness*).
- Su conocimiento histórico o experiencias anteriores (su memoria).
- El conocimiento histórico o experiencias anteriores de los individuos de la población.

Siguiendo ciertas reglas de interacción, los individuos en la población adaptan sus esquemas de creencias al de los individuos con más éxito de su entorno. Con

el tiempo, surge una cultura cuyos individuos tienen un conjunto de creencias estrechamente relacionado [47].

El principio natural en el que se basa PSO es el comportamiento de una bandada de aves o de un banco de peces: supongamos que una de estas bandadas busca comida en un área y que solamente hay una pieza de comida en dicha área. Los pájaros no saben dónde está la comida pero sí conocen su distancia a la misma, por lo que la estrategia más eficaz para hallar la comida es seguir al ave que se encuentre más cerca de ella. PSO emula este escenario para resolver problemas de optimización. Cada solución (partícula) es un “ave” en el espacio de búsqueda que está siempre en continuo movimiento y que nunca muere [14].

El cúmulo de partículas es un sistema multiagente, es decir, las partículas son agentes simples que se mueven por el espacio de búsqueda y que guardan (y posiblemente comunican) la mejor solución que han encontrado. Cada partícula tiene una aptitud, una *posición* y un *vector velocidad* que dirige su ‘movimiento’ [40]. El movimiento de las partículas por el espacio está guiado por las partículas óptimas en el momento actual.

Los algoritmos basados en cúmulo de partículas se han aplicado con éxito en diferentes campos de investigación. Algunos ejemplos son: optimización de funciones numéricas, entrenamiento de redes neuronales, aprendizaje de sistemas difusos, registro de imágenes, problema del agente viajero e ingeniería química [36].

En las siguientes secciones se realiza una descripción formal del algoritmo PSO, con sus principales factores, parámetros, topologías de vecindario y aspectos avanzados

3.1.1. Descripción del algoritmo PSO

Un algoritmo PSO consiste en un proceso iterativo y estocástico que opera sobre un cúmulo de partículas. La posición de cada partícula representa una solución potencial al problema que se está resolviendo. Generalmente, una partícula p_i está compuesta de tres vectores y dos valores de *fitness*:

- El vector $x_i = \langle x_{i1}, x_{i2}, \dots, x_{in} \rangle$ almacena la posición actual (localización) de la partícula en el espacio de búsqueda.
- El vector $pBest = \langle p_{i1}, p_{i2}, \dots, p_{in} \rangle$ almacena la posición de la mejor solución encontrada por la partícula hasta el momento.
- El vector de velocidad $v_i = \langle v_{i1}, v_{i2}, \dots, v_{in} \rangle$ almacena el gradiente (dirección) según el cual se moverá la partícula.
- El valor de *fitness* $fitness_{x_i}$ almacena el valor de adecuación de la solución actual (vector x_i).

- El valor de *fitness* $fitness_pBest_i$ almacena el valor de adecuación de la mejor solución local encontrada hasta el momento (vector $pBest_i$).

El cúmulo se inicializa generando las posiciones y las velocidades iniciales de las partículas. Las posiciones se pueden generar aleatoriamente en el espacio de búsqueda, de forma regular o con una combinación de ambas formas. Una vez generadas las posiciones, se calcula el *fitness* de cada una y se actualizan los valores de $fitness_x_i$ y $fitness_pbest_i$.

Las velocidades se generan aleatoriamente, con cada componente en el intervalo $[-v_{max}, v_{max}]$, donde v_{max} será la velocidad máxima que pueda tomar una partícula en cada movimiento. No es conveniente fijarlas a cero pues no se obtienen buenos resultados [13].

Inicializando el cúmulo, las partículas se deben mover dentro del proceso iterativo. Una partícula se mueve desde una posición del espacio de búsqueda hasta otra, simplemente, añadiendo al vector posición x_i el vector velocidad v_i para obtener un nuevo vector:

$$x_i^{k+1} \leftarrow x_i^k + v_i^{k+1} \quad (3.1)$$

Una vez calculada la nueva posición de la partícula, se evalúa actualizando $fitness_x_i$. Además, si el nuevo *fitness* es el mejor encontrado hasta el momento, se actualizan los valores de mejor posición $pBest_i$ y *fitness* $fitness_pBest_i$. El vector velocidad de cada partícula es modificado en cada iteración utilizando la velocidad anterior, un componente *cognitivo* y un componente *social*. El modelo matemático resultante y que representa el corazón del algoritmo PSO viene representado por las siguientes ecuaciones [27]:

$$v_i^{k+1} = w * v_i^k + \varphi_1 * rand_1 * (pBest_i - x_i^k) + \varphi_2 * rand_2 * (g_i - x_i^k) \quad (3.2)$$

$$x_i^{k+1} = x_i^k + v_i^{k+1} \quad (3.3)$$

La ecuación 3.2 refleja la actualización del vector velocidad de cada partícula i en cada iteración k . El componente *cognitivo* está modelado por el factor $\varphi_1 * rand_1 * (pBest_i - x_i^k)$ y representa la distancia entre la posición actual y la mejor conocida por esa partícula, es decir, la decisión que tomará la partícula influenciada por su propia experiencia a lo largo de su vida. El componente *social* está modelado por $\varphi_2 * rand_2 * (g_i - x_i^k)$ y representa la distancia entre la posición actual y la mejor posición del vecindario, es decir, la decisión que tomará la partícula según la influencia que el resto del cúmulo ejerce sobre ella. Una descripción más detallada de cada factor se realiza a continuación:

- v_i^k : velocidad de la partícula i en la iteración k
- w : factor de inercia

- φ_1, φ_2 : son tasas de aprendizaje (pesos) que controlan los componentes *cognitivo* y *social*
- $rand_1, rand_2$: números aleatorios entre 0 y 1.
- x_i^k : posición actual de la partícula i en la iteración k
- $pBest_i$: mejor posición (solución) encontrada por la partícula i hasta el momento
- g_i : representa la posición de la partícula con el mejor $pBest_fitness$ del entorno p_i (*lBest* o *localbest*) o de todo el cúmulo (*gBest* o *globalbest*).

La ecuación 3.3 modela el movimiento de cada partícula i en cada iteración k . En la figura 3.1 se muestra una representación gráfica del movimiento de una partícula en el espacio de soluciones.



Figura 3.1: Movimiento de una partícula en el espacio de soluciones. Tomada de [30]

En esta gráfica, las flechas de línea discontinua representan la dirección de los vectores de velocidad actual: v_{pBest}^k es la velocidad de la mejor posición tomada por la partícula, v_g^k es la velocidad de la mejor partícula encontrada en el vecindario y v^k es la velocidad actual de la partícula. La flecha de línea continua representa la dirección que toma la partícula para moverse desde la posición x^k hasta la posición x^{k+1} . El cambio de dirección de esta flecha depende de la influencia de las demás direcciones (gradiente) que intervienen en el movimiento.

3.1.2. Tipos de algoritmos de PSO

Se pueden obtener diferentes tipos de PSO atendiendo a diversos factores de configuración, por ejemplo, según la importancia de los pesos *cognitivo* y *social* y

según el tipo de vecindario utilizado. Por una parte, dependiendo de la influencia de los factores *cognitivos* y *social* (valores φ_1 y φ_2 respectivamente) sobre la dirección de la velocidad que toma una partícula en el movimiento (ecuación 3.3), Kennedy [26] identifica cuatro tipos de algoritmos:

- Modelo completo: $\varphi_1, \varphi_2 > 0$. Tanto el componente *cognitivo* como el *social* intervienen en el movimiento.
- Modelo sólo *Cognitivo*: $\varphi_1 > 0$ y $\varphi_2 = 0$. Únicamente el componente *cognitivo* interviene en el movimiento.
- Modelo sólo *Social*: $\varphi_1 = 0$ y $\varphi_2 > 0$. Únicamente el componente *social* interviene en el movimiento.
- Modelo sólo *Social* exclusivo: $\varphi_1 = 0$, $\varphi_2 > 0$ y $g_i \neq x_i$. La posición de la partícula en sí no puede ser la mejor de su entorno.

Existen estudios realizados sobre la influencia de las tasas de aprendizaje [13], en este trabajo se usará el modelo completo.

Por otra parte, desde el punto de vista del vecindario, es decir, la cantidad y posición de las partículas que intervienen en el cálculo de la distancia en el componente *social*, se clasifican dos tipos de algoritmos: PSO Local y PSO Global [36].

Algoritmo 1 PSO Local

```

S ← Inicializar Cumulo
mientras no se alcance la condición de parada hacer
  para  $i = 1$  hasta  $size(S)$  hacer
    evaluar cada partícula  $x_i$  del cumulo  $S$ 
    si  $fitness(x_i)$  es mejor que  $fitness(pBest_i)$  entonces
       $pBest_i \leftarrow x_i$ ;  $fitness(pBest_i) \leftarrow fitness(x_i)$ 
    fin si
  fin para
  para  $i = 1$  hasta  $size(S)$  hacer
    escoger  $lBest_i$ , la partícula con mejor fitness del entorno  $x_i$ 
     $v_i^{k+1} \leftarrow w * v_i^k + \varphi_1 * rand_1 * (pBest_i - x_i^k) + \varphi_2 * rand_2 * (lBest_i - x_i^k)$ 
     $x_i^{k+1} \leftarrow x_i^k + v_i^{k+1}$ 
  fin para
fin mientras
devolver la mejor solución encontrada

```

En el PSO Local, se calcula la distancia entre la posición actual de la partícula y la posición de la mejor partícula encontrada en el entorno local de la primera. El

entorno local consiste en las partículas inmediatamente cercanas en la topología del cúmulo. En el Algoritmo 1 se muestra el pseudocódigo de la versión Local de PSO.

Algoritmo 2 PSO Global

S \leftarrow Inicializar Cúmulo
mientras no se alcance la condición de parada **hacer**
 para $i = 1$ hasta $size(S)$ **hacer**
 evaluar cada partícula x_i del cúmulo S
 si $fitness(x_i)$ es mejor que $fitness(pBest_i)$ **entonces**
 $pBest_i \leftarrow x_i$; $fitness(pBest_i) \leftarrow fitness(x_i)$
 fin si
 si $fitness(pBest_i)$ es mejor que $fitness(gBest)$ **entonces**
 $gBest \leftarrow pBest_i$; $fitness(gBest) \leftarrow fitness(pBest_i)$
 fin si
 fin para
 para $i = 1$ hasta $size(S)$ **hacer**
 $v_i^{k+1} \leftarrow w * v_i^k + \varphi_1 * rand_1 * (pBest_i - x_i^k) + \varphi_2 * rand_2 * (gBest - x_i^k)$
 $x_i^{k+1} \leftarrow x_i^k + v_i^{k+1}$
 fin para
fin mientras
devolver la mejor solución encontrada

Para el PSO Global, la distancia en el componente *social* viene dada por la diferencia entre la posición de la partícula actual y la posición de la mejor partícula encontrada en el cúmulo $gBest_i$ (pseudocódigo en el Algoritmo 2).

La versión Global converge más rápido pues la visibilidad de cada partícula es mejor y se acerca más a la mejor del cúmulo favoreciendo la intensificación, por esta razón, también cae más fácilmente en óptimos locales. El comportamiento de la versión Local es el contrario, es decir, le cuesta más converger favoreciendo en este caso la diversificación, pero no cae fácilmente en óptimos locales. En este trabajo se usó PSO Global [36].

3.2. Algoritmos genéticos

Los Algoritmos Genéticos (GA por sus siglas en inglés) son métodos adaptativos que pueden usarse para resolver problemas de búsqueda y optimización, son capaces de encontrar soluciones para problemas complejos [4].

Los principios básicos de los Algoritmos Genéticos fueron establecidos por Holland [22], y se encuentran bien descritos en varios textos -Goldberg [18], Davis

[10], Michalewicz [34], Reeves [42]-.

En el caso de que existan técnicas especializadas para resolver un determinado problema, lo más probable es que superen al Algoritmo Genético, tanto en rapidez como en eficacia. El gran campo de aplicación de los Algoritmos Genéticos se relaciona con aquellos problemas para los cuales no existen técnicas especializadas [44]. En algoritmo 3 se muestra el pseudocódigo del Algoritmo Genético Simple.

Algoritmo 3 Algoritmo Genético Simple

```

S ← Generar una población inicial S
mientras no se alcance la condición de parada hacer
  evaluar cada individuo de la población S
  para  $i = 1$  hasta  $size(S)/2$  hacer
    Seleccionar dos individuos de la generación anterior.
    Cruzar con cierta probabilidad los dos individuos obteniendo dos descendientes.
    Mutación de individuos.
    Insertar los dos descendientes mutados en la nueva generación
  fin para
fin mientras
devolver la mejor solución encontrada

```

3.2.1. Terminología de los algoritmos genéticos

La población

Es un conjunto de individuos que representan el conjunto de soluciones evaluadas durante una generación (iteración). Idealmente la primera población debería estar formada por individuos contenidos en todo el espacio de búsqueda. Es por esto que la generación de la población inicial usualmente es aleatoria para impedir una convergencia prematura hacia soluciones próximas a óptimos locales. No obstante, en determinados casos donde se conoce la proximidad del óptimo global, se suelen emplear soluciones conocidas previamente [37].

La determinación del tamaño de la población no es una tarea fácil, este depende de la complejidad del problema, de modo que cuando mayor es el espacio de búsqueda, mayor debe ser el tamaño de la población para evitar un estancamiento prematuro en un óptimo relativo. Es evidente que cuanto mayor sea el número de individuos se explorarán más zonas del espacio de soluciones, pero también es bastante obvio que esto acarreará un costo computacional mayor, necesitando mayor tiempo de cálculo y, a veces, de convergencia. Por eso se debe buscar un

compromiso entre el número de individuos utilizados y la calidad que se desea alcanzar [21].

Los individuos

Es necesario codificar de alguna manera el dominio del problema para obtener estructuras manejables que puedan ser manipuladas por el AG. Cada una de estas estructuras constituye el equivalente al genotipo de un individuo en términos biológicos. El elemento del dominio del problema al que se mapea este genotipo es el análogo al fenotipo. Una vez que se ha definido la manera de codificar los elementos del dominio del problema y se conoce la forma de pasar de un elemento a su código y viceversa, es necesario fijar un punto de partida. En resumen, un individuo representa una de las soluciones del problema planteado. Para determinar cuáles de estos individuos corresponden a buenas propuestas de solución y cuáles no, es necesario calificarlos de alguna manera. Cada individuo de cada generación de un algoritmo genético recibe una calificación o, para usar el término biológico, una medida de su grado de adaptación. El objetivo de este número es que permita distinguir propuestas de solución buenas de aquéllas que no lo son.

Los genes

Es una porción del cromosoma que generalmente codifica el valor de un sólo parámetro acotado dentro de un cierto rango o dominio. La cantidad de valores que puede tomar un parámetro suele ser discreta y debe tenerse en cuenta que a mayor número, más complejo tiende a ser el espacio de búsqueda. La elección de la codificación dependerá también del problema a resolver.

La función de aptitud

La adaptación de cada individuo de una población al medio se realiza mediante la función de aptitud, ϕ o *fitness*. Para cuantificar esta adaptación primero debe decodificarse el cromosoma, ya que esta función está definida en el espacio fenotípico. El grado de adaptación de un individuo será el que determine su probabilidad de reproducirse o incluso su eliminación de la población.

Codificación de las variables de diseño

Desde la aparición de los Algoritmos Genéticos se han utilizado diferentes formas de codificación binaria, real, hexadecimal, etc. Sin embargo la mejora evolutiva es independiente de la codificación, como se demuestra en el artículo de Antonisse [3] o como Goldberg advertía. “El empleo de codificaciones reales o de punto flotante tiene un papel controvertido en la historia de los esquemas de

búsqueda genética y evolutiva, y su uso parece estar últimamente al alza. Este incremento en su uso ha sorprendido a los investigadores familiarizados con la teoría fundamental de los Algoritmos Genéticos, porque un análisis simple parece sugerir que el uso de alfabetos de baja cardinalidad mejora el proceso de esquemas, lo cual es aparentemente una contradicción directa con los resultados empíricos donde codificaciones reales han funcionado bien en un gran número de problemas prácticos” [19]. Es importante distinguir entre dos conceptos, que a menudo se confunden en la literatura, como son la infactibilidad (infeasible) y la legalidad. Una solución no factible se genera cuando ésta no cumple con alguna o varias de las restricciones que se imponen al problema, es decir, de no existir éstas serían una solución factible. La factibilidad de una solución depende de las restricciones y no de la codificación, en estos casos existen diversas técnicas que obligan a los individuos a evolucionar hacia la zona de soluciones posibles.

Por otro lado una solución ilegal se genera debido a la naturaleza de la codificación. dado que los cromosomas ilegales no pueden traducirse al espacio de fenotipos, no pueden ser evaluados y por lo tanto no se puede emplear una técnica de penalización. Pueden aplicarse otras técnicas como la reparación que transforman los genotipos ilegales en genotipos legales, aunque puede que no factibles. Otra técnica consiste en comprobar la legalidad de individuos tras su nacimiento, el cual en caso de ser ilegal será eliminado y sustituido por un nuevo individuo.

3.2.2. Operadores genéticos

Los operadores de reproducción

Los Algoritmos Genéticos se fundamentan en el proceso de la reproducción, son estos operadores los que dominan el proceso de búsqueda. La exploración hace referencia a la búsqueda de soluciones más allá de las soluciones codificadas por los padres, mientras que la explotación se refiere a la búsqueda del óptimo guiada y acotada por las soluciones de los padres.

El dilema *exploración vs explotación* ya fue planteado por Holland [22] y plantea la problemática de los Algoritmos Genéticos. Una exploración demasiado intensiva hace el proceso de búsqueda lento al no generar suficientes esquemas, mientras que una explotación excesiva puede provocar la convergencia prematura en un óptimo local. Por otra parte el ajuste entre ambos es dependiente del tipo de problema a resolver por lo que debe ser establecido según cada caso [50].

El operador de mutación

Este operador es el responsable de realizar búsquedas más allá de los puntos generados por los genomas de los padres, es responsable por lo tanto de la explo-

ración y, cuando el algoritmo converge y la población pierde diversidad, es casi el único mecanismo explorativo de este.

Al igual que las mutaciones naturales los hijos mutados usualmente tienen una aptitud muy inferior a los padres, por lo que el porcentaje de individuos mutados no puede ser muy elevado. Pero también es cierto que muchas veces sirve para saltar de un óptimo local a otro e incluso al óptimo global. En el caso de los algoritmos genéticos, no podemos confiar la exploración exclusivamente a este operador porque en ese caso se comportaría como una búsqueda aleatoria, lo cual es del todo indeseable.

Técnicamente es un operador unario porque solo actúa sobre un individuo, a diferencia de los operadores reproductivos. La probabilidad de mutación de cada gen de un individuo suele ser $p_m \in [0.001, 0.01]$. Aunque este parámetro debe ajustarse de acuerdo al problema que se esté resolviendo.

El operador de selección

La selección es el proceso mediante el cual se selecciona por diferentes procedimientos individuos de la población (padres) para participar en el proceso de reproducción. El conjunto de los individuos seleccionados formará a los hijos tras aplicarles los operadores de cruce y mutación. Se mencionara solo aquella que se utilizó.

- *La selección por torneo* propuesta por Wetzel [48] y popularizada por Deb y Goldberg [20] es de las más efectivas y sencillas de implementar. Se seleccionan dos o más individuos de la población de modo que compitan entre sí para ser uno de los padres de la nueva generación. El vencedor será el individuo con mayor aptitud, repitiendo este proceso hasta elegir todos los padres. Este algoritmo tiene la ventaja de no requerir la ordenación de la población en función de su aptitud ¹ con el correspondiente ahorro computacional. Al no utilizar la aptitud para realizar la selección se asegura la diversidad de la población. Debido a sus características [5] es uno de los operadores de selección más eficientes y suele conducir a la obtención de óptimos globales [49].

3.3. Evolución diferencial

Evolución Diferencial (DE por sus siglas en inglés) es un algoritmo evolutivo que a primera vista no está basado en ningún proceso natural. Fue propuesto por Storm y Price [46, 39] en 1997. Es un modelo evolutivo que enfatiza la mutación

¹Solo se utiliza para determinar si un individuo es mejor que su oponente

y utiliza un operador de cruce/recombinación a posteriori de la mutación. Fue propuesto para optimización con parámetros reales. Se trata de una técnica no determinista basada en la evolución de una población de vectores (individuos) de valores reales que representan las soluciones en el espacio de búsqueda. La generación de nuevos individuos se lleva a cabo mediante operadores diferenciales de mutación y cruce.

En la *inicialización* se definen previamente los límites inferiores y superiores para cada variable de decisión $lim_{inf} \leq v_i \leq lim_{sup}$; posteriormente se seleccionan aleatoria y uniformemente los valores iniciales de las variables de decisión sobre los intervalos $[lim_{inf}, lim_{sup}]$.

Mediante la *mutación diferencial* se añade la diferencia proporcional de dos individuos (v_2, v_3) elegidos aleatoriamente de la población a un tercer individuo (v_1 , *individuo objetivo*) también elegido aleatoriamente. El nuevo individuo w_i se denomina *individuo mutado* o *vector mutado*. Los vectores v_1, v_2, v_3 son mutuamente exclusivos (distintos entre sí) y a su vez diferentes de w_i .

$$w_i = v_1 + \mu * (v_2 - v_3) \quad (3.4)$$

La constante de mutación $\mu > 0$ establece el rango de diferenciación entre los individuos v_2 y v_3 con el objetivo de evitar el estancamiento en el proceso de búsqueda. Tras la *mutación*, se realiza una operación de *recombinación* sobre cada individuo v_i (*individuo objetivo*) para generar un *individuo intermedio* u_i (*trial*). El individuo intermedio u_i es construido mezclando las componentes de w_i y v_i , bajo una probabilidad predefinida $C_r \in [0, 1]$.

$$u_i(j) = \begin{cases} w_i(j) & \text{si } rand \leq C_r \\ v_i(j) & \text{en otro caso} \end{cases} \quad (3.5)$$

Finalmente el *operador de selección* decide con base a la mejora del fitness, si el individuo intermedio u_i es aceptado y reemplaza al individuo objetivo v_i ; o si por el contrario el individuo intermedio es rechazado y se conserva el individuo objetivo en la siguiente generación. Además del operador de mutación binario, *bin*, descrito anteriormente existen otros operadores que dan paso a distintas variantes de este modelo.

En el algoritmo 4 se muestra el pseudocódigo para la versión *DE/rand/1/bin*.

3.3.1. Variantes de DE

Existen diferentes variantes de evolución diferencial. Las más importantes están descritas en [33]:

■

DE/rand/1/bin

Algoritmo 4 Evolución Diferencial

```

 $G = 0$ 
Inicializar la población de soluciones  $\vec{x}_{i,G} \forall i, i = 1, \dots, M$ 
Evaluar la población  $f(\vec{x}_{i,G} \forall i, i = 1, \dots, M)$ 
para  $G = 1$  hasta  $MAX\ NUM\ ITERACIONES$  hacer
  para  $i = 1$  hasta  $M$  hacer
    seleccionar  $r_1 \neq r_2 \neq r_3$  aleatoriamente
     $j_{rand} = randint(1, D)$ 
    para  $j = 1$  hasta  $D$  hacer
      si  $rand_j[0, 1) < CR$  o  $j = j_{rand}$  entonces
        calcula el valor  $u_{i,j,G+1} = X_{r3,j,G} + F * (x_{r1,j,G} - x_{r2,j,G})$ 
      si no
         $u_{i,j,G+1} = x_{i,j,G}$ 
      fin si
    fin para
    si  $f(\vec{u}_{i,G+1}) \leq f(\vec{x}_{i,G})$  entonces
      Seleccionar  $\vec{x}_{i,G+1} = \vec{u}_{i,G+1}$ 
    si no
      Seleccionar  $\vec{x}_{i,G+1} = \vec{x}_{i,G}$ 
    fin si
  fin para
   $G = G + 1$ 
fin para
devolver la mejor solución encontrada
  
```

El modelo explicado hasta ahora.

$$w_i = v_1 + \mu * (v_2 - v_3)$$

■

$$DE/best/1/bin$$

En este modelo se elige al mejor vector para la recombinación.

$$w_i = best + \mu * (v_1 - v_2)$$

■

$$DE/rand/2/bin$$

Reemplazando el valor 1 (número de pares de soluciones elegidas), por un valor mayor, es posible una selección entre más de un par de soluciones.

$$w_i = v_1 + \mu * (v_2 + v_3 - v_4 - v_5)$$

■

$$DE/best/2/bin$$

En este modelo se elige al mejor vector para la recombinación y dos aleatorios.

$$w_i = best + \mu * (v_1 + v_2 - v_3 - v_4)$$

En este trabajo se utilizó

$$DE/rand/1/bin$$

.

3.3.2. Parámetros del algoritmo

Valores de μ en el rango $[0.4, 1]$ y C_r en el rango $(0, 1]$ suelen ser efectivos para los parámetros de DE cuando se utiliza el esquema *DE/rand/1/bin*. Sin embargo, estos parámetros suelen depender en gran medida del problema que está siendo optimizado.

Un valor grande de μ permite una mayor exploración del espacio de búsqueda, mientras que un valor pequeño permite mayor una explotación de las zonas con buenas soluciones.

Con el operador de cruza, el valor de C_r es una indicación de cuántos cambios se esperan en los vectores de la población. Un valor pequeño de C_r hará que pocas coordenadas del vector cambien, y por tanto, los vectores se moverán de forma ortogonal. Un valor grande de C_r hará que los vectores se puedan mover en cualquier dirección.

Capítulo 4

Neurón generalizado

4.1. Redes neuronales artificiales

Las redes neuronales artificiales son modelos inspirados en el funcionamiento del cerebro, representan una alternativa para resolver cierto tipo de problemas que requieren de una gran rapidez y que, en las computadoras seriales, por muy poderosas que sean la obtención de la solución sería muy lenta o imposible, por ejemplo en [35] se reporta que una ANN es capaz de predecir el consumo de éxtasis con un margen de error pequeño a partir de las respuestas dadas a un cuestionario. Las redes están constituidas por un arreglo de elementos, procesadores, o neuronas artificiales, que pueden tener una memoria local muy simple. A estos elementos se les llama nodos, procesadores o neuronas formales para remarcar que son sólo una imitación de las neuronas biológicas.

Estas neuronas están interconectadas de acuerdo a una arquitectura específica, y distribuidas en una o más capas de nodos y con pesos que indican el valor de la conexión entre dos procesadores y que se puede modificar. Su forma de procesamiento es paralela y distribuida ya que trabajan simultáneamente.

Los modelos de redes neuronales tratan de retomar y abstraer aquellos mecanismos que son los responsables del procesamiento de información, sacrificando muchos detalles de los sistemas biológicos. Los nodos procesan información de acuerdo con una regla preestablecida, la cual puede ser definida arbitrariamente, siempre y cuando sea local, es decir, que dependa solamente de los valores de entrada y de los valores almacenados en la memoria local de ese procesador.

La eficiencia de las conexiones entre los procesadores tienen un valor llamado *peso* y el proceso de entrenamiento de una red consiste en encontrar los valores adecuados para los *pesos* ya que de esto depende el comportamiento de una red. El proceso de aprendizaje es pues la modificación de las interacciones hasta encontrar

los valores con los cuales la red produce las respuestas correctas.

De cierta forma los pesos representan la influencia que puede tener una neurona sobre otra a la que está conectada, esto es, la fuerza que excita o inhibe. Por lo tanto, el nivel de actividad de cada neurona es función de las entradas que recibe y el resultado se envía como una señal a través de sus conexiones con otras neuronas. Cada neurona sólo puede dar un valor de salida al mismo tiempo.

Una red neuronal está caracterizada fundamentalmente por :

- 1. La topología de la red o interconexiones.
- 2. Las funciones de transferencia de las neuronas, las capas ocultas, capas de entrada y salida.
- 3. El valor de los pesos y umbrales.

4.1.1. Neurón generalizado

El neurón generalizado (GN por sus siglas en inglés) fue propuesto en [29] y ha sido aplicado en problemas de clasificación y aproximación de funciones [9].

La estructura general del modelo son dos funciones de agregación (suma y producto) y dos funciones de transferencia (sigmoideal y gaussiana). Por lo tanto el GN tiene flexibilidad y resistencia a las no linealidades. En la figura 4.1 se puede observar la estructura del neurón generalizado.

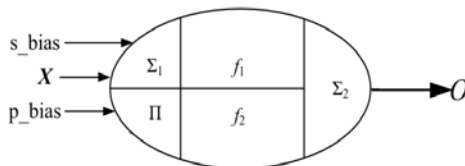


Figura 4.1: Neurón Generalizado. Modificada de [29]

A diferencia del modelo de neurona común que tiene o bien funciones de agregación \prod (producto) o \sum (suma), el modelo de neurón generalizado tiene ambas funciones de agregación. La función sigmoideal característica (f_1) se usa con la función de suma \sum_1 , mientras que la función gaussiana (f_2) se usa con la función producto \prod . La salida de la parte \sum_1 con la función sigmoideal de activación es :

$$O_{\Sigma} = f_1(S_{net}) = \frac{1}{1 + \exp(-\lambda_s * S_{net})}$$

donde:

$$S_{net} = \sum W_{\Sigma_i} X_i + X_{0\Sigma}$$

y λ_s , $X_{O\Sigma}$ son la ganancia y el sesgo de la parte Σ respectivamente. La salida de la parte Π con la función de activación gaussiana para f_2 es:

$$O_{\Pi} = f_2(P_{net}) = \exp(-\lambda_p * (P_{net}))$$

donde

$$P_{net} = \Pi W_{\Pi i} X_i * X_{O\Pi}$$

y λ_p , $X_{O\Pi}$ son la ganancia y el sesgo de la parte Π respectivamente. La salida final O_{pk} del neuron es una función de las dos salidas O_{Σ} y O_{Π} con los pesos W y $1 - W$ respectivamente y se puede escribir como:

$$O_{GN} = W * O_{\Sigma} + (1 - W) * O_{\Pi}$$

Para problemas de múltiples salidas, diversos modelos de neurón generalizado en paralelo son requeridos. El número de pesos en el caso del neurón generalizado es el doble del número de entradas más dos pesos del sesgo, más dos pesos de la ganancia más un peso que corresponde a W (porcentaje de contribución de cada una de las partes de la estructura del neurón: suma y producto). Esto es mucho menos comparado con el número de pesos en una red multicapa. Al reducir el número de pesos desconocidos el tiempo de entrenamiento se reduce.

Capítulo 5

Metodología para clasificar microarreglos de ADN

En este trabajo se exploran tres metodologías para desempeñar una tarea de clasificación binaria de microarreglos de ADN las cuales se dividen en dos etapas:

- La primera dedicada a seleccionar el conjunto de genes que mejor describen el microarreglo de ADN.
- La segunda enfocada a entrenar un neurón generalizado para mejorar la precisión de la tarea de clasificación.

El primer paso propone una reducción dimensional del microarreglo de ADN, aminorando el número de características con las cuales será entrenado el neurón generalizado. El segundo paso consiste en utilizar esta información para entrenar un neurón generalizado y realizar la clasificación. Estos dos pasos son repetidos hasta que el número máximo de iteraciones en cada caso es alcanzado.

Una descripción más detallada de las etapas se describen a continuación:

5.1. Reducción de dimensionalidad

De acuerdo con [16], la selección del conjunto con los mejores genes puede ser definido en términos de un problema de optimización. En este trabajo se usó un algoritmo genético para explorar el espacio de soluciones. Los individuos son subconjuntos de genes del microarreglo de ADN de longitud fija. La función *fitness*, es el mínimo número de elementos mal etiquetados al usar esos genes para entrenar un GN. Con la base propuesta se entrenan distintos modelos de GN, cada uno

de ellos con un resultado de clasificación distinta. Tomando como valor de *fitness* la clasificación con menos errores. Un ejemplo de codificación es:

$$[19, 58, 7, 325, 205] \quad (5.1)$$

las entradas se encuentran entre 0 y el número de características que tiene la base original menos uno, indican los índices de las columnas que serán tomadas para formar una nueva base de datos como se muestra en la figura 5.1.

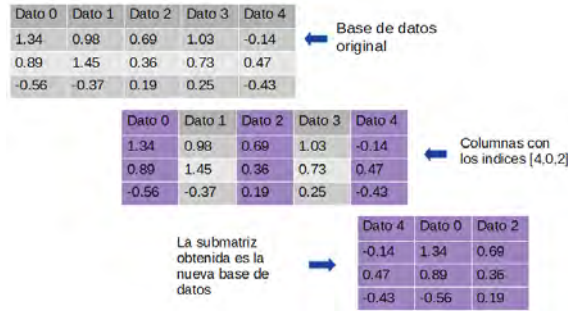


Figura 5.1: Creación de las nuevas bases de datos

La cruza usada es: dados los individuos $Ind_1 = [I_{11}, I_{12}, \dots, I_{1n}]$ y $Ind_2 = [I_{21}, I_{22}, \dots, I_{2n}]$, se generan dos descendientes, $H_k = [h_{k1}, h_{k2}, \dots, h_{kn}]$, $k = 1, 2$ donde:

$$H_1 = Ind_1 + \alpha * (Ind_2 - Ind_1) \quad (5.2)$$

$$H_2 = Ind_2 + \alpha * (Ind_1 - Ind_2) \quad (5.3)$$

con $\alpha \in [-0.25, 1.25]$. La mutación es un operador unario, dado un individuo $Ind_1 = [I_{11}, I_{12}, \dots, I_{1n}]$ este tiene una probabilidad de mutar del 10% (en la mayoría de los casos es baja). Si el individuo muta, cada entrada I_{1k} tendrá la probabilidad $\frac{1}{n}$ de mutar, es decir:

$$Im_{1k} = I_{1k} + (\beta - 0.5) * (p_2 - p_1) * 0.1 \quad (5.4)$$

donde $\beta \in (0, 1)$, $p_2 = 4$ y $p_1 = -4$. Después de aplicar estos operadores, se aplica la función piso a todos los individuos, comprobando en cada individuo que las entradas son todas distintas entre sí, de no ser el caso se reemplazará con un aleatorio.

5.2. Clasificación

Una vez propuesto el conjunto de características, estos genes formarán una nueva base de datos que es particionada en dos conjuntos: entrenamiento y validación. El conjunto de entrenamiento posee el 70 % y el conjunto de validación el 30 % restante. Esta partición se realiza de forma aleatoria para asegurar que los conjuntos contengan elementos con ambas etiquetas. Después el GN es entrenado usando uno de estos tres algoritmos: PSO, Algoritmo Genético o Evolución Diferencial, todos con codificación real; en [6] se reporta un desempeño muy similar al entrenar una ANN con PSO y GA.

Las soluciones generadas (individuos), codifican la estructura del GN en términos de los pesos sinápticos (W_{Σ}, W_{Π}, W), sesgo y parámetros de la función de activación (λ) para cada tipo de neurona (Σ y Π); las entradas se encuentran en el intervalo $(-4, 4)$. Un ejemplo de codificación es :

$$\begin{aligned}
 & [0.98, 2.12, -3.45, -0.34, 1.52, \\
 & \quad 0.872, -3.369, 2.548, -0.125, 1.417, \\
 & \quad - 2.834, 0.723, -1.396, 2.196, -3.592] \quad (5.5)
 \end{aligned}$$

si el número de características en la base es 5, la longitud de los individuos es de $(2 * 5) + 5$. La función escalón es aplicada a la salida del GN para determinar la clase a la que pertenece. En la figura 5.3 se muestra la gráfica de dicha función. La función *fitness* es el número de elementos mal etiquetados, se obtiene la matriz de confusión comparando las etiquetas reales con las etiquetas propuestas; el número de falsos negativos más falsos positivos es el valor de la función.

En la figura 5.2 se puede notar que la base de datos reducida, sirve como entrada a cada uno de los neurones generalizados, cada configuración propuesta de pesos sinápticos es probada, obteniendo el número de elementos clasificados en el conjunto de entrenamiento y validación, el valor de *fitness* es definido como el mínimo número de elementos mal etiquetados en el entrenamiento, si se produce un empate se decide usando el mínimo número de elementos mal etiquetados en la validación.

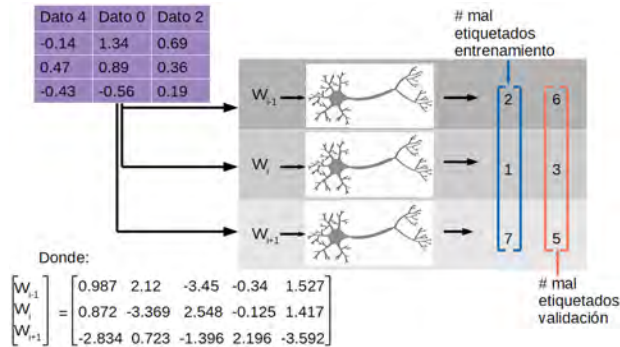


Figura 5.2: Se muestra como despues de reducir el número de características en la base de datos (genes), esta información es usada como entrada para el neurón generalizado, combinandose con los pesos sinapticos para realizar una clasificación.

Una vez entrenado el GN, se procede a evaluar la capacidad de generalización usando el conjunto de prueba; las métricas usadas son precisión, *recall* y *f1-score*.

- La *precisión* es el cociente $\frac{tp}{(tp+fp)}$ donde *tp* es el número de verdaderos positivos y *fp* el número de falsos positivos. La precisión es intuitivamente la capacidad del clasificador de no etiquetar como positiva una muestra que es negativa.
- *Recall* es el cociente $\frac{tp}{(tp+fn)}$ donde *tp* es el número de verdaderos positivos y *fn* es el número de falsos negativos. Intuitivamente es la habilidad para encontrar las muestras positivas.
- F_1 es la media armónica de precisión y recall, definida como $2 * \frac{\text{precisión} * \text{recall}}{\text{precisión} + \text{recall}}$, alcanza su valor máximo en 1 (precisión y recall perfecto), el valor mínimo es cero.

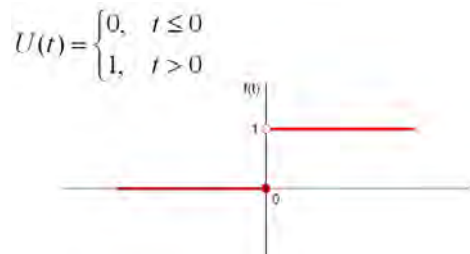


Figura 5.3: Gráfica de la función escalón de Heaviside

5.3. GA y PSO

La primer versión consiste en usar un algoritmo genético para reducir la dimensión y utilizar un PSO para entrenar un neurón generalizado. La representación esquemática se puede observar en la figura 5.4.

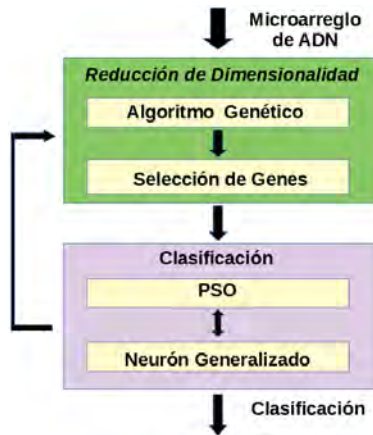


Figura 5.4: Esquema de la metodología AG y PSO

5.4. GA y AG

La segunda versión consiste en usar un algoritmo genético (genético 2) para reducir la dimensión y utilizar también un algoritmo genético (genético 1) para entrenar un neurón generalizado, la representación esquemática se puede observar en la figura 5.5.

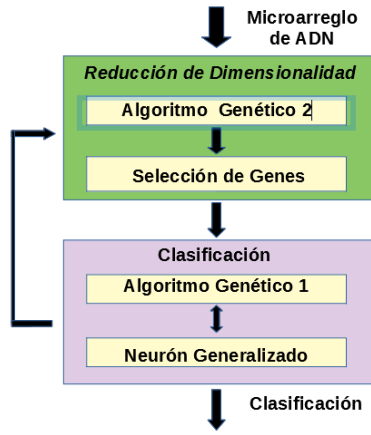


Figura 5.5: Esquema de la metodología AG y AG

5.5. GA y DE

La tercer versión consiste en usar un algoritmo genético para reducir la dimensión y utilizar evolución diferencial para entrenar un neurón generalizado. La representación esquemática se puede observar en la figura 5.6.

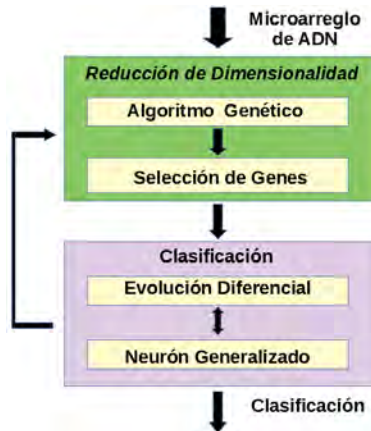


Figura 5.6: Esquema de la metodología AG y DE

Capítulo 6

Resultados

En este capítulo, se analizarán los resultados obtenidos con las metodologías propuestas, usando distintas bases de microarreglos. En los casos donde la base de datos se encuentra dividida en datos de entrenamiento y validación, se reagrupa y se realiza una partición aleatoria, 70 % para entrenamiento y 30 % para validación.

6.1. GA y PSO

Para el algoritmo genético se definieron los parámetros de forma empírica: selección por torneo, probabilidad de cruce = 100 %, probabilidad de mutación del individuo = 10 %, probabilidad de mutación de cada gen = $\frac{1}{\text{número de genes}}$ y elitismo. Para PSO se define $\phi_1 = 1.2$, $\phi_2 = 0.75$

- La metodología fue aplicada para clasificar dos tipos de cáncer: la leucemia linfocítica aguda y la leucemia mieloide aguda, datos obtenidos de la base de datos *Leukemia ALL – AML. Leukemia benchmark ALL – AML* contiene las medidas correspondientes a muestras de *ALL* y *AML* de médula ósea y sangre periférica. Originalmente se compone de 38 muestras para entrenamiento (27 *ALL* y 11 *AML*) y 34 muestras para prueba (20 *ALL* y 14 *AML*) donde cada muestra contiene información de 7129 expresiones génicas. Al realizar la partición se tienen 50 muestras en el conjunto de entrenamiento y 22 en el conjunto de validación. En el cuadro 6.1 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos = 30 para el GA, estos parámetros fueron establecidos de forma empírica.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.90	1.0	0.88	1.0	0.87	1.0
15	0.88	1.0	0.88	1.0	0.88	1.0
20	0.87	0.92	0.86	0.91	0.86	0.91
25	0.95	1.0	0.94	1.0	0.94	1.0
30	0.90	0.91	0.88	0.91	0.88	0.91

Cuadro 6.1: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.2 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.84	0.96	0.84	0.95	0.84	0.95
15	0.88	1.0	0.88	1.0	0.88	1.0
20	0.92	0.96	0.92	0.95	0.92	0.96
25	0.88	0.87	0.88	0.86	0.88	0.86
30	0.88	0.96	0.86	0.95	0.85	0.95

Cuadro 6.2: Ent.= Entrenamiento, Val.= Validación

El número de genes es un factor que interviene en el tiempo de cómputo, por lo que se repitió el experimento con una base que contiene solo el 10 % del total de los genes con la varianza más alta. En el cuadro 6.3 se muestran los resultados, definiendo el tamaño en el GA de la población= 50 y máximo número de ciclos= 50.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.89	0.93	0.88	0.91	0.88	0.91
15	0.88	0.96	0.88	0.95	0.88	0.95
20	0.90	0.96	0.90	0.95	0.90	0.95
25	0.96	0.90	0.96	0.86	0.96	0.86
30	0.90	0.96	0.90	0.95	0.90	0.95

Cuadro 6.3: Ent.= Entrenamiento, Val.= Validación

Se puede notar que al usar 15 genes se obtiene un buen clasificador, obteniendo precisión de hasta el 100 % en el conjunto de validación.

- La base de datos *colon Tumor* contiene 62 muestras, de las cuales 22 son positivas para Tumor en colon y 40 negativas. Al realizar la partición se tienen 43 muestras para el entrenamiento y 19 para validación. Cada muestra contiene 2000 genes. En el cuadro 6.4 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.87	0.96	0.86	0.95	0.86	0.95
10	0.79	0.91	0.79	0.89	0.78	0.89
15	0.88	0.95	0.88	0.95	0.88	0.94
20	0.81	1.0	0.81	1.0	0.81	1.0
30	0.79	0.95	0.79	0.95	0.79	0.95

Cuadro 6.4: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.5 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.84	1.0	0.84	1.0	0.83	1.0
10	0.93	0.95	0.93	0.95	0.93	0.95
15	0.84	1.0	0.84	1.0	0.84	1.0
20	0.87	1.0	0.86	1.0	0.85	1.0
30	0.79	1.0	0.79	1.0	0.79	1.0

Cuadro 6.5: Ent.= Entrenamiento, Val.= Validación

El experimento con una base que contiene solo el 10% del total, obtuvo resultados como los que se muestran en el cuadro 6.6 definiendo el tamaño en el GA de la población= 50 y máximo número de ciclos= 50.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.90	1.0	0.88	1.0	0.89	1.0
10	0.90	0.95	0.88	0.95	0.88	0.94
15	0.91	0.95	0.91	0.95	0.91	0.95
20	0.94	0.95	0.93	0.95	0.93	0.95
30	0.89	1.0	0.88	1.0	0.88	1.0

Cuadro 6.6: Ent.= Entrenamiento, Val.= Validación

Se puede apreciar que con 20 genes se obtiene un buen clasificador, que alcanza hasta el 100 % de precisión en el conjunto de validación.

- La base *Prostate cancer* se compone de 102 muestras para entrenamiento (52 con tumor de próstata y 50 con no-tumor de próstata "normal"). El conjunto de prueba contiene 25 muestras con tumor de próstata y 9 normal, cada muestra contiene 12,600 genes. Al realizar la partición se tienen 95 muestras para el entrenamiento y 41 para validación. En el cuadro 6.7 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.81	0.88	0.80	0.88	0.80	0.88
20	0.83	0.90	0.82	0.90	0.82	0.90
30	0.88	0.95	0.86	0.95	0.86	0.95
40	0.90	0.95	0.89	0.95	0.90	0.95
50	0.91	0.93	0.91	0.93	0.91	0.93

Cuadro 6.7: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.8 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.92	1.0	0.92	1.0	0.92	1.0
20	0.95	0.96	0.95	0.95	0.95	0.95
30	0.89	0.93	0.88	0.93	0.88	0.93
40	0.85	1.0	0.84	1.0	0.84	1.0
50	0.89	0.96	0.89	0.95	0.89	0.95

Cuadro 6.8: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.9 se muestran los resultados obtenidos usando solo el 10 % de la base original, definiendo el tamaño de la población= 50 y máximo número de ciclos= 50 en el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.88	0.96	0.87	0.95	0.87	0.95
20	0.91	0.93	0.89	0.93	0.90	0.93
30	0.91	0.98	0.91	0.98	0.91	0.98
40	0.93	1.0	0.93	1.0	0.93	1.0
50	0.97	0.92	0.97	0.90	0.97	0.90

Cuadro 6.9: Ent.= Entrenamiento, Val.= Validación

Se puede observar que usando 40 genes se alcanza una precisión de hasta el 100 % en la etapa de validación.

6.2. GA y GA

En la versión *GA y GA* se definieron los siguientes parámetros para ambos algoritmos: selección por torneo, probabilidad de cruza= 100 %, probabilidad de mutación del individuo = 10 %, probabilidad de mutación de cada gen= $\frac{1}{\text{número de genes}}$ y elitismo. Para el algoritmo genético 1 se define tamaño de la población = 30 y máximo número de ciclos= 30

- Base de datos *Leukemia ALL – AML*. En el cuadro 6.10 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.75	0.96	0.72	0.95	0.67	0.95
15	0.91	0.96	0.90	0.95	0.90	0.95
20	0.88	0.92	0.88	0.91	0.88	0.91
25	0.94	0.96	0.92	0.95	0.92	0.96
30	0.80	0.96	0.80	0.95	0.79	0.96

Cuadro 6.10: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.11 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.82	0.96	0.80	0.95	0.78	0.95
15	0.81	0.96	0.82	0.95	0.81	0.95
20	0.96	0.96	0.96	0.95	0.96	0.95
25	0.85	1.0	0.84	1.0	0.83	1.0
30	0.90	1.0	0.90	1.0	0.90	1.0

Cuadro 6.11: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.12 se muestran los resultados obtenidos usando solo el 10% de la base original, definiendo el tamaño de la población= 50 y máximo número de ciclos= 50 en el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.86	0.86	0.86	0.86	0.86	0.86
15	0.90	0.92	0.90	0.91	0.90	0.91
20	0.93	0.92	0.92	0.91	0.92	0.91
25	0.92	0.93	0.92	0.91	0.92	0.91
30	0.90	0.96	0.88	0.95	0.87	0.95

Cuadro 6.12: Ent.= Entrenamiento, Val.= Validación

Se aprecia que al usar 15 genes se obtiene precisión de hasta el 96% al realizar la validación.

- Base de datos *colon Tumor*. En el cuadro 6.13 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.86	0.85	0.84	0.84	0.84	0.84
10	0.89	0.95	0.88	0.95	0.88	0.95
15	0.82	0.95	0.81	0.95	0.81	0.94
20	0.88	1.0	0.86	1.0	0.86	1.0
30	0.96	0.91	0.95	0.89	0.95	0.89

Cuadro 6.13: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.14 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.80	0.79	0.79	0.79	0.79	0.79
10	0.81	0.91	0.81	0.89	0.81	0.89
15	0.82	1.0	0.81	1.0	0.81	1.0
20	0.89	0.95	0.88	0.95	0.88	0.95
30	0.89	0.95	0.88	0.95	0.88	0.95

Cuadro 6.14: Ent.= Entrenamiento, Val.= Validación

El experimento con una base que contiene solo el 10% del total, obtuvo resultados como los que se muestran en el cuadro 6.15 definiendo el tamaño en el GA de la población= 50 y máximo número de ciclos= 50.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.91	1.0	0.91	1.0	0.91	1.0
10	0.86	1.0	0.86	1.00	0.86	1.00
15	0.84	1.0	0.84	1.00	0.84	1.0
20	0.86	0.91	0.86	0.89	0.86	0.89
30	0.88	1.0	0.88	1.0	0.88	1.0

Cuadro 6.15: Ent.= Entrenamiento, Val.= Validación

Se puede advertir que al usar 10 genes se obtiene precisión de hasta el 100%

- Base *Prostate cancer*. En el cuadro 6.16 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.95	0.79	0.95	0.78	0.95	0.78
20	0.80	0.86	0.79	0.85	0.79	0.85
30	0.87	0.89	0.86	0.88	0.86	0.88
40	0.95	0.90	0.95	0.90	0.95	0.90
50	0.84	0.90	0.84	0.90	0.84	0.90

Cuadro 6.16: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.17 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA 2.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.83	0.87	0.83	0.85	0.83	0.85
20	0.87	0.93	0.87	0.93	0.87	0.93
30	0.94	0.93	0.94	0.93	0.94	0.93
40	0.81	0.88	0.81	0.88	0.81	0.88
50	0.83	0.91	0.81	0.90	0.81	0.90

Cuadro 6.17: Ent.= Entrenamiento, Val.= Validación

El experimento con una base que contiene solo el 10% del total, obtuvo resultados como los que se muestran en el cuadro 6.18 definiendo el tamaño en el GA de la población= 50 y máximo número de ciclos= 50.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.95	0.96	0.95	0.95	0.95	0.95
20	0.88	0.93	0.87	0.93	0.87	0.93
30	0.95	0.91	0.95	0.90	0.95	0.90
40	0.86	0.95	0.85	0.95	0.85	0.95
50	0.91	0.96	0.91	0.95	0.91	0.95

Cuadro 6.18: Ent.= Entrenamiento, Val.= Validación

Se observa que al usar 10 genes se obtiene precisión de hasta el 96 % en el conjunto de validación.

6.3. GA y DE

Los parámetros para el GA son: selección por torneo, probabilidad de cruce= 100 %, probabilidad de mutación del individuo = 10 %, probabilidad de mutación de cada gen= $\frac{1}{\text{número de genes}}$ y elitismo y para el DE son: $F = 0.8$, $CR = 0.9$, con una población de tamaño= 30 y un número máximo de iteraciones= 200

- Base de datos *Leukemia ALL – AML*. En el cuadro 6.19 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.84	0.91	0.84	0.91	0.84	0.91
15	1.0	0.96	1.0	0.95	1.0	0.96
20	0.92	0.96	0.92	0.95	0.92	0.96
25	0.92	1.0	0.92	1.0	0.92	1.0
30	0.95	1.0	0.94	1.0	0.94	1.0

Cuadro 6.19: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.20 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.87	1.0	0.86	1.0	0.85	1.0
15	0.94	0.96	0.94	0.95	0.94	0.95
20	0.98	1.0	0.98	1.0	0.98	1.0
25	0.96	0.93	0.96	0.91	0.96	0.91
30	1.0	0.96	1.0	0.95	1.0	0.95

Cuadro 6.20: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.21 se muestran los resultados obtenidos usando solo el 10% de la base original, definiendo el tamaño de la población= 50 y máximo número de ciclos= 50 en el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
3	0.74	0.96	0.74	0.95	0.74	0.96
15	0.90	0.96	0.90	0.95	0.90	0.96
20	0.98	0.96	0.98	0.95	0.98	0.95
25	0.98	1.0	0.98	1.0	0.98	1.0
30	0.98	1.0	0.98	1.0	0.98	1.0

Cuadro 6.21: Ent.= Entrenamiento, Val.= Validación

Se puede observar que al usar 20 genes se obtiene precisión de hasta el 100% en el conjunto de validación.

- Base de datos *colon Tumor*. En el cuadro 6.22 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.89	1.0	0.86	1.0	0.85	1.0
10	0.89	1.0	0.88	1.0	0.88	1.0
15	0.91	1.0	0.91	1.0	0.91	1.0
20	0.91	1.0	0.91	1.0	0.91	1.0
30	0.95	1.0	0.95	1.0	0.95	1.0

Cuadro 6.22: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.23 se muestran algunos resultados obtenidos definiendo

el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.84	0.96	0.84	0.95	0.83	0.95
10	0.91	1.0	0.91	1.0	0.91	1.0
15	0.96	1.0	0.95	1.0	0.95	1.0
20	0.91	1.0	0.91	1.0	0.91	1.0
30	0.91	1.0	0.91	1.0	0.91	1.0

Cuadro 6.23: Ent.= Entrenamiento, Val.= Validación

El experimento con una base que contiene solo el 10% del total, obtuvo resultados como los que se muestran en el cuadro 6.24 definiendo el tamaño en el GA de la población= 50 y máximo número de ciclos= 50.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
5	0.93	1.0	0.93	1.0	0.93	1.0
10	0.95	0.91	0.95	0.89	0.95	0.89
15	0.93	1.0	0.93	1.0	0.93	1.0
20	0.93	1.0	0.93	1.0	0.93	1.0
30	0.94	1.0	0.93	1.0	0.93	1.0

Cuadro 6.24: Ent.= Entrenamiento, Val.= Validación

Se puede notar que los resultados difieren en un porcentaje muy pequeño, cambiando principalmente en los resultados para el conjunto de entrenamiento.

- Base *Prostate cancer*. En el cuadro 6.25 se muestran algunos de los resultados obtenidos definiendo el tamaño de la población = 30 y máximo número de ciclos= 30 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.95	0.96	0.95	0.95	0.95	0.95
20	0.95	0.88	0.95	0.88	0.95	0.88
30	0.97	0.90	0.97	0.90	0.97	0.90
40	0.93	0.91	0.93	0.90	0.93	0.90
50	0.93	0.95	0.93	0.95	0.93	0.95

Cuadro 6.25: Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.26 se muestran algunos resultados obtenidos definiendo el tamaño de la población = 50 y máximo número de ciclos = 40 para el GA.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.98	0.91	0.98	0.90	0.98	0.90
20	0.95	0.96	0.95	0.95	0.95	0.95
30	0.91	0.95	0.91	0.95	0.91	0.95
40	0.89	0.96	0.87	0.95	0.87	0.95
50	0.94	0.98	0.93	0.98	0.93	0.98

Cuadro 6.26: Ent.= Entrenamiento, Val.= Validación

El experimento con una base que contiene solo el 10 % del total, obtuvo resultados como los que se muestran en el cuadro 6.27 definiendo el tamaño en el GA de la población= 50 y máximo número de ciclos= 50.

Genes	Precisión		Recall		f1-score	
	Ent.	Val.	Ent.	Val.	Ent.	Val.
10	0.96	0.95	0.96	0.95	0.96	0.95
20	0.93	1.0	0.93	1.0	0.93	1.0
30	0.92	1.0	0.92	1.0	0.92	1.0
40	0.92	0.98	0.92	0.98	0.92	0.98
50	0.97	0.98	0.97	0.98	0.97	0.98

Cuadro 6.27: Ent.= Entrenamiento, Val.= Validación

Se puede ver que al usar 20 genes se puede obtener precisión de hasta el 100 %

6.4. Resumen de los resultados

En el cuadro 6.28 se muestran los mejores resultados obtenidos al clasificar la base de datos *Leukemia ALL – AML*.

Alg. Bio.	Base	Genes	# It.	Precisión		Recall		f1-score	
				Ent.	Val.	Ent.	Val.	Ent.	Val.
PSO	ALL-AML	3	30	0.90	1.0	0.88	1.0	0.87	1.0
PSO	ALL-AML	15	50	0.88	1.0	0.88	1.0	0.88	1.0
GA	ALL-AML	15	30	0.91	0.96	0.90	0.95	0.90	0.95
GA	ALL-AML	25	50	0.85	1.0	0.84	1.0	0.83	1.0
DE	ALL-AML	15	30	1.0	0.96	1.0	0.95	1.0	0.96
DE	ALL-AML	3	50	0.87	1.0	0.86	1.0	0.85	1.0

Cuadro 6.28: Alg. Bio.= Algoritmo Bioinspirado, Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.29 se muestran los mejores resultados obtenidos al clasificar la base de datos *colon Tumor*.

Alg. Bio.	Base	Genes	# It.	Precisión		Recall		f1-score	
				Ent.	Val.	Ent.	Val.	Ent.	Val.
PSO	Colon Tumor	20	30	0.81	1.0	0.81	1.0	0.81	1.0
PSO	Colon Tumor	5	50	0.84	1.0	0.84	1.0	0.83	1.0
GA	Colon Tumor	10	30	0.89	0.95	0.88	0.95	0.88	0.95
GA	Colon Tumor	15	50	0.82	1.0	0.81	1.0	0.81	1.0
DE	Colon Tumor	5	30	0.89	1.0	0.86	1.0	0.85	1.0
DE	Colon Tumor	10	50	0.91	1.0	0.91	1.0	0.91	1.0

Cuadro 6.29: Alg. Bio.= Algoritmo Bioinspirado, Ent.= Entrenamiento, Val.= Validación

En el cuadro 6.30 se muestran los mejores resultados obtenidos al clasificar la base de datos *Prostate cancer*.

Alg. Bio.	Base	Genes	# It.	Precisión		Recall		f1-score	
				Ent.	Val.	Ent.	Val.	Ent.	Val.
PSO	Prostate cancer	30	30	0.88	0.95	0.86	0.95	0.86	0.95
PSO	Prostate cancer	10	50	0.92	1.0	0.92	1.0	0.92	1.0
GA	Prostate cancer	40	30	0.95	0.90	0.95	0.90	0.95	0.90
GA	Prostate cancer	20	50	0.87	0.93	0.87	0.93	0.87	0.93
DE	Prostate cancer	10	30	0.95	0.96	0.95	0.95	0.95	0.95
DE	Prostate cancer	20	50	0.95	0.96	0.95	0.95	0.95	0.95

Cuadro 6.30: Alg. Bio.= Algoritmo Bioinspirado, Ent.= Entrenamiento, Val.= Validación

En [11] se usó un MPL con 20 neuronas en la capa oculta para clasificar la base de datos *ALLAML* sin reducir la dimensionalidad de los datos, los resultados reportados usando *crossvalidation* son precisión: 0.97, *f1-score*: 0.98 y *accuracy*: 0.98, donde *accuracy* es el porcentaje de bien etiquetados. Comparando con los resultados obtenidos se aprecia comportamientos similares.

Yang [51] reporta para *prostate cancer* una reducción de la base a solo el 3.3% de genes, usando el sistema híbrido IG-GA/KNN y *crossvalidation* con un *accuracy* del 0.96. Contrastando los resultados se puede notar que con el sistema propuesto en este trabajo se obtiene una reducción a menos del 1% y precisión en el conjunto de validación entre el 0.90 y 1.0.

Jain en [25] reporta una reducción a 4 características de la base *ALLAML* con un *accuracy* de 1.0; la base de datos *colonTumor* se redujo a 4 genes con un *accuracy* de 0.95. Los resultados son semejantes a los obtenidos.

En [31] Luque presenta los siguientes resultados al utilizar un GA para seleccionar los genes y una red neuronal (C-MANTEC) para clasificar: *ALLAML*, 7 genes y *accuracy* de 0.99038; *ColonTumor*, 11 genes y *accuracy* de 0.94315; *Prostate cancer*, 8 genes y *accuracy* de 0.98681. Se puede notar que los resultados son consistentes con los obtenidos.

Después de contrastar los resultados se puede observar que los resultados expuestos en este trabajo son comparables con los reportados en la literatura.

Capítulo 7

Conclusiones

Los diversos experimentos permiten determinar el comportamiento de la metodología propuesta en la clasificación de microarreglos de ADN.

Durante la primera etapa, se aplicó con éxito una reducción de dimensionalidad sobre los conjuntos de datos *Leukemia benchmark ALL – AML*, *Colon Tumor* y *Prostate cancer* para seleccionar el conjunto de genes que mejor describe una enfermedad en particular, utilizando un algoritmo bioinspirado. El problema de reducción de dimensionalidad puede tratarse como un problema de optimización, debido a que la disminución dimensional de un microarreglo de ADN puede verse como un problema combinatorio que trata de encontrar entre miles de genes los más relevantes.

Los resultados obtenidos utilizaron, en la mayoría de los casos, menos del uno por ciento de los genes para realizar una tarea de detección o clasificación, coincidiendo con los resultados reportados en la literatura.

El conjunto de genes de interés en cada caso se encuentra en un espacio de búsqueda muy amplio. Por ejemplo para la base *colon Tumor* donde cada muestra contiene 2000 genes, al buscar subconjuntos de 5 genes se tiene un espacio de 265335665000400 soluciones; al buscar subconjuntos con 10 elementos el espacio es de $2.7589878594600562 \times 10^{26}$ lo cual es mayor que la edad del universo en segundos. Al no poder evaluar todas las posibles soluciones, hacer uso de los algoritmos bioinspirados es una opción viable.

En la segunda etapa, se evaluó el desempeño del GN. Los resultados obtenidos mostraron que todo el conjunto de datos se resolvió con una buena precisión, por lo que los algoritmos bioinspirados son técnicas adecuadas para entrenar un GN. Estos GN fueron entrenados usando el conjunto de genes propuestos en la primera etapa.

Se puede observar que el entrenamiento del GN se puede realizar utilizando

distintos algoritmos, obteniendo resultados similares.

El número de pesos usados para entrenar el modelo es reducido gracias a que se disminuye la dimensionalidad de los microarreglos y a la estructura compacta del GN, por lo que permite acotar la complejidad numérica del problema manteniendo un buen desempeño.

El GN entrenado con la metodología propuesta es capaz de detectar, predecir y clasificar una enfermedad con una precisión aceptable. Comparando los resultados obtenidos con los descritos en [11], podemos ver que son equiparables y el GN tiene un menor número de pesos a entrenar. Los resultados alcanzados al pre-procesar las bases de datos reduciéndolas al 10% con la varianza mas alta y despues usar un GA para reducirlas nuevamente son muy similares a los obtenidos usando las bases completas con el GA.

Comparando los resultados obtenidos al entrenar con los tres algoritmos bioinspirados y subconjuntos de la base *Leukemia ALLAML* se puede concluir que un subconjunto adecuado deberá contener entre 15 y 20 genes. Un subconjunto adecuado para la base *colon Tumor*, deberá contener entre 30 y 40 genes y para la base *Prostate cancer* el subconjunto deberá contener entre 10 y 15 genes. Los subconjuntos de índices obtenidos son todos distintos entre sí, aún cuando el porcentaje de precisión es muy similar, lo que podría sugerir que existen múltiples óptimos.

Los experimentos confirman que los algoritmos bioinspirados dependen en gran medida de la inicialización de la población y del número de posibles soluciones que evalúe.

Usando la base de datos *ALLAML* los mejores resultados son los obtenidos al usar DE para entrenar el neurón generalizado.

Con la base de datos *Colon tumor* el algoritmo con mejor desempeño para entrenar el neurón es DE.

PSO es el algoritmo con mejores resultados al entrenar el neurón generalizado usando la base *Prostate cancer*.

Siglas

ADN ácido desoxirribonucleico. 1

ADNc ácido desoxirribonucleico complementario. 6

ANN redes neuronales artificiales. 1

ARNm ácido ribonucleico mensajero. 6

DE evolución diferencial. 8

GA algoritmos genéticos. 8

GN neurón generalizado. 1

KNN kvecinos cercanos. 8

LDA análisis discriminante lineal. 8

MPL perceptrón multicapa. 1, 8

PSO algoritmo basado en cúmulos de partículas. 8

SFS selección secuencial hacia adelante. 8

SVM máquina de soporte vectorial. 8

Bibliografía

- [1] ALBELDA, S. M., AND SHEPPARD, D. Functional genomics and expression profiling: be there or be square. *American journal of respiratory cell and molecular biology* 23, 3 (2000), 265–269.
- [2] ANGENENDT, P., GLÖKLER, J., SOBEK, J., LEHRACH, H., AND CAHILL, D. J. Next generation of protein microarray support materials:: Evaluation for protein and antibody microarray applications. *Journal of chromatography A* 1009, 1-2 (2003), 97–104.
- [3] ANTONISSE, J. A new interpretation of schema notation that overturns the binary encoding constraint. In *Proceedings of the 3rd international conference on genetic algorithms* (1989), Morgan Kaufmann Publishers Inc., pp. 86–91.
- [4] ARRANZ DE LA PEÑA, J., AND PARRA TRUYOL, A. Algoritmos genéticos. *Recuperado el 20* (2007), 06–07.
- [5] BLICKLE, T., AND THIELE, L. A comparison of selection schemes used in evolutionary algorithms. *Evolutionary Computation* 4, 4 (1996), 361–394.
- [6] BRAIK, M., SHETA, A., AND ARIEQAT, A. A comparison between gas and pso in training ann to model the te chemical process reactor. In *AISB 2008 Convention communication, interaction and social intelligence* (2008), vol. 1, p. 24.
- [7] BUSQUETS, X., AND AGUSTÍ, A. Chip genético (adn array): el futuro ya está aquí. *Archivos de Bronconeumología* 37, 9 (2001), 394–396.
- [8] CANO GUTIÉRREZ, C. Extracción de conocimiento de microarrays y literatura biomédica para el estudio de la regulación genética [tesis doctoral]. *Granada: Editorial de la Universidad de Granada* (2010).
- [9] CHATURVEDI, D. K., MOHAN, M., SINGH, R. K., AND KALRA, P. K. Improved generalized neuron model for short-term load forecasting. *Soft Computing* 8, 1 (2003), 10–18.

- [10] DAVIS, L. Applying adaptive algorithms to epistatic domains. In *IJCAI* (1985), vol. 85, pp. 162–164.
- [11] DWIVEDI, A. K. Artificial neural network model for effective cancer classification using microarray gene expression data. *Neural Computing and Applications* 29, 12 (2018), 1545–1554.
- [12] EBERHART, R., AND KENNEDY, J. Particle swarm optimization, proceeding of ieee international conference on neural network. *Perth, Australia* (1995), 1942–1948.
- [13] EBERHART, R. C., SHI, Y., AND KENNEDY, J. *Swarm intelligence*. Elsevier, 2001.
- [14] ENGELBRECHT, A. P. *Fundamentals of computational swarm intelligence*. John Wiley & Sons, 2006.
- [15] ESPITIA CUCHANGO, H. E., AND SOFRONY ESMERAL, J. I. Algoritmo de optimización basado en enjambres de partículas con comportamiento de vorticidad y búsqueda individual y grupal. *Tecnura* 18, 42 (2014).
- [16] GARRO, B. A., RODRÍGUEZ, K., AND VÁZQUEZ, R. A. Classification of dna microarrays using artificial neural networks and abc algorithm. *Applied Soft Computing* 38 (2016), 548–560.
- [17] GLÖKLER, J., AND ANGENENDT, P. Protein and antibody microarray technology. *Journal of Chromatography B* 797, 1-2 (2003), 229–240.
- [18] GOLDBERG, D. E. Genetic algorithms in search, optimization and machine learning, 1989.
- [19] GOLDBERG, D. E. Real-coded genetic algorithms, virtual alphabets, and blocking. *Complex systems* 5, 2 (1991), 139–167.
- [20] GOLDBERG, D. E., AND DEB, K. An investigation of niche and species formation in genetic function optimization. In *Proceedings of the third international conference on Genetic algorithms* (1989), pp. 42–50.
- [21] HIDALGO, J. I., AND DE VEGA, F. F. Distribución equilibrada del esfuerzo de cómputo en algoritmos genéticos.
- [22] HOLLAND, J. H. Adaptation in natural and artificial systems. an introductory analysis with application to biology, control, and artificial intelligence. *Ann Arbor, MI: University of Michigan Press* (1975), 439–444.

- [23] HUYNH, H. T., KIM, J.-J., AND WON, Y. Classification study on dna micro array with feed forward neural network trained by singular value decomposition. *International Journal of Bio-Science and Bio-Technology 1* (2009), 17–24.
- [24] ILLANA-RICO, E. Análisis bioinformático de datos de expresión genética obtenidos mediante tecnología de microarrays.
- [25] JAIN, I., JAIN, V. K., AND JAIN, R. Correlation feature selection based improved-binary particle swarm optimization for gene selection and cancer classification. *Applied Soft Computing 62* (2018), 203–215.
- [26] KENNEDY, J. The particle swarm: social adaptation of knowledge. In *Evolutionary Computation, 1997., IEEE International Conference on* (1997), IEEE, pp. 303–308.
- [27] KENNEDY, J. Particle swarm optimization. In *Encyclopedia of machine learning*. Springer, 2011, pp. 760–766.
- [28] KIRAN, R., JETTI, S. R., AND VENAYAGAMOORTHY, G. K. Online training of a generalized neuron with particle swarm optimization. In *Neural Networks, 2006. IJCNN'06. International Joint Conference on* (2006), IEEE, pp. 5088–5095.
- [29] KULKARNI, R. V., AND VENAYAGAMOORTHY, G. K. Generalized neuron: Feedforward and recurrent architectures. *Neural Networks 22* (2009), 1011–1017.
- [30] LEZA, M. V. *Optimización mediante cúmulos de partículas con tamaño de población variable*. PhD thesis, Facultad de Informática, 2008.
- [31] LUQUE-BAENA, R. M., URDA, D., SUBIRATS, J. L., FRANCO, L., AND JEREZ, J. M. Application of genetic algorithms and constructive neural networks for the analysis of microarray cancer data. *Theoretical Biology and Medical Modelling 11*, 1 (2014), S7.
- [32] MELIÁN, B., PÉREZ, J. A. M., AND VEGA, J. M. M. Metaheurísticas: Una visión global. *Inteligencia Artificial. Revista Iberoamericana de Inteligencia Artificial 7*, 19 (2003), 0.
- [33] MEZURA-MONTES, E., VELÁZQUEZ-REYES, J., AND COELLO COELLO, C. A. A comparative study of differential evolution variants for global optimization. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation* (2006), ACM, pp. 485–492.

- [34] MICHALEWICZ, Z. Evolution strategies and other methods. In *Genetic algorithms+ data structures= evolution programs*. Springer, 1996, pp. 159–177.
- [35] MONTAÑO MORENO, J. J., ET AL. Redes neuronales artificiales aplicadas al análisis de datos.
- [36] NIETO, J. G., AND POLO, G. J. L. Algoritmos basados en cúmulos de partículas para la resolución de problemas complejos. *Septiembre de* (2006).
- [37] OCAMPO, E. M. T., ECHEVERRY, M. G., AND ROMERO, R. Algoritmo memético aplicado a la solución del problema de asignación generalizada. *Tecnura* 8, 16 (2005), 55–63.
- [38] PLOUS, C. V. Microarreglos de adn y sus aplicaciones en investigaciones biomédicas. *Revista CENIC. Ciencias Biológicas*, 2 (2007), 132–135.
- [39] PRICE, K., STORN, R. M., AND LAMPINEN, J. A. *Differential evolution: a practical approach to global optimization*. Springer Science & Business Media, 2006.
- [40] QUINTERO, M. A. Reconocimiento de rostros con visión por computadora y optimización basada en cúmulos de partículas. *CULCyT: Cultura Científica y Tecnológica*, 40 (2010), 69–79.
- [41] RAMÍREZ, J., CHÁVEZ, L., SANTILLÁN, J. L., AND GUZMÁN, S. Microarreglos de dna. *Mensaje Bioquímico* 27 (2003), 97–120.
- [42] REEVES, C. R. *Modern heuristic techniques for combinatorial problems*. John Wiley & Sons, Inc., 1993.
- [43] RIZWAN, M., JAMIL, M., AND KOTHARI, D. Generalized neural network approach for global solar energy estimation in india. *IEEE Transactions on Sustainable Energy* 3 (2012), 576–584.
- [44] ROMAN REYES, H. *SOLUCION DE PROBLEMAS DE OPTIMIZACION MEDIANTE ALGORITMOS GENETICOS APLICANDO COMPUTO DE ALTO RENDIMIENTO*. PhD thesis.
- [45] STORN, R. P. kenneth p, differential evolution—a simple and efficient adaptive scheme for global optimization over continuous spaces. Tech. rep., Technical Report TR-95-012, ICSI, 1995.
- [46] STORN, R., AND PRICE, K. Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

- [47] VAN DEN BERGH, F., AND ENGELBRECHT, A. P. A study of particle swarm optimization particle trajectories. *Information sciences* 176, 8 (2006), 937–971.
- [48] WETZEL, A. Evaluation of the effectiveness of genetic algorithms in combinatorial optimization.
- [49] WHITLEY, L. D., ET AL. The genitor algorithm and selection pressure: Why rank-based allocation of reproductive trials is best. In *ICGA* (1989), vol. 89, Fairfax, VA, pp. 116–123.
- [50] WOLPERT, D. H., AND MACREADY, W. G. No free lunch theorems for optimization. *IEEE transactions on evolutionary computation* 1, 1 (1997), 67–82.
- [51] YANG, C.-H., CHUANG, L.-Y., YANG, C. H., ET AL. Ig-ga: a hybrid filter/wrapper method for feature selection of microarray data. *Journal of Medical and Biological Engineering* 30, 1 (2010), 23–28.