Algoritmos Genéticos: Optimización de Hiperparámetros de una Red Neuronal Convolucional

---

Asignatura: Modelos Bioinspirados y Heurísticas de Búsqueda

Alumno: Alberto Fernández Merchán

ETSI ESCUELA TÉCNICA SUPERIOR de INGENIERÍA uhu.es

Universidad de Huelva

Índice

Introducción

La **clasificación de imágenes** es una tarea importante en el campo de la visión por computador, trata áreas de aplicación tales como detección, localización y segmentación de instancias.

Para realizar dichas tareas se utilizan, normalmente, **redes neuronales convolucionales** (CNN) que demuestran resultados incluso mejores que el de los humanos. Sin embargo, una CNN está compuesta de muchos hiper parámetros (número de capas convolucionales, número de filtros, tamaños…).

En este trabajo se consigue aprender un conjunto de hiper parámetros óptimos para la red utilizando un algoritmo genético basado en la ***propagación de la élite***.

Primera Aproximación

Realizada a **mano**

30.8% accuracy

Valores de los hiperparámetros **no son óptimos**

Layer	Layer Name	Layer Properties
1	Image Input	227x227x3 images with 'zerocenter' normalization
2	Convolution	96 11x11x3 convolutions with stride [4 4] and padding [0 0 0 0]
3	ReLU	ReLU
4	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
5	Convolution	256 5x5x96 convolutions with stride [1 1] and padding [2 2 2 2]
6	ReLU	ReLU
7	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
8	Convolution	384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]
9	ReLU	ReLU
10	Convolution	384 3x3x384 convolutions with stride [1 1] and padding [1 1 1 1]
11	ReLU	ReLU
12	Convolution	256 3x3x384 convolutions with stride [1 1] and padding [1 1 1 1]
13	ReLU	ReLU
14	Max Pooling	3x3 max pooling with stride [2 2] and padding [0 0 0 0]
15	Fully Connected	4096 fully connected layer
16	ReLU	ReLU
17	Fully Connected	4096 fully connected layer
18	ReLU	ReLU
19	Fully Connected	256 fully connected layer
20	Softmax	softmax
21	Classification Output	crossentropyex

Estructura del Algoritmo Genético

- Codificación del cromosoma
- Función Objetivo (*Fitness*)
- Mecanismo de Selección
- Cruce y Mutación
- Evolución

Estructura del Algoritmo Genético

Codificación del cromosoma ⇐ Cada cromosoma = Solución del problema

Función Objetivo (*Fitness*)

Mecanismo de Selección

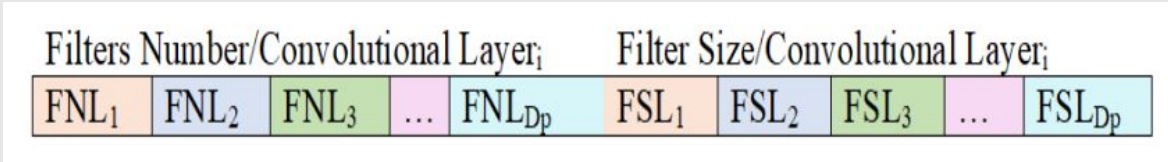
Cruce y Mutación

Evolución

El tamaño del cromosoma será de $2*D_p$ donde:
D_p : número de capas de convolución

La posición FNL_i simboliza el **número de filtros** por cada capa.

La posición FSL_i simboliza el **tamaño del filtro**.



Inicialización de la población

Valores **Aleatorios** Uniformes

Tamaño del Cromosoma: 5 Capas Convolucionales → **10 genes**

Tamaño de la población: **8 individuos**

Estructura del Algoritmo Genético

- Codificación del cromosoma
- **Función Objetivo (*Fitness*)**
- Mecanismo de Selección
- Cruce y Mutación
- Evolución

Minimizar el error de clasificación de la CNN:

$$score = 100 - accuracy$$

$$accuracy = \frac{numero\_de\_instancias\_bien\_clasificadas}{size(DSTest)}$$

Estructura del Algoritmo Genético

Codificación del cromosoma

Función Objetivo (*Fitness*)

Mecanismo de Selección

Cruce y Mutación

Evolución

Se aplica al comienzo de cada generación

Elitista (20%)

Para elegir a los padres se utiliza una **ruleta proporcional**

Se crea un 40% de los hijos por **crossover**

Estructura del Algoritmo Genético

- Codificación del cromosoma
- Función Objetivo (*Fitness*)
- Mecanismo de Selección
- **Cruce y Mutación**
- Evolución

Cruce:

- Selecciona a los 2 padres
- Genera un vector binario aleatorio
- Los genes del primer padre se heredan si en el vector anterior hay un 1, por el contrario, se heredan los del segundo padre.

Mutación:

- Selecciona una fracción de los genes del hijo ($p_r = 0.01$)
- Cada gen se reemplaza por un número aleatorio dentro del rango correspondiente

Estructura del Algoritmo Genético

Codificación del cromosoma

Función Objetivo (*Fitness*)

Mecanismo de Selección

Cruce y Mutación

Evolución

Criterio de parada: Iteraciones fijas (5)

Evaluación de la calidad del mejor individuo.

Se **reinicia** si es necesario

Pseudocódigos

Algorithm 1: E-CNN-MP main algorithm

Input: D, convolutional layers number (NumConLayers), Max generations number (MaxG), Generation size N
Output: Best individual Fit*I*, Fit*CNN*
Loading images dataset D
Initialization of the training dataset (TrainingDS), the test dataset (TestDS), Best Accuracy, Fraction of elites (f_e), Fraction of crossover created children(f_c)
Randomly (Uniform distribution) create an initial population P of N chromosomes.
for g **in** 1 **to** MaxG
do
[S_1,S_2 ,..,S_N, BestAccuracy, Fit*I*, Fit*CNN*]← **FitnessCNN** (P, NumConLayers, BestAccuracy, TrainingDS, TestDS);
P'← **Recombination** (P , f_e, f_c);
P←P';
end for;
return Fit*I*;
save Fit*CNN*;

Pseudocódigos

Algorithm 2: FitnessCNN function

Input: Population of **N** chromosomes, NumConLayers, BestAccuracy, TrainingDS, TestDS
Output: Vector of chromosome scores S_i, BestAccuracy, Fit*I*, Fit*CNN*

```
/*CNN model generation*/
Layers←[imageInputLayer([227 227 3]);
for C in 1 to N
for i in 1 to length(C)/2
FN←C[i];
FS←C[i+length(C)/2];
Layers←concatenate(layers,convolution2dLayer(FS,FN,'padding',FS/2),reluLayer, maxPooling2dLayer(2,'Stride',2));
end for;
layers←concatenate(layers, fullyConnectedLayer, reluLayer, fullyConnectedLayer, reluLayer, fullyConnectedLayer(),
softmaxLayer ,classificationLayer());
/*From scratch CNN training*/
CNN←training(TrainingDS, layers, trainingoptions);
TestLabel ← classify(CNN, TestDS);
ImageLabel ← TestDS.Labels;
accuracy ← 100*sum(TestLabel==ImageLabel)/size(TestDS);
scores(C) ← 100-accuracy;
if accuracy > BestAccuracy
then
        BestAccuracy←accuracy;
        save(C);
        save(CNN);
 end if;
end for;
return C;
return CNN;
```

Pseudocódigos

Algorithm 3: Successor generations creation algorithm

Input: Population of **N** Chromosomes P, f_e, f_c
Output: Population of **N** Chromosomes P'
/* Elite individual propagation*/
E←N* f_e;
Select E Elite chromosomes from P and place them in P';
/*Create N' child chromosomes by crossover method*/
N'←N* f_c;
For i 1 to N'
do
initialize a child vector C;
select 2 parents P1 and P2 from P based on roulette method;
randomly generate a binary vector R with length(R)=length(P1);
for j in 1 to length(R)
  do
    if R[j]==1
      then
        C[j]←P1[j]; //choose the first parent gene to be conserved in the child
      else
        C[j]←P2[j]; //choose the second parent gene to be conserved in the child
    end if;
end for;
/*Mutation operation*/
randomly select child gene position i to be mutated:
 i← random (1,length (C));
 randomly select x from the child gene initial range;
C(i)← x;
end for;
/*Randomly generation of remaining individuals in P'*/
N''← N-(E+N');
Inject randomly N'' chromosomes in P' (To guaranty diversity);
Return P';

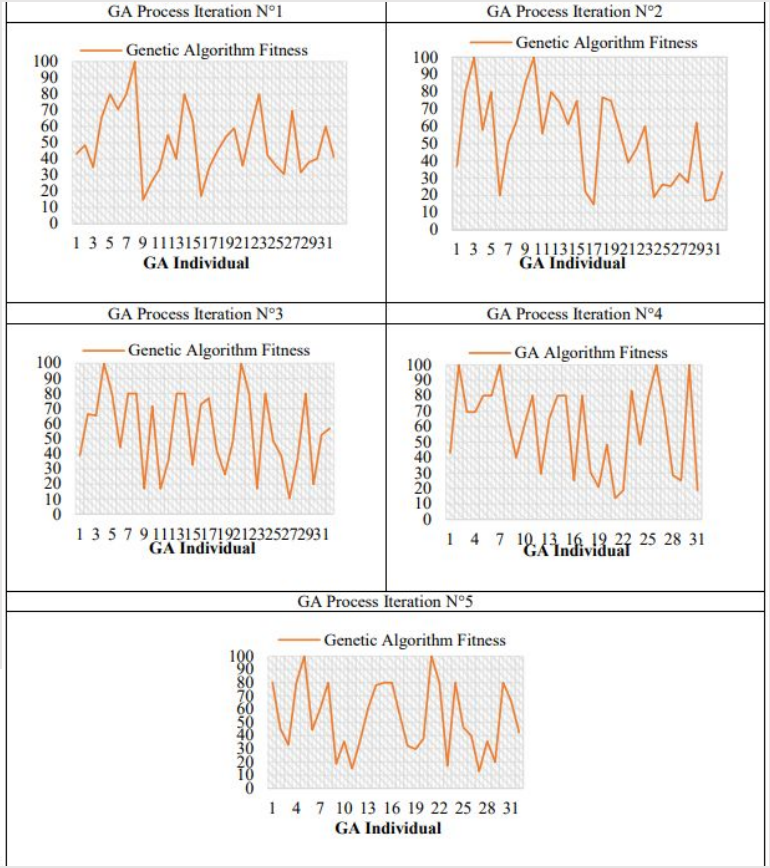
Hiperparámetros del Algoritmo Genético

GA Settings	Value
Number of convolutional layers	5
Number of GA variables to optimized	10
GA population size	8
Individual score	Classification Error
GA stopping criteria is the maximum number of generations	4
Elite individuals fraction	20%
Fraction of children created by crossover	40%

Resultados (I)

Este enfoque ofrece un mejor resultado que cuando se hizo la CNN a mano. Sin embargo, podemos ver como el *score* oscila demasiado entre las generaciones. Esto provoca que la media no supere el 50%.

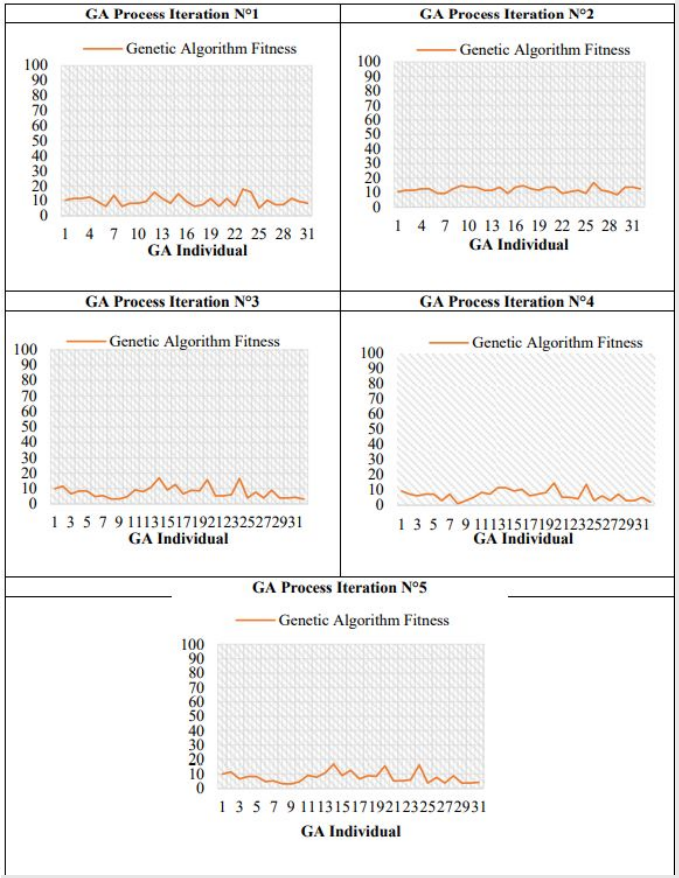
Para mejorar el resultado lo que se hace es utilizar una normalización por lotes (***batch-normalization***). Estas capas se colocan entre las convolucionales y las ReLU.



GA process	Max Accu.%	Min Accu.%	Average Accu. %	Best Network Encoding
1	85.27	0	49.81	73 49 61 96 73 18 7 8 7 16
2	85.26	0	47.73	42 75 57 48 76 2 5 11 17 16
3	89.47	0	44.55	80 80 55 70 66 14 4 2 18 2
4	86.32	0	42.11	70 70 60 77 58 17 7 3 6 9
5	87.02	0	46.85	60 73 52 76 61 11 9 3 4 2

Resultados (II)

Con esta pequeña mejora, la red puede alcanzar ahora hasta un **98.94%** de precisión.



GA process	Max Accu.%	Min Accu.%	Average Accu.%	Best Network Encoding
1	94.74	82.11	89.9	57 98 94 98 38 8 9 7 3 7
2	91.57	83.15	87.89	31 84 50 81 86 13 3 12 20 6
3	94.93	80	87.79	57 98 94 98 37 8 9 7 3 7
4	98.94	85.26	93.25	59 37 81 79 41 19 4 6 17 5
5	96.97	83.11	92.37	60 73 52 76 61 11 9 3 4 2

Estructura de la CNN generada

Layer	Layer Name	Layer Properties
1	Image Input	227x227x3 images with 'zerocenter' normalization
2	Convolution	59 19x19x3 convolutions with stride [1 1] and padding [9 9 9 9]
3	Batch Normalization	Batch normalization with 59 channels
4	ReLU	ReLU
5	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
6	Convolution	37 4x4x59 convolutions with stride [1 1] and padding [2 2 2 2]
7	Batch Normalization	Batch normalization with 37 channels
8	ReLU	ReLU
9	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
10	Convolution	81 6x6x37 convolutions with stride [1 1] and padding [3 3 3 3]
11	Batch Normalization	Batch normalization with 81 channels
12	ReLU	ReLU
13	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
14	Convolution	79 17x17x81 convolutions with stride [1 1] and padding [8 8 8 8]
15	Batch Normalization	Batch normalization with 79 channels
16	ReLU	ReLU
17	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
18	Convolution	41 5x5x79 convolutions with stride [1 1] and padding [2 2 2 2]
19	Batch Normalization	Batch normalization with 41 channels
20	ReLU	ReLU
21	Max Pooling	2x2 max pooling with stride [2 2] and padding [0 0 0 0]
22	Fully Connected	256 fully connected layer
23	Softmax	softmax
24	Classification Output	crossentropyex with 5 classes

Bibliografía

Loussaief, S., & Abdelkrim, A. (2018). Convolutional neural network hyper-parameters optimization based on genetic algorithms. *International Journal of Advanced Computer Science and Applications*, *9*(10).