Implementación de un Algoritmo Genético como optimizador en un Modelo Lineal bajo Aprendizaje Supervisado

Adán Mora-Fallas - Tecnológico de Costa Rica - ajmorafallas@gmail.com
Iván Calvo Pérez - Tecnológico de Costa Rica - ivanfelipecp@gmail.com

Resumen—Hay distintas formas de implementar un modelo lineal para un algoritmo de clasificación bajo aprendizaje supervisado, en este proyecto se evaluó la viabilidad de implementar un Algoritmo Genético para optimizar el modelo y lograr la mejor eficiencia posible. Se obtuvo que esta implementación es dependiente de la dominio del problema, pues entre más cerrados sean los rangos de los valores, mejor se irán ajustando los datos aleatorios a los datos de prueba, este caso se probó con el set de datos Iris y se obtuvo una eficiencia máxima de 96 %; en cambio, si el rango de números aleatorios es grande, costará más que se obtenga un modelo eficiente, quedó probado con el set de datos CIFAR-10, donde el rango de números va de 0 a 255 y se obtuvo una eficiencia máxima de 34 %.

Index Terms—IA, Algoritmo Genético, Modelo Lineal, Machine Learning

1. Introducción

Actualmente, Machine Learning es una disciplina en constante innovación e investigación para encontrar más mejores métodos para hacer que las máquinas aprendan a clasificar.

En el domino del problema planteado, se utilizará un modelo lineal para que aprenda a reconocer imágenes de 4 clases distintas y flores a partir de características propias de cada una. Estos modelos son muy utilizados en este tipo de problemas, y el proceso de aprendizaje se hará mediante la implementación de un algoritmo genético.

Por lo tanto, sabiendo que esto implica que no se puede obtener un resultado eficiente, el objetivo es experimentar en el proceso genético del algoritmo para tratar de optimizar el clasificador, por ejemplo, tamaño de población y la forma de hacer cruces.

2. Trabajo relacionado

Kalaivani y Shunmuganathan [4] implementaron en 2014 un clasificador KNN (*K Nearest Neighboirs*) con Algoritmos genéticos para un set de datos de *reviews* de libros y películas, y contrastaron esa implementación con un clasificador KNN común, los resultados demostraron que el Algoritmo Genético fue más eficiente que el ya ineficiente KNN común. El cruce de dicho algoritmo consiste en combinar mitad y mitad de cada padre, y la mutación se realiza según una probabilidad de que suceda (0.01).

Hilda y Rajalaxmi [3] hicieron un trabajo parecido, implementaron un Algoritmo Genético para KNN, pero le agregaron paralelismo, realizaron pruebas con 5 distintos set de datos, como el de cáncer de pecho recopilado por la Universidad de Wisconsin y dio una eficiencia de más de un 99 %, también resaltan que la implementación de un clasificador con Algoritmos Genéticos es buena con pocas características, por ejemplo en el caso del set de datos de Wisconsin, cada dato posee solo 10 características.

Yann Carbonne y Christelle Jacob [5] hicieron un trabajo el cuál emplea aprendizaje supervisado y algoritmos genéticos como aprendizaje para determinar que tan parecidos eran dos perfiles de diferentes contextos(redes sociales, foros, etc...), usando procesamiento del lenguaje natural y modelos de espacio vectorial para representar la información de los perfiles como nombres, apellidos, ciudad, etc. Ellos concluyen que la solución propuesta en el paper es adaptable y genérica ya que el modelo no está restringido a un conjunto fijo de etiquetas. Siempre y cuando las etiquetas están presentes en el conjunto de entrenamiento, tanto el modelo matemático y el algoritmo genético logra adaptarse a un nuevo conjunto de etiquetas.

3. Metodología

Para nuestro algoritmo de clasificación, utilizaremos un modelo lineal dado por la forma:

$$f(X, W) = W.X + b$$

Donde W y b son parámetros aprendidos por el modelo para clasificar los elementos de X. Mediante el *Bias Trick*, se junta la W y la b:

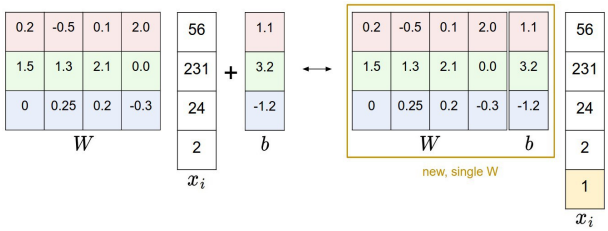

Figura 1. *Bias Trick*

La W (junto con el *Bias Trick*) es el parámetro que clasifica la imagen y determina qué tan bueno es el modelo lineal. Para efectos de este proyecto, se implementó un Algoritmo Genético para generar y optimizar el W según las mejores características de cada individuo generado. Sin embargo, se decidió eliminar la b del modelo porque no era significante para el problema.

Los Algoritmos Genéticos basan su metodología en la Selección Natural. En síntesis, primero se da un conjunto inicial de individuos con ciertas características aleatorias, mediante una función de adaptabilidad se determina cuáles son la mejores características y se les asigna un valor según el criterio de supervivencia, los resultados se ordenan del más apto al menos apto, y se empieza a hacer cruces entre los individuos siguiendo algún criterio para obtener otra generación (normalmente se cruzan solo dos individuos, pero se puede experimentar a criterio propio), no hay poligamia, es decir, una vez cruzados, ya no pueden cruzarse más; también puede intervenir un proceso de mutación, esto puede derivar en mantener ciertos genes entre generaciones, o generar genes con características predefinidas.

El modelo limita la cantidad de generaciones que puede generar mediante un hiperparámetro.

3.1. Criterios del Algoritmo Genético implementado

3.1.1. Población inicial

La cantidad de individuos que se genere será un hiperparámetro del modelo, la forma en como se genere depende del set de datos, para Iris se generan vectores con números entre 0 y 10, pues, los valores de cada dato oscilan entre ese rango, y para CIFAR-10, se generan vectores con números entre 0 y 255, pues son pixeles en escala de grises.

3.1.2. Cruce

Para realizar los cruces entre individuos, primero hay que ordenar el conjunto de $W's$ del más apto al menos apto, luego se separa siempre el 50 % como los más aptos para cruzar, hay un hiperparámetro que indica el porcentaje de individuos menos aptos a cruzar para mantener la variabilidad. Se cruzan todos los individuos que actualmente conforman la población, los $W's$ de la pasada generación vuelven a cruzarse pero no necesariamente con el mismo W de la generación pasada.

Los cruces son simples, cada fila del W resultante posee la mitad de la fila correspondiente de cada W padre, a menos que se cumpla el criterio de mutación, por ejemplo:

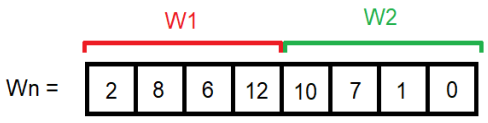

Figura 2. *Ejemplo básico de cruce.*

El resultado de hacer el cruce entre W_1 y W_2 es W_n, donde el vector resultante corresponde a la mitad de los padres, W_1 y W_2.

3.1.3. Mutación

El criterio para que haya una mutación se basa en la eficiencia que obtuvo el W con cada clase del set de datos, y se maneja mediante un hiperparámetro que me indica el porcentaje de eficiencia que debe tener una clase para que mute.

Para el modelo, mutar significa que va a mantener una fila (cada fila representa una clase) en el nuevo W, si la eficiencia de esta es igual o superior a la puesta en el hiperparámetro.

3.1.4. Fitness

Para determinar qué tan bueno es un W, se utiliza una función de pérdida (*Loss function*), en este caso, se utilizó la función Hinge Loss vista previamente en clase, esta función tiene la forma:

$$s = f(X, W)$$

$$L = 1/N \sum_{i}^{N} L_i$$

$$L_i = \sum_{j \neq y_i}^{N} max(0, s_j - s_{y_i} + 1)$$

Donde s es el resultado de multiplicar el W con X (conjunto de datos de prueba).[1]

3.2. Sets de datos utilizados

Este modelo lineal aprenderá a partir de dos set de datos comúnmente usados en Machine Learning, Iris, un conjunto de datos sobre tres tipos de flores sobre sus medidas de pétalo y sépalo, en total son 150 datos, 50 para cada clase; y CIFAR-10, un conjunto de datos de 10 clases de imágenes de 32x32 pixeles que consta de un set de datos de entrenamiento con 5000 imágenes para cada clase, y un set de datos de prueba con 1000 imágenes para cada clase. Sin embargo, solo se utilizaran 4 clases de este set de datos.

Con CIFAR-10 tenemos un problema de dimensionalidad, pues, para cada imagen hay que analizar en total 32x32x3 (3072) elementos, por lo que se hace realmente costoso en tiempo y procesamiento usar este ser de datos, por lo que, se procedió a convertir las imágenes de color a escala de grises usando la siguiente fórmula:

$$Y = 0{,}299R + 0{,}587G + 0{,}114B$$

Donde Y es el pixel resultante, y R, G y B representa la intensidad del color rojo, verde y azul respectivamente en el pixel original. [2]

4. EXPERIMENTOS

Se realizó un total de 3 experimentos para cada set de datos y con diferentes hiperparámetros, para poder tener distintos escenarios y comprobar qué tan buena es esta implementación de un modelo lineal para clasificación.

Con el fin de mostrar la variabilidad que produce la generación $W's$ con números random, cada prueba sea realizó 5 veces, y así contrastar esos resultados.

Además, se mostrarán gráficas que representan cómo cambia el Loss y la eficiencia obtenida en cada W seleccionada como más apta para el modelo lineal, después de n generaciones.

4.1. Iris

Para Iris, cada W se genera con números flotantes random, particularmente para este set de datos, se generaron números random con dos rangos distintos, entre 0 y 10, y entre el valor mínimo y el máximo de cada característica del set de datos, por lo que, las tres pruebas se realizarán para cada rango.

Se utilizaron los siguientes hiperparámetros para cada prueba:

	P_1	P_2	P_3
Población inicial(n)	100	1000	5000
Cantidad generaciones(n)	20	10	5
Cruce con menos aptos(%)	20	10	5
Mínimo aceptación(%)	90	85	95
Porcentaje de mutación(%)	80	90	90

4.2. CIFAR-10

Se utilizaron los siguientes hiperparámetros para cada prueba:

	P_1	P_2	P_3
Población inicial(n)	100	500	1000
Cantidad generaciones(n)	10	5	5
Cruce con menos aptos(%)	10	20	30
Mínimo aceptación(%)	50	50	55
Porcentaje de mutación(%)	80	90	95

5. RESULTADOS

Los resultados están organizados por set de datos, por cada prueba realizada y con una tabla con los resultados para cada iteración (I_i) realizada y contienen la eficiencia para cada clase del set de datos (C_i). Después uno de los gráficos e imágenes más relevante de cada prueba.

5.1. Iris

5.1.1. Prueba 1

	I_1	I_2	I_3	I_4
Eficiencia total(%)	92	92.6	92	92.6
Loss total(n)	0.22	0.40	0.17	0.25
Generaciones(n)	14	9	13	13
Eficiencia C_0 (%)	100	98	100	100
Eficiencia C_1 (%)	80	94	86	98
Eficiencia C_2 (%)	96	86	90	80

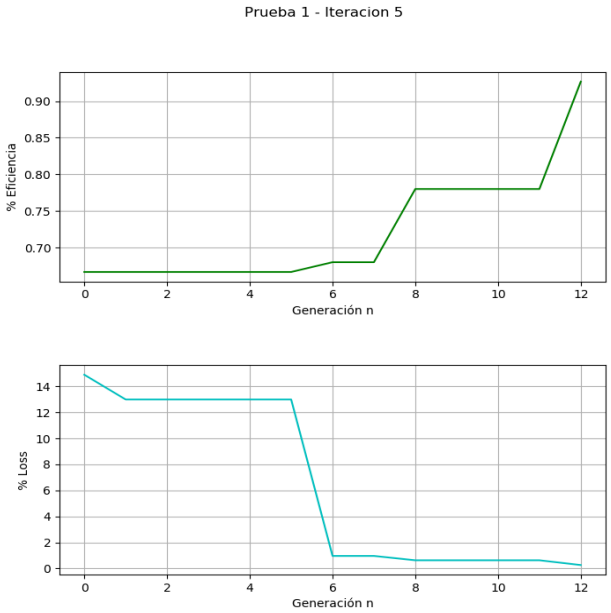

Figura 3. *Gráficos del porcentaje de Eficiencia y Loss por generación. Prueba 1, Iteración 4 de Iris.*

5.1.2. Prueba 2

	I_1	I_2	I_3	I_4
Eficiencia total(%)	89.3	89.3	91.3	90.6
Loss total(%)	0.30	0.32	0.24	0.20
Generaciones(n)	5	6	7	3
Eficiencia C_0 (%)	100	100	100	100
Eficiencia C_1 (%)	90	74	98	94
Eficiencia C_2 (%)	78	94	76	78

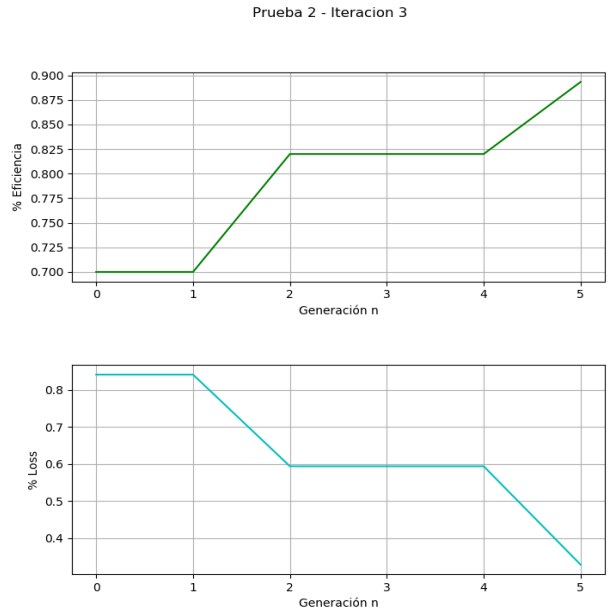

Figura 4. *Gráficos del porcentaje de Eficiencia y Loss por generación. Prueba 2, Iteración 2 de Iris.*

5.1.3. *Prueba 3*

	I_1	I_2	I_3	I_4
Eficiencia total(%)	92.6	96	96	92
Loss total(%)	0.14	0.13	0.10	0.53
Generaciones(n)	5	3	2	5
Eficiencia C_0 (%)	100	100	100	98
Eficiencia C_1 (%)	86	90	94	84
Eficiencia C_2 (%)	92	98	94	94

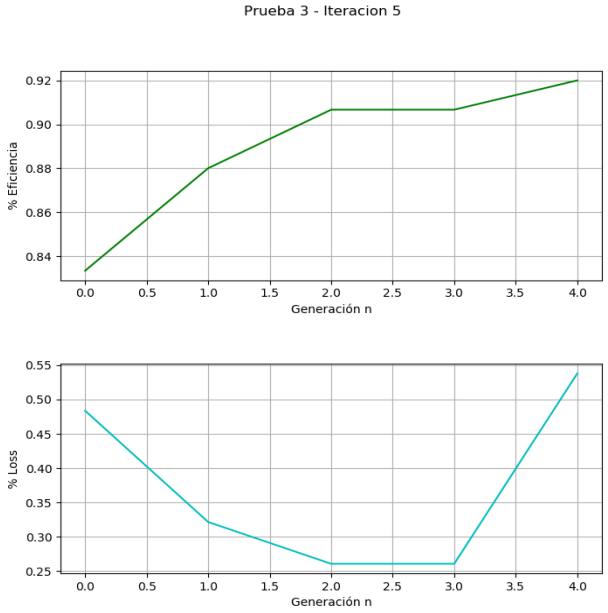

Figura 5. *Gráficos del porcentaje de Eficiencia y Loss por generación. Prueba 3, Iteración 4 de Iris.*

Adicionalmente, se realizó una prueba extra con los hiperparámetros de la prueba 1, pero con una eficiencia mínima de 99 %, es decir, llegar al máximo que puede converger el modelo con un límite de 20 poblaciones a partir de 100 individuos inicialmente.

Se obtuvo el siguiente gráfico:

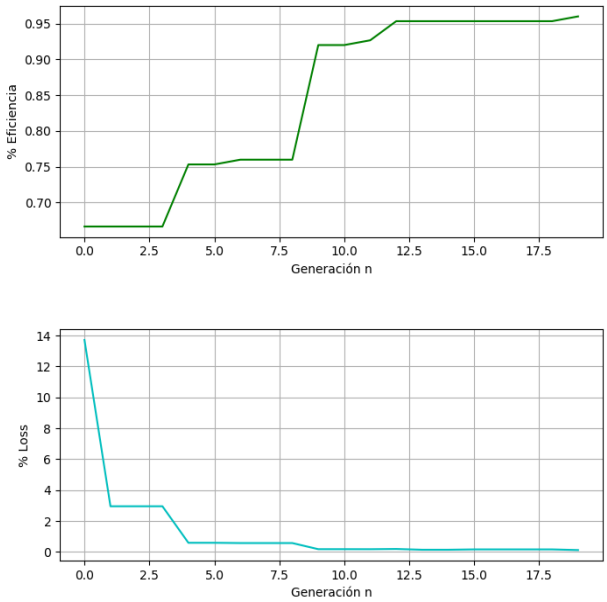

Figura 6. *Prueba con 20 generaciones, máximo de 96 % de eficiencia.*

5.2. CIFAR-10

5.2.1. *Prueba 1*

	I_1	I_2	I_3	I_4
Eficiencia total(%)	32.6	32.05	33.05	31.87
Loss total(%)	229321	198935	351577	469104
Generaciones(n)	5	5	5	5
Eficiencia C_0 (%)	0.8	72.3	0.4	0.4
Eficiencia C_1 (%)	51.3	30.2	52.7	53.5
Eficiencia C_2 (%)	22.9	9.5	75.8	0.4
Eficiencia C_3 (%)	55.6	16.2	3.3	73.2

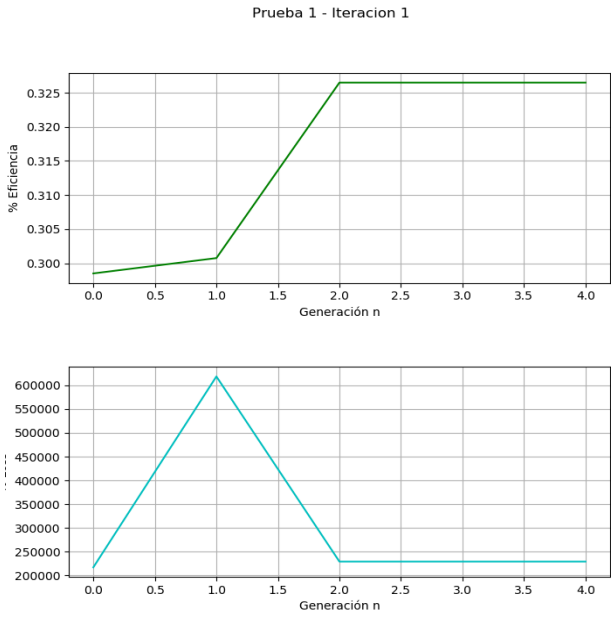

Figura 7. *Gráficos del porcentaje de Eficiencia y Loss por generación. Prueba 1, Iteración 1 de CIFAR-10.*

5.2.2. Prueba 2

	I_1	I_2	I_3	I_4
Eficiencia total(%)	32.75	33.67	33.9	32.27
Loss total(%)	704622	216076	220777	837480
Generaciones(n)	5	5	5	5
Eficiencia C_0 (%)	0	57.3	6.2	87
Eficiencia C_1 (%)	73.5	43.1	45.4	25.5
Eficiencia C_2 (%)	57.5	2.4	79.9	0
Eficiencia C_3 (%)	0	31.9	4.1	16.6

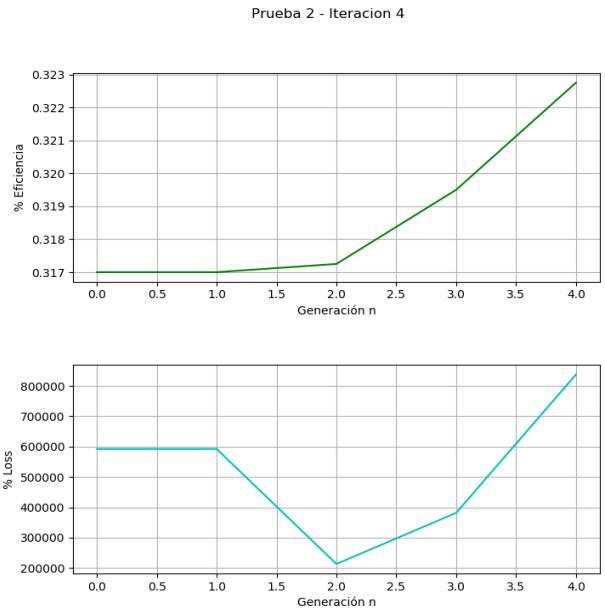

Figura 8. *Gráficos del porcentaje de Eficiencia y Loss por generación. Prueba 2, Iteración 4 de CIFAR-10.*

5.2.3. Prueba 3

	I_1	I_2	I_3	I_4
Eficiencia total(%)	33.47	34.17	33.1	32.8
Loss total(%)	365719	286348	525821	368422
Generaciones(n)	5	5	5	5
Eficiencia C_0 (%)	76.1	4.3	0	70
Eficiencia C_1 (%)	37	54.6	67.6	29.4
Eficiencia C_2 (%)	0.1	76.7	0.1	0.1
Eficiencia C_3 (%)	20.7	1.1	64.7	31.7

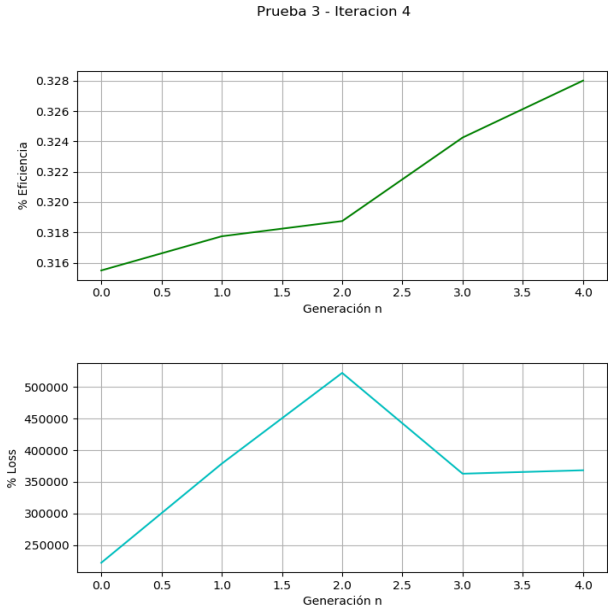

Figura 9. *Gráficos del porcentaje de Eficiencia y Loss por generación. Prueba 3, Iteración 4 de CIFAR-10.*

5.2.4. Imágenes promedio de cada clase

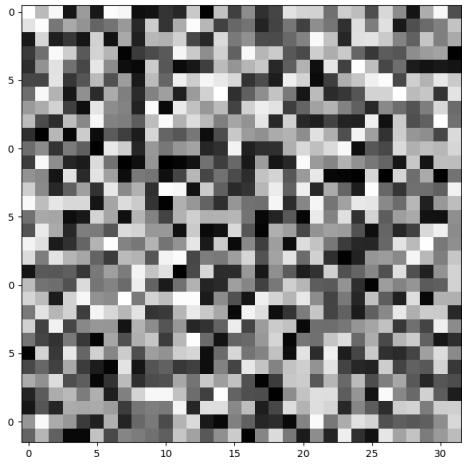

Figura 10. *Imagen promedio de la clase Avión en la Prueba 3, Iteración 1.*

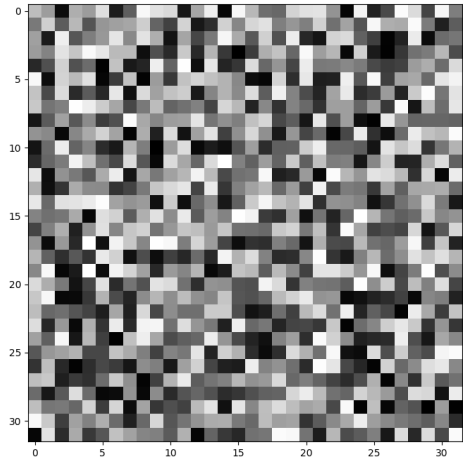

Figura 11. *Imagen promedio de la clase Automóvil en la Prueba 2, Iteración 1.*

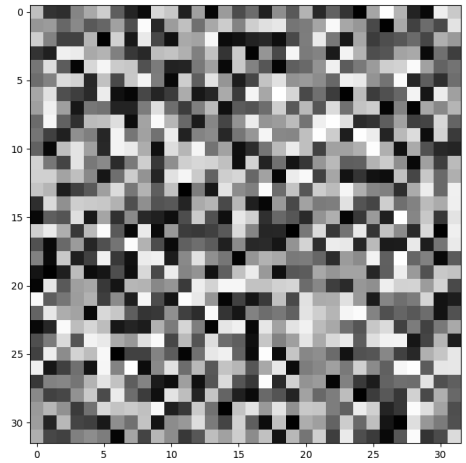

Figura 12. *Imagen promedio de la clase Pájaro en la Prueba 2, Iteración 3.*

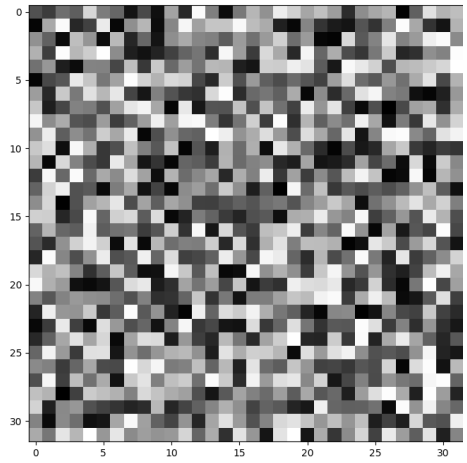

Figura 13. *Imagen promedio de la clase Gato en la Prueba 1, Iteración 4.*

6. Conclusiones

Tenemos dos caras distintas, el modelo lineal optimizado con un Algoritmo Genético dio buenos resultados con Iris, un máximo de 96 % de eficiencia, pero con CIFAR-10, la historia es otra, es completamente ineficiente, un máximo de 34 %.

Esto se debe en su mayoría por los tipos de datos de cada set de datos, pues, cada clase de Iris posee solo 4 características, el largo y ancho del pétalo y el sépalo, en cambio, CIFAR-10 son imágenes que aunque sean pequeñas y se conviertan a escala de grises, la dimensionalidad alta (32x32 = 1024 pixeles) para un algoritmo genético. Esto queda ejemplificado en la cantidad de generaciones necesarias para converger, en algunos casos Iris converge en las primeras tres generaciones, pero casi nunca llega al límite de generaciones permitidas como en el caso de CIFAR-10, nunca converge a algo mejor que un 34 % en cada prueba, y siempre con un Loss muy alto.

Un ejemplo claro de la eficiencia máxima de Iris, es la prueba extra realizada a un 99 % como tope para unas 20 generaciones, la cual solo pudo converger a 96 %, que es un porcentaje bastante aceptable, ver Figura 6.

Las distintas pruebas que se realizaron poseen hiperparámetros que tratan de medir cuánto dura en converger el modelo para cada set de datos. Analizando los gráficos nos damos cuenta como a medida que la eficiencia sube, el Loss baja , por lo que se puede decir que son inversamente proporcionales, por ejemplo en la Figura 3, el Loss se mantiene alto hasta que hay una leve mejora en la eficiencia y el Loss baja considerablemente y sigue bajando mientras la eficiencia sube. Sin embargo, hay casos como en la Figura 5 y la Figura 8, donde el Loss sube con la eficiencia porque se ha encontrado un punto donde no converge a un W mejor que el que ya hay, y en cambio se generan $W's$ ineficientes que afectan ese Loss.

En cuanto a las imágenes promedio generadas para cada clase de CIFAR-10, hay que tener imaginación para poder ver ciertas figuras o partes del objeto o animal que representa, pues son solo 32x32 pixeles, y apenas se distingue en la imagen real. Pero, hay ciertos elementos como alas de avión y llantas que se distinguen.

En conclusión, se puede decir que los Algoritmos Genéticos no son una forma eficiente para un modelo de clasificación lineal, pues depende del dominio del problema para que logre converger en una eficiencia razonable, como el caso de Iris.

Referencias

[1] Rosebrock, A., *Multi-class SVM Loss*, PyImageSearch, 2016, tomado marzo 6, 2018 de https://www.pyimagesearch.com/2016/09/05/multi-class-svm-loss/

[2] Saravanan, C., *Color image to grayscale image conversion.*, In Computer Engineering and Applications (ICCEA), 2010 Second International Conference on, vol. 2, pp. 196-199. IEEE, 2010.

[3] Hilda, G. T., Rajalaxmi, R. R. *Effective feature selection for supervised learning using genetic algorithm*. Paper presented at the 2015 2nd International Conference on Electronics and Communication Systems (ICECS), 909-914. 10.1109/ECS.2015.7125046

[4] Kalaivani, P., Shunmuganathan, K. L. *An improved K-nearest-neighbor algorithm using genetic algorithm for sentiment classification*. Paper presented at the 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], 1647-1651. 10.1109/ICCPCT.2014.7054826

[5] Carbonne, Y., Jacob, C. *Genetic algorithm as machine learning for profiles recognition*. Paper presented at the 2015 7th International Joint Conference on Computational Intelligence (IJCCI), , 1 157-166.