

Segmentación de Imágenes Médicas mediante un algoritmo genético generacional

Adriana Montserrat García Carrillo and Juan René Hernández Sánchez

¹ Centro de investigación en computación del Instituto Politécnico Nacional, CDMX, México.

² Springer Heidelberg, Tiergartenstr. 17, 69121 Heidelberg, Germany
lncs@springer.com

Abstract. A los pacientes con síntomas de un tumor cerebral se les prescribe realizarse una resonancia magnética con el fin de tener un diagnóstico adecuado. El tumor cerebral es una afección médica muy grave caracterizada por la proliferación de células anormales en el tejido cerebral, por lo tanto, es necesario un análisis rápido y correcto de la presencia o ausencia de tumores a partir de imágenes. El principal objetivo de esta investigación es la correcta segmentación de imágenes en tres distintas regiones: cerebro, fondo y tumor, a través de métodos heurísticos.

Keywords: Algoritmo Genético, Métodos Heurísticos, Segmentación de Imágenes.

1 Introducción

1.1 Métodos Heurísticos

Existen diferentes tipos de tumores cerebrales. Algunos tumores pueden ser benignos, es decir, no son cancerígenos y otros pueden ser cancerígenos, es decir, malignos. Dicho lo anterior, un tumor cerebral es una masa o crecimiento de células anormales en el cerebro. Los síntomas relacionados a este pueden incluir: dolores de cabeza, pérdida del equilibrio, visión borrosa, convulsiones, confusión, entre otros, pero en algunos casos se puede presentar la posibilidad de que no existan síntomas. Algunos de los tratamientos para los tumores cerebrales pueden ser: cirugía, radiación, quimioterapia y otros más.

La gravedad del tumor cerebral lo ha convertido en la principal causa de mortalidad relacionada con el cáncer, por lo tanto, confiar sólo en la inspección humana para una detección correcta puede llevar a una situación muy peligrosa. Por ese motivo, es importante mejorar y acelerar el proceso de diagnóstico manual mediante sistemas automatizados de diagnóstico.

Por lo tanto, es muy importante la segmentación de regiones en las resonancias magnéticas con el fin de separar lo que es cerebro, tumor y fondo. Este tema ha sido un campo de gran interés y ha llevado a la realización de diversos estudios. Durante las primeras investigaciones relacionadas con la segmentación de imágenes, la gran mayoría de los científicos utilizaban métodos estándar de procesamiento de imágenes,

como el método basado en umbrales y el método basado en regiones. En esta investigación se abordará el uso de una técnica de segmentación basada en métodos heurísticos, concretamente en algoritmos genéticos, y se describirá su implementación, ventajas y resultados.

Los procedimientos heurísticos son una clase de métodos aproximados que están diseñados para resolver problemas complejos de optimización. Las metaheurísticas proporcionan un marco general para crear nuevos algoritmos híbridos combinando diferentes conceptos derivados de la inteligencia artificial, la evolución biológica y los mecanismos estadísticos.

1.2 Algoritmos Genéticos

Los algoritmos genéticos (AG) son algoritmos inspirados en la selección natural que han logrado semejar todo el proceso evolutivo de los seres vivos para resolver problemas de optimización, búsqueda y aprendizaje en las máquinas.

La teoría de los algoritmos genéticos fue desarrollada por John H. Holland, en la Universidad de Michigan, en un proyecto que buscaba explicar el proceso de adaptación natural de los sistemas y diseñar sistemas artificiales que conservaran los más importantes mecanismos de los sistemas naturales. Casi todos los conceptos de los algoritmos genéticos están basados en conceptos de biología y genética.

Las características de los seres vivos se encuentran registradas en los cromosomas y los genes, lo que hace que cada individuo sea diferente dentro de una población. En algoritmos genéticos los cromosomas se representan mediante cadenas (strings) y cada posición de la cadena representa un gen.

El primer paso desarrollar un algoritmo genético es codificar el parámetro deseado como una cadena y usar un alfabeto. La longitud de un cromosoma está determinada por el número de genes que lo conforman.

En los algoritmos genéticos, el conjunto ordenado de genes de un cromosoma se conoce como genotipo y el fenotipo corresponde a la decodificación del cromosoma. Cada cromosoma es una solución, buena o mala, del problema. A medida que el algoritmo genético se va desarrollando, las soluciones se acercan cada vez más a la óptima, simulando los sistemas de selección natural a los que se encuentran sometidos los organismos vivos. Los AG utilizan tres operadores básicos para solucionar el problema: Selección, Cruzamiento y Mutación.

En la presente investigación se hará uso de los AG generacionales, este modelo consiste en que durante cada iteración se crea una población completa con nuevos individuos. La nueva población reemplaza directamente a la antigua.

2 Estado del Arte

Para comenzar con el estado del arte se mencionarán investigaciones relacionadas a la segmentación y clasificación de tumores cerebrales. A continuación, se describen algunos trabajos de investigación relevantes en los últimos años:

Suzuki [1] utilizó un algoritmo de umbral iterativo para la segmentación. Este método ofrece resultados muy prometedores, pero cuando el contraste de la imagen es bajo resulta difícil seleccionar el umbral.

Por otra parte, Salman [2] demostró que el crecimiento de una región es un método eficaz de segmentación de tumores cerebrales. En contraste con otros métodos no basados en regiones, la cantidad de cálculo es menor, especialmente para tejidos y regiones homogéneas. Aunque los dos métodos mencionados anteriormente son fáciles de implementar y no requieren una gran cantidad de cálculo, los resultados de la precisión de la segmentación no cumplen con las expectativas prácticas, ya que se espera que devuelvan una segmentación acertada entre tumor, cerebro y fondo, pero siguen existiendo ambigüedades.

De igual manera, muchos investigadores han propuesto la segmentación de tumores cerebrales basándose en métodos de clasificación o agrupamiento. Fletcher-Heath [3] propuso un algoritmo de agrupamiento difuso no supervisado, que combina el conocimiento del dominio y la tecnología de procesamiento de imágenes para lograr la segmentación del tumor.

Zhou [4] utilizó un método basado en una máquina de soporte vectorial de una sola clase para extraer tumores cerebrales a partir de imágenes de resonancia magnética. Subbanna [5] planteó un marco probabilístico jerárquico totalmente automatizado para segmentar el tumor cerebral basado en filtros de Gabor de múltiples ventanas y un marco de Markov Random Field adaptado. Estos métodos pueden mejorar la precisión, pero todavía se necesitan técnicas con mejores resultados para la práctica clínica.

Años más tarde, Alpaydin [6] mencionó la importancia del Aprendizaje Automático y cómo se ha aplicado gradualmente en el análisis de imágenes médicas.

3 Desarrollo de la Solución

3.1 Descripción del problema planteado

Se quieren segmentar de forma automática imágenes de cerebros para saber si en ellos hay o no tumores. Para ello, se cuenta con un conjunto de imágenes, y se define el siguiente problema de optimización:

Se tiene una función a optimizar F , que recibe tres parámetros:

- s : es un vector solución de tamaño 3 (tumor, cerebro y fondo) donde $s_i \in [0..255]$, que representa el valor de gris de cada región
- d : es una función de desempeño (delegado) que recibe un parámetro: s
- \max : es un valor binario que dice si la función d es de maximizar (verdadero) o minimizar (falso)

Encuentre, mediante algoritmos metaheurísticos, el vector que resuelva el problema de optimización, para la siguiente función de desempeño:

$$d = \frac{\text{Mínima distancia entre pixeles de diferente grupo}}{\text{Máxima distancia entre pixeles del mismo grupo}} \quad (1)$$

La función d calcula la dispersión entre los pixeles de grupos diferentes y pixeles del mismo grupo. Cada píxel se asigna a su grupo más cercano, y los empates se resuelven aleatoriamente. Es por este motivo que se busca maximizar esta función. La distancia D puede ser la distancia de Minkowski de orden $p=1$. La distancia de Minkowski para orden 1 se reduce a una diferencia entre los valores de los pixeles entre 0 y 255.

3.2 Modelado

L Con el fin de encontrar una solución al problema planteado se propone el siguiente modelado basado en algoritmos genéticos para buscar resolver el problema de optimización:

- **Estado inicial:** un vector s de tamaño 3 donde $s_i \in [0, 255]$.
- **Estado final:** un vector s' tal que $d(s') = \{ \max_{1 \leq i \leq N} s^i \}$, $s^i \in S$, S siendo el conjunto de todos los vectores analizados y N es el cardinal de S , $N \in \mathbb{Z}^+$.

- **Test objetivo:**

$$d(s) = \frac{\text{Mínima distancia entre pixeles de diferente grupo}}{\text{Máxima distancia entre pixeles del mismo grupo}}$$

$\text{Mínima distancia entre pixeles de diferente grupo} = \min_{i,j \in I} (|s_a^i - s_b^j|^p)^{\frac{1}{p}}$, donde $a, b \in [1,3]$, $a \neq b$, donde $1 \leq i, j \leq I$, tal que $p = 1$.

$\text{Máxima distancia entre pixeles del mismo grupo} = \max_{i,j \in I} (|s_a^i - s_a^j|^p)^{\frac{1}{p}}$, donde $a \in [1,3]$, donde $1 \leq i, j \leq I$, $i \neq j$, tal que $p = 1$.

- **Acciones posibles:**

$f: s \rightarrow s'$ tal que $\exists s_i \neq s'_i, i \in [1,3]$

3.3 Descripción de la solución al problema planteado

Se propuso un algoritmo genético generacional con la siguiente configuración y los siguientes parámetros:

- Configuración:
 - Selección por torneo
 - Cruzamiento por un punto de corte
 - Mutación por cambio de bits

- Parámetros:

- Tamaño de la imagen = 5 x 5
- Tamaño de la población = 10
- Iteraciones = 20
- Tamaño de la población del torneo = 2
- Probabilidad de cruzamiento = 0.6
- Tasa de mutación = 0.8

- **Operador de selección por torneo**

Para implementar la selección de torneos, sólo se necesita elegir k individuos al azar con reemplazo y comparar los valores de aptitud de esos individuos; dicha operación es el torneo. El individuo que posea la mejor aptitud gana el torneo y se selecciona en el grupo de apareamiento. El parámetro k se denomina tamaño del torneo y controla la presión selectiva. La selección de torneos más utilizada es la binaria, en la que $k = 2$.

- **Cruzamiento por un punto de corte**

Para dos individuos que son seleccionados con el fin de realizar el cruzamiento, se asignará de forma aleatoria un punto de corte entre 1 y $l-1$, donde l es la longitud del cromosoma. Esto significa que se generará un entero aleatorio en el rango $[1, l-1]$. Posteriormente, los genes después del punto de corte serán cambiados entre los padres y los cromosomas resultantes serán los descendientes (ver Fig 1). Los descendientes también tendrán cromosomas de longitud l .

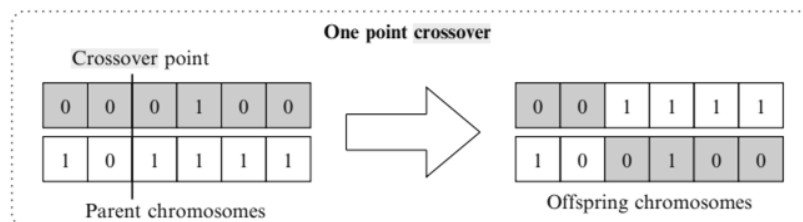


Fig. 1. Operador de Cruzamiento por un punto de corte.

- **Mutación por cambio de bits**

Cada gen de un individuo se muta con una probabilidad p_m , llamada tasa de mutación. Siempre que deba mutar el gen j , se hace un cambio de bit, es decir, se cambia de 1 a 0 o de 0 a 1. A este operador se le llama mutación de cambio de bits (ver Fig 2).

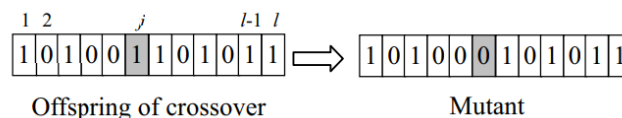


Fig. 2. Operador de mutación.

La configuración propuesta fue seleccionada debido a ciertas ventajas que se observaron respecto a otros operadores. Se eligió la selección por torneo debido a que no posee ruido estocástico, además de que es eficiente, funciona en arquitecturas paralelas y permite que la presión de selección sea fácilmente ajustable. De igual manera, se optó por el cruzamiento de un punto de corte debido a que representa una mejora, cuando se añaden más puntos de cruce el comportamiento del algoritmo se ve afectado. La tasa de mutación es alta con el fin de que exista mayor variabilidad genética.

De igual manera, el tamaño población al no ser tan pequeño permite tener mayor diversidad. Al unir estos tres parámetros (operador de cruzamiento de un punto de corte, tasa de mutación y tamaño de población) aseguramos una presión selectiva alta.

Por otra parte, se puede observar que el algoritmo posee distintos parámetros que influyen en su comportamiento, así que deben ser ajustados. El procedimiento para realizar dicho ajuste consistió en tomar el parámetro a configurar y fijar los demás parámetros. Después, se eligió el valor del parámetro que devolviera los mejores resultados en el desempeño del algoritmo, esto se realizó con todos los parámetros.

Debido a que en el algoritmo interviene el azar, una sola ejecución no es suficiente para saber si un parámetro es adecuado. Por lo tanto, se decidió ejecutar el algoritmo 500 veces por cada ajuste en cualquier parámetro.

3.4 Implementación

```
import random

tam_cromosoma = 3
tam_poblacion = 10
k_torneo = 2 #tamaño de la población del torneo
humbral = 0.8
prob_cruzamiento = 0.6
iteraciones = 50
tam_imagen = 5

imagen = [[random.randint(0, 255) for _ in
            range(tam_imagen)]
           for _ in range(tam_imagen)]

class Gen:
    cromosoma = []
    aptitud = float
    generacion = int

    def __init__(self, generacion, cromosoma):
        self.cromosoma = cromosoma
        self.calcular_aptitud()
```

```

        self.generacion = generacion

#*****

#Función que calcula la aptitud
def calcular_aptitud(self):
    parametros = [[], [], []]
    for i in imagen:
        for j in i:
            dist_pixel = []
            dist_pixel.append(minkowsky(j,
self.cromosoma[0]))
            dist_pixel.append(minkowsky(j,
self.cromosoma[1]))
            dist_pixel.append(minkowsky(j,
self.cromosoma[2]))

            segmentacion =
dist_pixel.index(min(dist_pixel))

            if segmentacion == 0:
                parametros[0].append(j)
            elif segmentacion == 1:
                parametros[1].append(j)
            elif segmentacion == 2:
                parametros[2].append(j)

    dist_min = [[], [], []]

    for i in range(3):
        aux = []
        if i == 0:
            aux.extend(parametros[1])
            aux.extend(parametros[2])
        elif i == 1:
            aux.extend(parametros[0])
            aux.extend(parametros[2])
        elif i == 2:
            aux.extend(parametros[0])
            aux.extend(parametros[1])
        for j in range(len(parametros[i])):
            aux2 = [minkowsky(parametros[i][j], x)
for x in aux]
            dist_min[i].extend(aux2)

```

```

dist_max = [[0], [0], [0]]

for i in range(3):
    for j in range(len(parametros[i])):
        for k in range(j+1, len(parametros[i])):
            dist_max[i].append(
                minkowsky(parametros[i][j], par-
ametros[i][k]))

    for i in dist_min:
        if len(i) == 0:
            i.append(0)

    self.aptitud = min([min(e) for e in dist_min]) /
\
    max([max(u) for u in dist_max])

#*****

def mutacion(self):
    for i in range(tam_cromosoma):
        if random.random() < humbral:
            self.cromosoma[i] = random.randint(0,
255)

def __str__(self):
    return "< Crom: " + str(self.cromosoma) + " Apt:
" + str(round(self.aptitud, 3)) + " Gen: " +
str(self.generacion) + ">"

def generar_cromosoma():
    cromosoma = [random.randint(0, 255) for _ in
range(tam_cromosoma)]

    return cromosoma

def minkowsky(x, y):
    return abs(x-y)

def seleccion_torneo(poblacion):
    torneo = []
    for i in range(k_torneo):
        seleccion = random.randint(0, tam_poblacion - 1)
        gen = poblacion[seleccion]

```



```

        torneo.append(gen)
    return max(torneo, key=lambda x: x.aptitud)

def one_point_crossover(gen1, gen2):
    descendienteA = []
    descendienteB = []

    punto_corte = random.randint(1, tam_cromosoma-1)

    descendienteA = gen1.cromosoma[:punto_corte] +
    gen2.cromosoma[punto_corte:]
    descendienteB = gen2.cromosoma[:punto_corte] +
    gen1.cromosoma[punto_corte:]

    return descendienteA, descendienteB

def replacement(poblacion, nueva_poblacion):
    poblacion.clear()
    poblacion.extend(nueva_poblacion)
    nueva_poblacion.clear()

#*****
poblacion = []
nueva_poblacion = []
generacion_global = 1

# Generacion Inicial aleatoria
for _ in range(tam_poblacion):
    poblacion.append(Gen(generacion_global, generar_cromosoma()))

# Iteracion para 500 generaciones
while generacion_global < iteraciones:
    # Generando nueva poblacion
    generacion_global += 1
    while len(nueva_poblacion) != tam_poblacion:

        if random.random() > probab_cruzamiento:
            continue

        gen1 = seleccion_torneo(poblacion)
        gen2 = seleccion_torneo(poblacion)
        # Cruzamiento de los padres
        crom_des1, crom_des2 = one_point_crossover(gen1,
gen2)

```

```

descendiente1 = Gen(generacion_global, crom_des1)
descendiente2 = Gen(generacion_global, crom_des2)
# Mutacion del descendiente
descendiente1.mutacion()
descendiente2.mutacion()

descendiente1.calcular_aptitud()
descendiente2.calcular_aptitud()

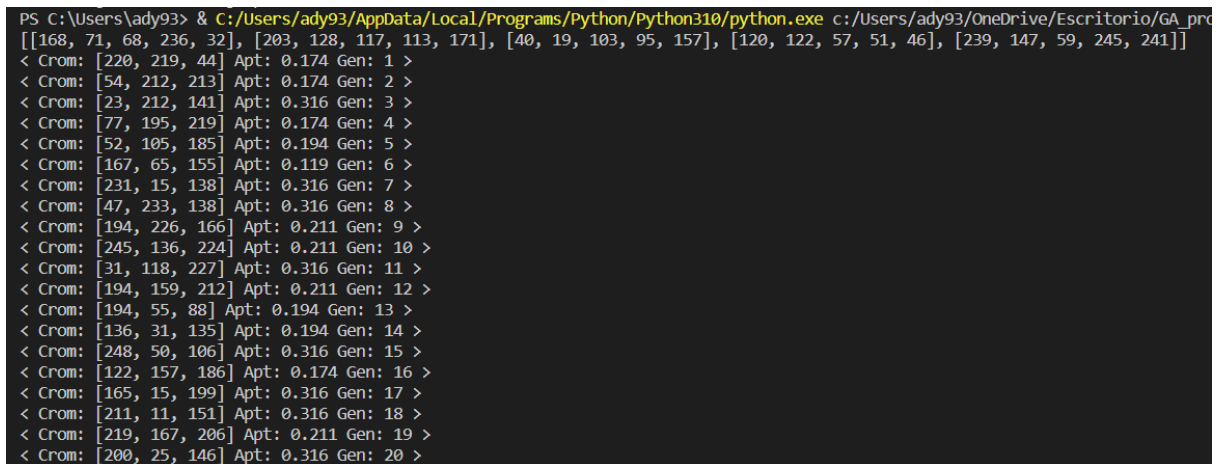
# Insertando el descendiente en la nueva poblacion
nueva_poblacion.append(descendiente1)
nueva_poblacion.append(descendiente2)

replacement(poblacion, nueva_poblacion)
print(max(poblacion, key=lambda x: x.aptitud))

```

3.5 Resultados

A continuación, se muestran los resultados obtenidos por el programa (ver Fig 3), en donde se especifica la aptitud calculada en cada una de las 20 generaciones. De igual manera, se presenta un gráfico para visualizar con mayor detalle el comportamiento del AG generacional propuesto.



```

PS C:\Users\ady93> & C:/Users/ady93/AppData/Local/Programs/Python/Python310/python.exe c:/Users/ady93/OneDrive/Escritorio/GA_prd
[[168, 71, 68, 236, 32], [203, 128, 117, 113, 171], [40, 19, 103, 95, 157], [120, 122, 57, 51, 46], [239, 147, 59, 245, 241]]
< Crom: [220, 219, 44] Apt: 0.174 Gen: 1 >
< Crom: [54, 212, 213] Apt: 0.174 Gen: 2 >
< Crom: [23, 212, 141] Apt: 0.316 Gen: 3 >
< Crom: [77, 195, 219] Apt: 0.174 Gen: 4 >
< Crom: [52, 105, 185] Apt: 0.194 Gen: 5 >
< Crom: [167, 65, 155] Apt: 0.119 Gen: 6 >
< Crom: [231, 15, 138] Apt: 0.316 Gen: 7 >
< Crom: [47, 233, 138] Apt: 0.316 Gen: 8 >
< Crom: [194, 226, 166] Apt: 0.211 Gen: 9 >
< Crom: [245, 136, 224] Apt: 0.211 Gen: 10 >
< Crom: [31, 118, 227] Apt: 0.316 Gen: 11 >
< Crom: [194, 159, 212] Apt: 0.211 Gen: 12 >
< Crom: [194, 55, 88] Apt: 0.194 Gen: 13 >
< Crom: [136, 31, 135] Apt: 0.194 Gen: 14 >
< Crom: [248, 50, 106] Apt: 0.316 Gen: 15 >
< Crom: [122, 157, 186] Apt: 0.174 Gen: 16 >
< Crom: [165, 15, 199] Apt: 0.316 Gen: 17 >
< Crom: [211, 11, 151] Apt: 0.316 Gen: 18 >
< Crom: [219, 167, 206] Apt: 0.211 Gen: 19 >
< Crom: [200, 25, 146] Apt: 0.316 Gen: 20 >

```

Fig. 3. Vista de los mejores vectores solución de cada generación.

La gráfica muestra la gran variedad de soluciones obtenidas por el GA, permite visualizar los mejores vectores de segmentación y los que no ofrecen una buena solución. Al ser generacional tiene un comportamiento errático, pero nos asegura obtener distintas posibles soluciones, lo que nos permite explorar varios óptimos locales, es decir, poder experimentar con distintas buenas soluciones (ver Fig 4).

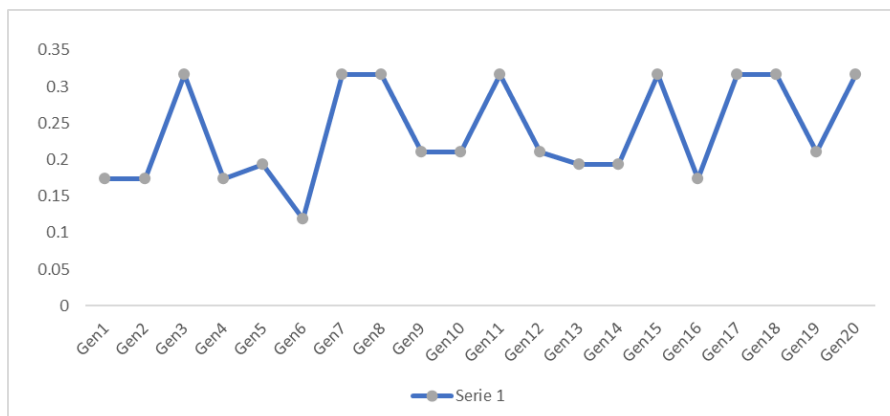


Fig. 4. Tabla del comportamiento del GA generacional.

4 Conclusiones y trabajo a futuro

Los AG no aseguran encontrar la solución óptima, pero en muchos casos proporcionan soluciones con un alto grado de acierto y, que por otros métodos hubiera sido imposible encontrar. Además, son paralelizables, ya que distribuyen la carga computacional en varios procesadores y logran acelerar la ejecución general del algoritmo, es decir, realizan una estimación de la función de evaluación menos precisa, pero a un menor costo computacional.

Como hemos podido observar, la principal ventaja de los algoritmos genéticos radica en su sencillez y tal vez el punto más complicado sea la definición de la función de evaluación, por tal motivo, se eligió a estos algoritmos como la base teórica para la resolución del problema de segmentación de imágenes (en cerebro, tumor y fondo), ya que logran un balance entre la eficacia y la eficiencia, es decir, proporcionan una buena relación entre la calidad de la solución y el costo.

La ventaja fundamental del algoritmo propuesto en esta investigación es que su principal objetivo es la optimización de los vectores de segmentación, a diferencia de otros métodos que buscan realizar la mejor segmentación por sí misma. Por el contrario, el algoritmo propuesto siempre asegura encontrar varios óptimo locales, dando la posibilidad de poder segmentar imágenes que son difíciles para otros algoritmos.

Inicialmente, se entrena sólo con imágenes que poseen tumores, pero posteriormente al encontrar el vector de solución óptimo, tendrá la capacidad de segmentar y distinguir imágenes que no poseen tumor de manera robusta.

Referencias

1. Suzuki H, Toriwaki J. Automatic segmentation of head MRI images by knowledge guided thresholding. *Comput Med Imaging Graph*. 1991;15:233–40.

2. Salman YM, Assal MA, Badawi AM, Alian SM, El-Bayome ME-. Validation techniques for quantitative brain tumors measurements. In: 2005 IEEE Engineering in Medicine and Biology 27th Annual Conference, 2005;pp. 7048–7051.
3. Fletcher-Heath LM, Hall LO, Goldgof DB, Murtagh FR. Automatic segmentation of non-enhancing brain tumors in magnetic resonance images. *Artif Intell Med.* 2001;21:43–63.
4. Zhou J, Chan K, Chong V, Krishnan SM. Extraction of brain tumor from MR images using one-class support vector machine. In: 2005 IEEE engineering in medicine and biology 27th annual conference; 2006. p. 6411–14.
5. Subbanna NK, Precup D, Collins DL, Arbel T. Hierarchical probabilistic GABOR and MRF segmentation of brain tumors in MRI volumes. In: International conference on medical image computing and computer-assisted intervention; 2013. p. 751–8.
6. Alpaydin E. Introduction to machine learning. London: MIT Press; 2020.