



**BENEMÉRITA UNIVERSIDAD AUTÓNOMA DE PUEBLA**

**FACULTAD DE CIENCIAS DE LA COMPUTACIÓN**

**INGENIERÍA EN TECNOLOGÍAS DE LA INFORMACIÓN**

**MATERIA: MINERÍA DE DATOS**

**PRIMER PARCIAL**

**TORRES GARCÍA JARED BEZAI 201907910**

**GUEVARA MOLINA JOSE EDUARDO 201929618**

**CAMARILLO GALENO GERARDO 201904202**

**FLORES SANCHEZ MARIA AZUCENA 201949353**

**FECHA DE ENTREGA: 03/04/22**

**PRIMAVERA 2022**

Entrenar una red neuronal artificial (RNA) para la detección de incendios forestales en imágenes digitales.

1. (1.0 pts) Construir una muestra (conjunto de imágenes) de incendios forestales, a partir de una estrategia de muestreo propuesta para el problema. Definir con claridad la estrategia seleccionada, estratos considerados, número de instancias, entre otros factores relevantes que consideren para la construcción de la muestra.

### **Estrategia de selección de imágenes**

La propuesta siguiente contempla la construcción de una base de datos con imágenes de incendios forestales:

1. Se obtuvieron 1000 imágenes las cuales representan a la población y se aplicó la siguiente estrategia.
2. Se colocaron dichas imágenes en un repositorio propio, el cuya dirección es <https://github.com/Gera1QWA/mineria.git>

De la cantidad total de imágenes obtenidas se tiene un dominio muy general en donde se encuentran incendios de: día, noche, solo humo; además se contempla vegetación variada: bosques de coníferas, encinos, ocote, matorral y pastizal. No se consideraron imágenes de No Incendios, dado que las imágenes ya recolectadas tienen presencia de áreas verdes, nubes o cielo.

Dentro de la propuesta se busca no generalizar tanto en el dominio de Incendio forestal, por lo que se va a acotar la selección de imágenes, donde:

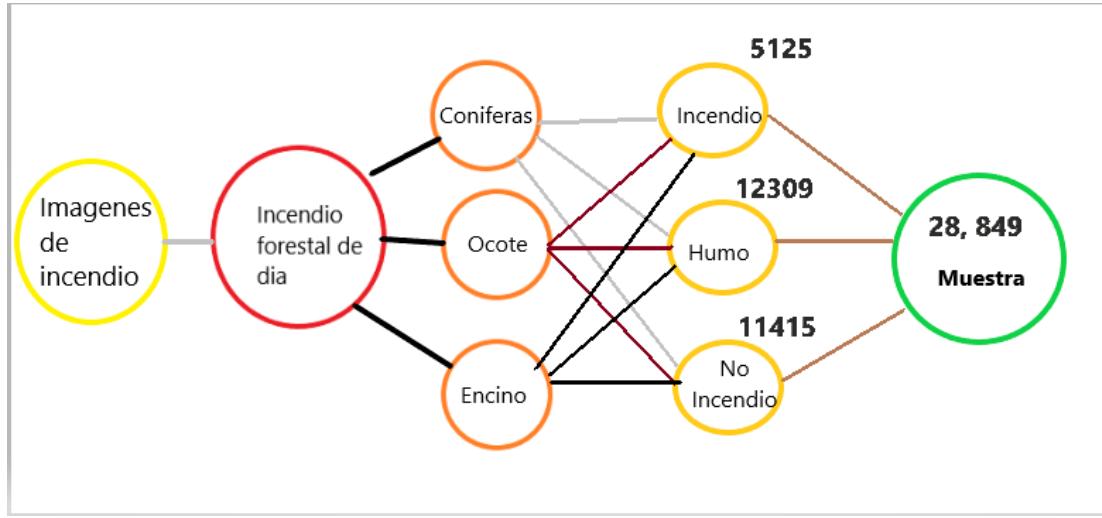
3. De las 1000 imágenes seleccionamos: día y bosque de coníferas, ocote o encinos (no se consideran pastizales y noche).
4. Se obtuvieron **585** imágenes:  
385 con dominio de Incendio (presencia de incendio).  
200 con dominio de humo (presencia de solo humo).

### **Muestreo estratificado**

Las 585 imágenes de incendios forestales (filtradas con el proceso mencionado anteriormente) se van a someter a una transformación de particionamiento donde cada imagen tendrá 72 particiones, es decir, una dimensión de 9 x 8.

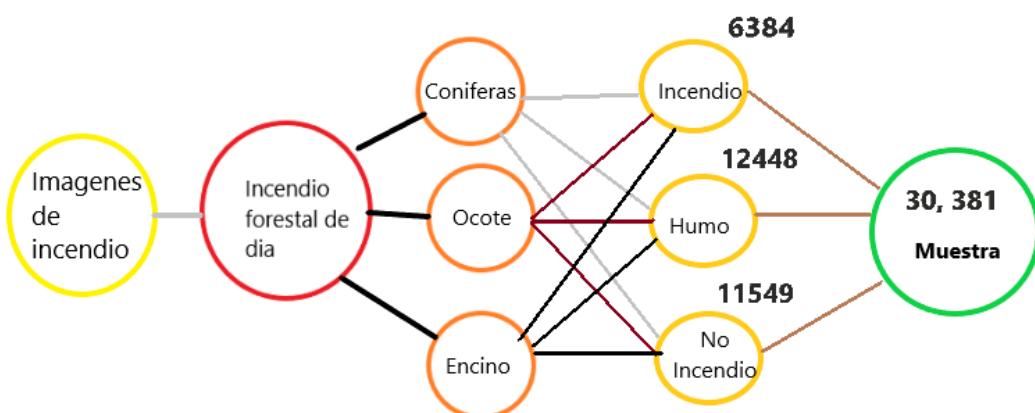
Por lo tanto se van a obtener 42,120 registros en total. Sin embargo N cantidad de registros va a ser descartada, dado que algunas particiones no son puras de clase, es decir en la partición se observan más de 1 clase.

Se etiquetaron y nos dio un total de 28, 849 de instancias (ya con registros descartados) y nuestra muestra quedó de la siguiente manera:



Observamos que teníamos solo 5, 125 instancias de fuego, la cual representaba un 17% de nuestras instancias totales, al detectar que esa era nuestra clase crítica se decidió buscar más imágenes donde el mayor porcentaje de la imagen hubiera presencia de incendio, la nueva distribución de registros quedó de la siguiente manera.

Se etiquetaron 140 imágenes más, obteniendo lo siguiente:



Logramos aumentar 1, 259 nuevos registros a nuestra clase incendio representando un 21% de nuestras instancias totales, se decidió no realizar un muestreo aleatorio simple debido a que no queríamos reducir nuestro volumen en las demás instancias.

Tras el proceso anteriormente mencionado, se detectaron 21,819 instancias impuras, por lo cual fueron descartadas, resultando en total 30,381 instancias utilizables para la siguiente fase.

2. (1.5 pts) Implementar un proceso de particionamiento para el etiquetado de zonas locales de una imagen con incendio forestal. El nivel de granularidad del particionamiento será definido y justificado. Se deberá permitir el etiquetado de las zonas locales como Incendio (I), Humo (H), no incendió (N). Las etiquetas de

clase deberán de recuperarse al momento de generar el vector de características de cada imagen local.

Para el proceso de particionamiento se implementó un programa en el lenguaje python el cual etiqueta las imágenes de incendio forestal.

### 1. Particionamiento de 9 X 8.

Se eligió ese particionamiento porque se consideró que el nivel de pureza respecto a las clases es más alto, además de que el tamaño de cada partición que se muestra en la interfaz es de 73 X 132 px y se aprecia mejor a la vista para ir clasificando.

Lo anteriormente mencionado se basa en lo siguiente:

Si elegimos un particionamiento de 5 x 5, 25 particiones en total, se observa lo siguiente:



Imagen en el etiquetador (5 x 5).

En la imagen se muestra que las particiones encerradas en circulo amarillo contemplan más de 1 clase, por ejemplo en la partición (5,3) hay presencia de clase humo, incendio y no Incendio. Si se determina ese nivel de particionamiento, dará problemas al clasificar cada partición, porque al no ser de clase pura se descarta y por lo tanto, se pierden registros que pueden ser significativos para nuestra muestra.

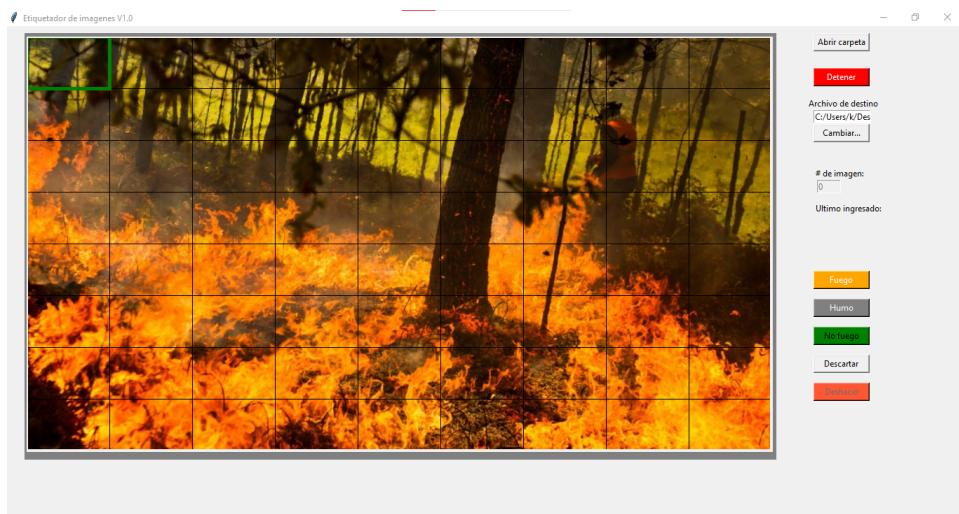
Para el particionamiento de 9 x 8, se observa los siguiente:



En la imagen se muestra que cada partición es más pura, por lo que el descarte de particiones va a ser menor, además de que al obtener registros más puros, sería un factor clave para aumentar el porcentaje de asertividad en el modelo de clasificación en el entrenamiento de la red neuronal.

Si se elige un particionamiento mayor, podría causar problema al momento de clasificar cada partición, ya que en promedio, los píxeles tienden a ser similares, donde se encontrará muy poca variabilidad y uniformidad alta. Y no se busca particularizar tanto en los registros de nuestra muestra.

2. Cada partición la clasifica en Fuego, Humo y No fuego (para cada uno hay un botón, en el que darás click e irás clasificando).
3. Tiene un botón con la funcionalidad de **descartar** aquellas particiones que no sean puras de clase.
4. Cada vez que se da click sobre alguno de los tres clasificadores (Incendio (I), Humo (H), No incendio (N)), se genera un registro, el cual contiene sus descriptores calculados y la clase que le corresponde.
5. Todos esos registros se almacenan en un archivo.txt.



Etiquetador de imágenes de incendio.

3. (1.0 pts) Definir un conjunto de valores característicos a extraer de las imágenes locales a partir de una imagen de entrada. Enlistar cada uno de ellos, definiendo con claridad cómo se calculan y argumentar brevemente por qué cada valor puede aportar a un proceso de entrenamiento de una RNA (considerar como base los valores característicos basados en métricas de tendencia central, así como otros descriptores propios de imágenes digitales – por lo menos otros 5 descriptores).

Para los valores característicos de extracción respecto de las imágenes son:

#### Valores de tendencia central

Para obtener el valor de la media de una imagen, se realizó lo siguiente:

De la imagen original se toma una partición dependiendo el número de celda, de esa partición se le aplicó la función *numpy.mean*, la cual ya está predefinida en python. Y lo que

calcula es la media aritmética a lo largo del eje especificado, y devuelve el promedio de los elementos de la matriz.

Para obtener el valor de la desviación estándar, a partir de la partición de una imagen, se le aplicó la función *numpy.std* (predefinida en python) que es calculada a lo largo de un eje especificado de la matriz, devolviendo la dispersión de una distribución, de los elementos de una matriz.

Estas dos funciones se aplicaron a los canales RGB.

De los canales RGB se obtuvo:

1. MediaR
2. MediaG
3. MediaB
4. DesviaciónR
5. DesviacionG
6. DesviaciónB

### **Matriz de co-ocurrencia**

Es un histograma de los niveles grises de dos dimensiones para un par de píxeles (píxel de referencia y vecino). Es decir, aproxima la probabilidad de distribución conjunta de un par de píxeles.

El cálculo se hace tomando un píxel de referencia, el cual corresponde al píxel central de la ventana y el resultado será un único valor que será colocado en la posición central, luego la ventana se mueve un píxel y se realiza el cálculo nuevamente para obtener un nuevo valor.

Procedimiento para la generación de imágenes de textura se definen cinco variables:

- i) Tamaño de la ventana
- ii) Banda espectral de entrada
- iii) Las texturas derivadas
- iv) Cuantización (número de bits) del canal de salida
- v) La componente espacial (la distancia interpíxel y el ángulo para el cómputo de la co-ocurrencia).

1. Respecto del tamaño de la ventana, esta es cuadrada y con número impar de píxeles.
2. La relación espacial entre el píxel de referencia y su vecino se analiza en 4 direcciones:  $0^\circ$ ,  $45^\circ$ ,  $90^\circ$  y  $135^\circ$ .
3. La distancia interpíxel es de 2.

Después de haber definido los parámetros y haber calculado la matriz de concurrencia se deberá someter a dos procesos más para normalizar, este proceso es necesario ya que de esta manera se podrán calcular los descriptores.

El primer proceso será obtener la matriz simétrica, la forma más sencilla es sumarle a esta matriz su matriz traspuesta. La matriz traspuesta se logra intercambiando las filas y columnas de la matriz original.

$$A^T(j, i) = A(i, j), \text{ siendo } i < m \text{ y } j < n.$$

*Fórmula de la matriz traspuesta*

$$A + B = (a_{ij} + b_{ij})$$

*Formula para sumar dos matrices*

El segundo proceso es expresar la matriz en probabilidad.

$$P_{i,j} = \frac{V_{i,j}}{\sum_{i,j=0}^{N-1} V_{i,j}}$$

*Fórmula para expresar la matriz en probabilidad*

Donde: i es el número de filas y j el número de columnas

V es el valor de la celda i,j en la ventana

P<sub>i,j</sub> es la probabilidad en la celda i,j

N es el número de filas o columnas.

Para implementar la matriz de concurrencia en nuestro programa de python se decidió utilizar una función ya existente, la cual tiene por nombre “graycomatrix” y se importa de la siguiente manera:

*from skimage.feature import graycomatrix*

Para obtener la matriz de concurrencia primero hay que transformar la imagen a escala de grises, y después en la función graycomatrix se especifican los parámetros, aquí se muestra un ejemplo.

```
glcm0 = graycomatrix(imageGp, distances=[2], angles=[0], levels=256, symmetric=True,  
normed=True)
```

Una vez obtenida la matriz de co-ocurrencia se pueden calcular los siguientes descriptores (de cada descriptor se obtienen 4 valores correspondientes al ángulo de inclinación (0°, 45°, 90° y 135°) ):

1. Entropía: El significado de entropía en física es la regularidad del objeto, cuanto más ordenado, menor es la entropía, más desordenada mayor es la entropía. La entropía aquí también representa la cantidad de información de la imagen. Es baja cuando los valores son cercanos a 0 o 1, y es alta cuando los valores de la matriz de co-ocurrencia son iguales.

$$\sum_{i,j}^{N-1} - P_{i,j} \ln(P_{i,j})$$

*Fórmula de la entropía*

Si, P<sub>i,j</sub> es una probabilidad y toma valores entre 0 y 1, entonces el ln (P<sub>i,j</sub>) siempre tomará valores de 0 o negativos. Cuanto más pequeño sea el valor de P<sub>i,j</sub>, es decir que la ocurrencia de esa combinación de píxeles es poco común, el valor absoluto de ln (P<sub>i,j</sub>) será mayor.

2. Correlación: refleja la consistencia de la textura de la imagen. Si hay texturas horizontales en la imagen, el COR de la matriz horizontal es mayor que los valores COR de las matrices restantes.

Los valores de correlación pueden ser de 1 y -1, a, es independiente de las otras medidas. Por lo tanto es esperable que pueda ser usada en combinación con otra medida textural. Algunas propiedades de la Correlación son:

- Un objeto tiene más alta correlación dentro de él que entre objetos adyacentes.
- Píxeles cercanos están más correlacionadas entre sí que los píxeles más distantes

Fórmula de la correlación

$$\sum_{i,j=0}^{N-1} P_{i,j} \left[ \frac{(i - \mu_i)(j - \mu_j)}{\sqrt{(\sigma_i^2)(\sigma_j^2)}} \right]$$

3. Energía: Es la suma de cuadrados de los valores de los elementos de la matriz de congruencia de niveles de grises, también se llama energía que es la uniformidad de distribución de grises de la imagen.

$$Energía = \sqrt{\sum_{i,j=0}^{N-1} P_{i,j}^2}$$

4. Homogeneidad: La homogeneidad es alta cuando la matriz de coocurrencia se concentra a lo largo de la diagonal. Esto ocurre cuando la imagen es localmente homogénea, porque al observar la ecuación de homogeneidad los valores de probabilidad en la matriz son mayores en la diagonal principal y su peso decrece exponencialmente al alejarse de la diagonal.

$$= \sum_{i,j=0}^{N-1} \frac{P_{i,j}}{1 + (i - j)^2}$$

5. ASM: Esta medida da valores altos cuando en la matriz de coocurrencia tiene pocas entradas de gran magnitud, y es baja cuando todas las entradas son similares. Es una medida de la homogeneidad local

$$\sum_{i,j=0}^{N-1} P_{i,j}^2$$

6. Contraste: Este concepto es totalmente opuesto a la homogeneidad donde el contraste tendrá un valor alto si los valores altos están concentrados lejos de la diagonal principal y el peso de la probabilidad aumenta pero en forma cuadrática.

$$\sum_{i,j=0}^{N-1} P_{i,j} (i - j)^2$$

7. Disimilaridad: Es similar al contraste, un alto contraste indica un valor alto de disimilaridad, la diferencia está en que su peso se incrementa linealmente.

$$\sum_{i,j=0}^{N-1} P_{i,j} \cdot |i - j|$$

Finalmente en total se obtuvieron 34 descriptores más el valor de clase (**Incendio**, **Humo**, **No Incendio**).

{Media**R**, Media**G**, Media**B**, Desviación**R**, Desviación**G**, Desviación**B**, Entropia**0°**, Entropia**45°**, Entropia**90°**, Entropia**135°**, Correlación**0°**, Correlación**45°**, Correlación**90°**, Correlación**135°**, Energía**0°**, Energía**45°**, Energía**90°**, Energía**135°**, Homogeneidad**0°**, Homogeneidad**45°**, Homogeneidad**90°**, Homogeneidad**135°**, ASM**0°**, ASM**45°**, ASM**90°**, ASM**135°**, Contraste**0°**, Contraste**45°**, Contraste**90°**, Contraste**135°**, Disimilaridad**0°**, Disimilaridad**45°**, Disimilaridad**90°**, Disimilaridad**135°**, I o H o N }.

Para obtener los descriptores también se decidió utilizar funciones ya existentes y se importa de la siguiente manera:

```
from skimage.feature import graycoprops
```

Para invocar alguna de las funciones de los descriptores se debe enviar la matriz de concurrencia y el nombre del descriptor deseado. Se muestra un ejemplo.

```
energia = graycoprops(MatrizConcurrencia, 'energy')
```

4. (1.0 pts) Realizar un análisis y argumentar que tipo de validación cruzada (que valor de K o valores) se usará en su experimentación, considerando que se debe usar la mayor cantidad de cruces posibles, garantizando en todo momento la correcta medición de lo que sucede en cada clase.

Para el análisis de validación cruzada se tomó en cuenta el número total de instancias de la muestra. El valor de validación cruzada se encuentra en el rango de valores de 2 a 10

donde el valor de k es el número de veces que se va a repartir la muestra, por ende si el valor de k es igual a dos se tendrá una partición de entrenamiento y otra de validación. Así respectivamente con 3,4,5,etc.

Es importante considerar el número de particiones de la muestra porque si tenemos una partición muy pequeña puede que los datos no sean suficientes para el grupo de entrenamiento y se dejen pocas instancias importantes (la clase crítica). También es importante reconocer el costo computacional.

Tomando en cuenta el número de valores de nuestra muestra (30 381) y el el número de registros en la clase crítica la de incendios con 6384 instancias, se hizo la siguiente tabla.

	total	2	3	4	5	6	7	8	9	10
	30381	15190.5	10127	7595.25	6076.2	5063.5	4340.14286	3797.625	3375.66667	3038.1
0	12448	6224	4149.33333	3112	2489.6	2074.66667	1778.28571	1556	1383.11111	1244.8
1	6384	3192	2128	1596	1276.8	1064	912	798	709.33333	638.4
2	11549	5774.5	3849.66667	2887.25	2309.8	1924.83333	1649.85714	1443.625	1283.22222	1154.9

Se puede ver que para nuestro caso la clase crítica limita mucho el valor que puede tomar, sobre todo para valores de arriba de 7 donde son muy pocas instancias comparadas con las de los otros estratos. Un valor que llama la atención es el valor de 4 porque el tamaño de muestras de clase humo son casi la mitad si se escogió este valor las particiones quedarían de 20% cada una. El otro valor que se tomó como candidato fue de 6 pero se pensó que el costo de los equipos se elevaría y no habría suficientes instancias para nuestra clase crítica. Así que tomamos el valor de 5, debido a que tiene suficientes instancias de cada clase y también sería más fácil de procesar con ese valor en nuestras computadoras, este valor también nos da un mayor número de cruces a comparación de 2 y 3.

5. Definir la metodología a utilizar para el entrenamiento de la RNA. Para ello, se deberá argumentar:

a. (0.5 pts) Número de neuronas (rango) que se usará para la RNA

El número de neuronas se encuentra en el siguiente rango:

Mínimo: **3** Corresponde al número de clases, el cual es el número menor entre clase y número de atributos.

Máximo: **38** Correspondiente a la suma del número de atributos más el número de clases.

b. (0.5 pts) Número de capas ocultas a usar así como rango de valores para el número de épocas.

El número de capas se define de acuerdo a los primeros resultados del entrenamiento de la red neuronal. En este caso se empieza con una arquitectura simple, es decir una sola capa.

Una vez que se obtengan resultados del primer entrenamiento con el número de neuronas, se elegirán 5 mejores resultados, estos resultados se les aplicará entrenamiento de 2 a 5 capas.

El número de épocas se define de acuerdo a la dimensión de la base de datos, en este caso se tomará entre 5% a 15% respecto del número total de instancias.

Al final se llegó a **2500** épocas, corresponde al 8.22% del total de la base de datos, 30 381 instancias.

c. (0.5 pts) Valores a usar para el Learning Rate

Este hiper-parámetro indica qué tan largo será el camino que tome el algoritmo de optimización y está en un rango entre 0 y 1.

Se recomienda un valor entre [0 ... 0.3].

Y se optó por **0.3**, ya que se busca que la actualización de pesos no sea tan fuerte con un valor mayor o los movimientos sean lentos, con un valor más pequeño.

Si el valor es muy pequeño la actualización se puede quedar atrapada en un mínimo local y los valores de los pesos ( $W$ ) no cambien correctamente, además la red neuronal tardará mucho más tiempo en optimizar, a pesar de que el ajuste en el hiperplano en el hiperespacio sea preciso.

Por otro lado si los valores son muy altos la actualización puede pasarse del punto perfecto que sería el mínimo global y nunca encontrarlo y aunque la tasa de aprendizaje sea más rápido nunca llegará al mínimo global.

d. (0.5 pts) Valores a usar para el Momentum

Referente al movimiento inercial en el hiperespacio, el cual ayuda a la aceleración de la tasa de aprendizaje, los valores a seleccionar deben estar entre 0 y 1.

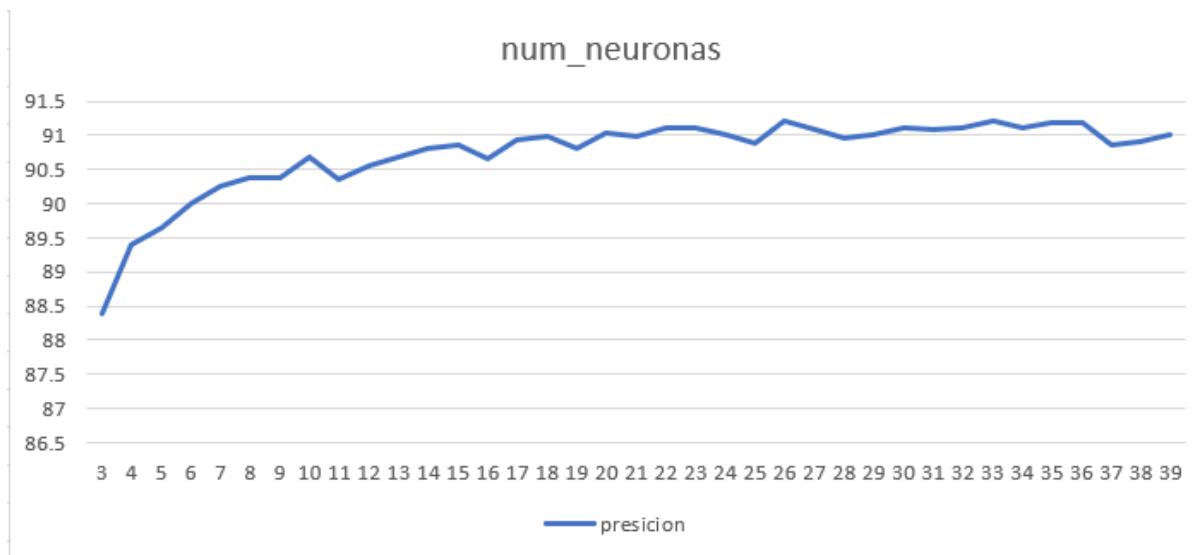
Se eligió un valor de 0.2 para el primer experimento.

Para la fase de entrenamiento con distintos valores de este parámetro, el rango recomendado para empezar a experimentar es de [0 - 0.2].

6. (2.5) Realizar un proceso de búsqueda de parámetros que permitan encontrar la mejor clasificación que se pueda obtener con la RNA y la muestra a trabajar. Para ello, describir:

a. Mostrar a través de una gráfica y explicar cuántas neuronas presentan los mejores resultados por cada capa oculta (de acuerdo a la cantidad de capas que se definió en el punto 4.b)

Se implementó un algoritmo en lenguaje JAVA, el cual, usando la librería de funciones de weka, obtiene el porcentaje de las instancias correctamente clasificadas en el conjunto de prueba usando los parámetros anteriormente especificados, adicionalmente se realizaron experimentos con hasta 39 neuronas, dicho experimento no significó un cambio en el resultado final, puesto que el porcentaje de instancias clasificadas correctamente mantuvo la tendencia de oscilación. Los resultados se han graficado y se interpretan de la siguiente manera.



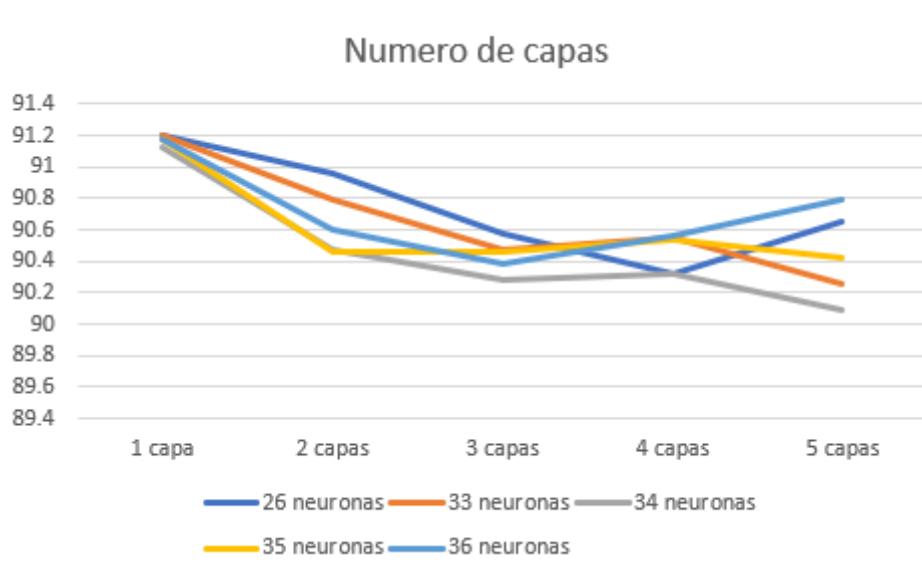
Entrenamiento de 3 a 39 neuronas, con 1 capa, 2500 épocas  
0.3 de learning rate y 0.2 de momentum.

Sobre la gráfica anterior, se pueden determinar algunos elementos, como por ejemplo la gráfica presenta un aumento en las instancias correctamente clasificadas, sin embargo, hay una oscilación muy marcada hacia los últimos números de neuronas, y debido a que no hay una sola cantidad de neuronas que ofrezca un resultado convincente, se decidió seleccionar los 5 mejores resultados, los cuales son:

- 26 neuronas (91.2017%)
- 33 neuronas (91.2017%)
- 36 neuronas (91.1819%)
- 35 neuronas (91.1754%)
- 34 neuronas (91.1227%)

En base a estos resultados, se realizará la siguiente etapa de entrenamiento.

Usando una modificación del programa antes mencionado, se realizaron 4 experimentos más por cada neurona seleccionada, en los que se prueba la precisión de los resultados por 2, 3, 4, y 5 capas en los resultados obtenidos en el punto anterior.



Entrenamiento de 1 a 5 capas, con los 5 mejores resultados de número de neuronas en la primera capa.  
Parámetros: 2500 épocas ,0.3 de learning rate y 0.2 de momentum.

Tal como se puede apreciar, todos los experimentos nos llevan a una conclusión, una sola capa obtiene resultados mejores que emplear un número de capas más grande, alcanzando valores superiores al 91% de instancias correctamente clasificadas, incluso se experimentó con un menor número de épocas (400 y 1000 épocas) y no vimos mejoras al aumentar el número de capas.

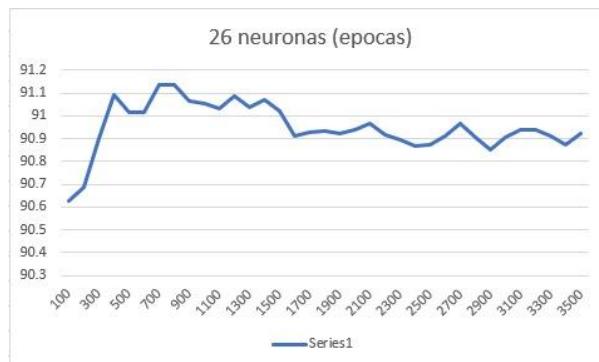
b. Mostrar a través de una gráfica los experimentos realizados para determinar cuál es el número de épocas donde se maximizan los resultados de precisión.

Una vez que se eligieron los mejores resultados en el número de capas y neuronas (1 capa y 5 mejores resultados de número de neuronas), se procedió a la experimentación con el número de épocas.

### Entrenamiento con épocas.

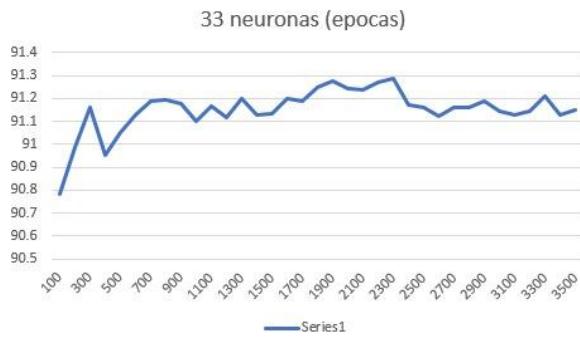
El rango de experimentación fue de 100 a 3500 épocas con una sumatoria de 100 en 100. Los demás parámetros se quedaron con el valor asignado al principio de la experimentación (learning rate = 0.3, momentum = 0.2)

- 26 neuronas y 1 capa



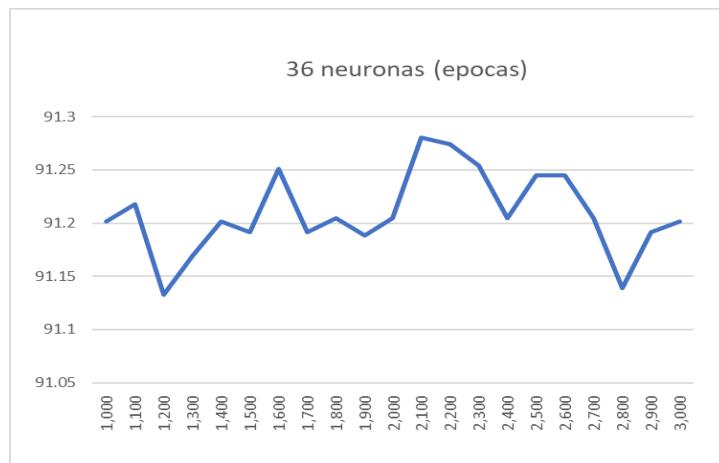
Sobre el experimento con 26 neuronas con el propósito de obtener un número óptimo de épocas, se obtuvo que el número óptimo es 2300, con 91.17% de instancias correctamente clasificadas

- 33 neuronas y 1 capa



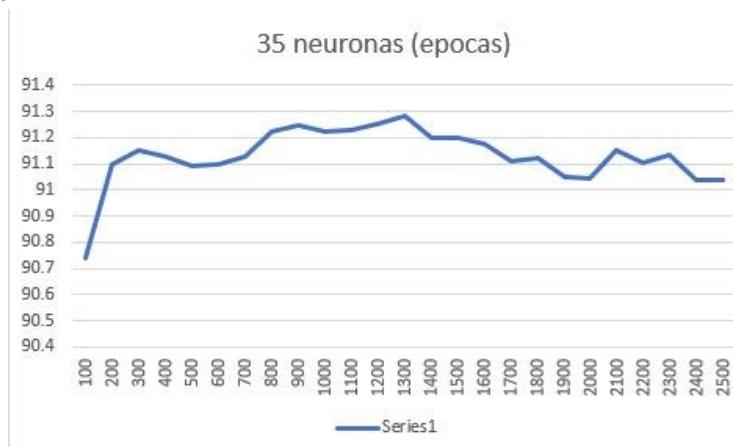
Como se puede apreciar, los resultados toman un comportamiento oscilatorio, sin embargo se decidió optar por el resultado que arroja mayor número de instancias correctamente clasificadas, lo cual nos indica que el número óptimo de épocas para 33 neuronas es 2300, con 91.28%

- 36 neuronas y 1 capa



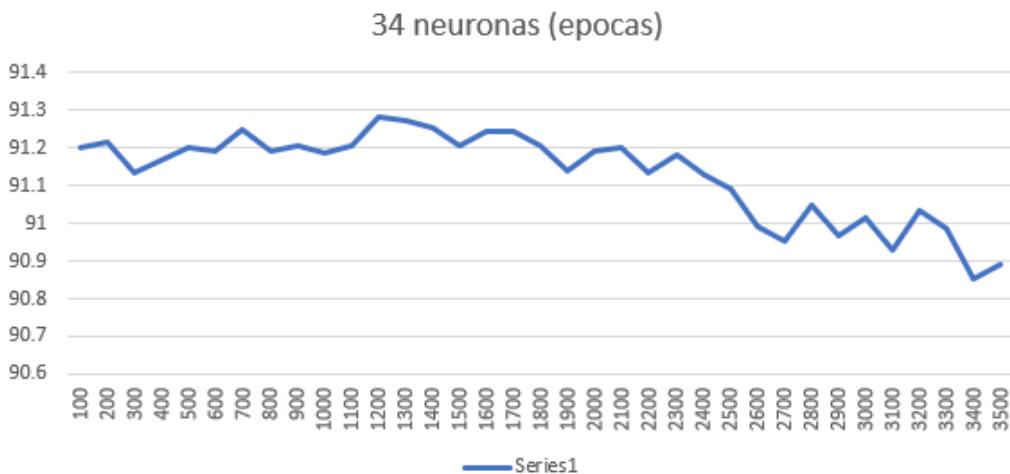
Los resultados para el número óptimo de épocas para 36 neuronas se muestran en la gráfica y el resultado con mejor porcentaje es del 91.27% con 2200 épocas.

- 35 neuronas y 1 capa



En el caso de 25 neuronas, la gráfica resulta tener un comportamiento oscilatorio y el mejor resultado obtenido es 91.28%, usando 1300 épocas.

- 34 neuronas y 1 capa



En el caso del experimento de 34 neuronas, el resultado evidente es emplear únicamente 1200 épocas, con 91.2% de precisión al clasificar

- c. Mostrar y explicar los experimentos realizados para encontrar los valores ideales del Learning Rate y Momentum que optimizan la precisión global

La modificación al programa ya antes mencionado para realizar la experimentación del learning rate y momentum fue la siguiente:

Dentro del ciclo para la evaluación individual de cada experimento, se modifica el valor por defecto del parámetro deseado a través de alguno de estos métodos:

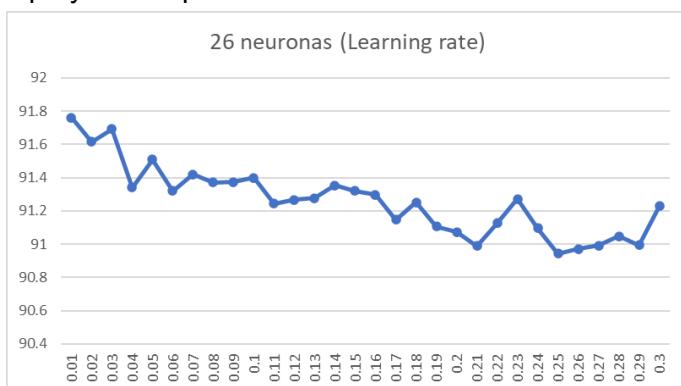
- set LearningRate(n);
- set Momentum(n);

### Experimentos con Learning rate

Los experimentos realizados para **learning Rate** se muestran en las siguientes gráficas:

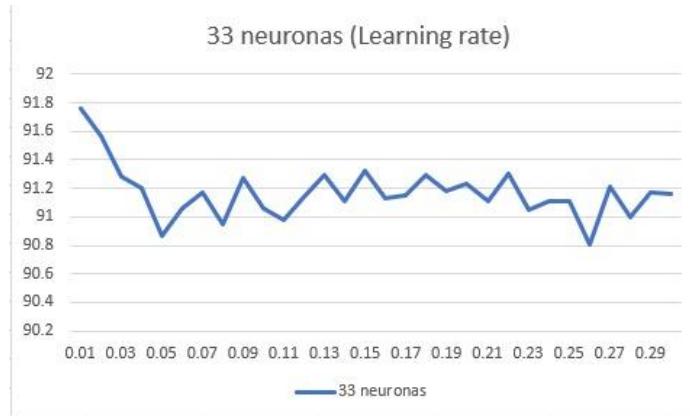
El momentum se quedó con el valor asignado al principio de la experimentación (0.2).

- 26 neuronas, 1 capa y 2300 épocas



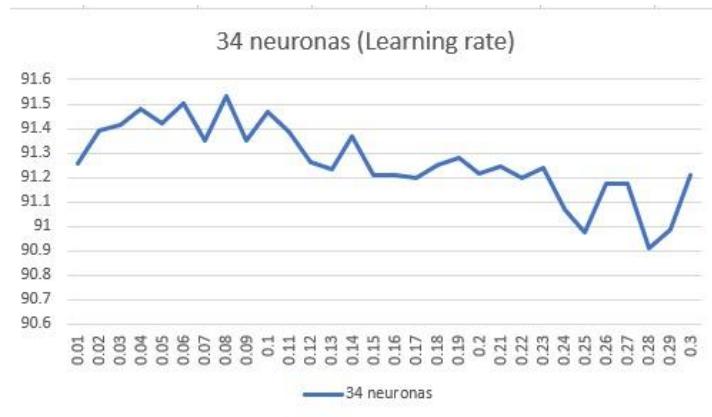
Como se muestra en la gráfica el valor con mejor resultado de learning rate es 0.01 con un 91.76% de instancias correctamente clasificadas.

- 33 neuronas, 1 capa y 2300 épocas



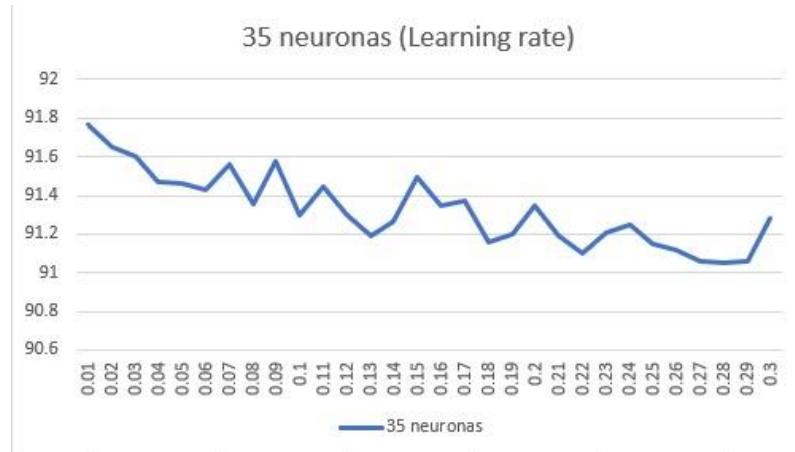
En la gráfica se observa que el mejor resultado es 0.01 con 91.75% de instancias correctamente clasificadas.

- 34 neuronas, 1 capa y 1200 épocas



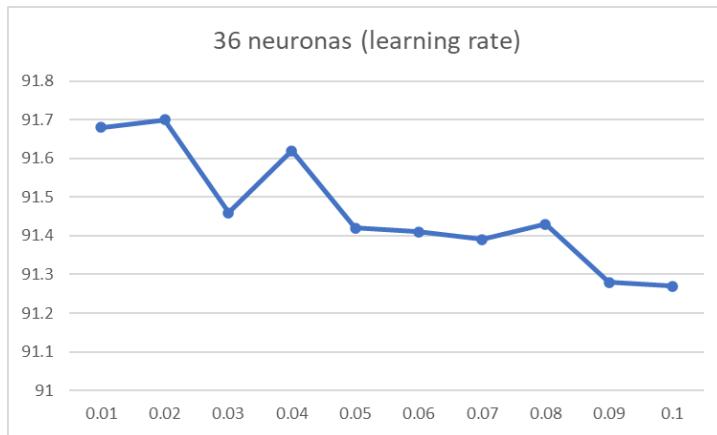
La gráfica muestra que 0.08 como valor de learning rate tiene 91.35%, el cual es el de mayor porcentaje en instancias correctamente clasificadas.

- 35 neuronas, 1 capa y 1300 épocas



Para este caso, el experimento con mejor resultado es con 0.01 con un 91.76% de exactitud.

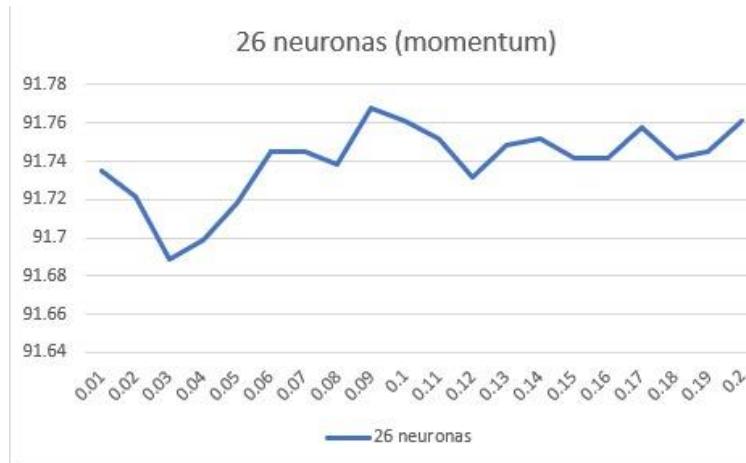
- 36 neuronas, 1 capa y 2200 épocas



En este entrenamiento el valor con más exactitud es el de 0.02 con 91.7%.

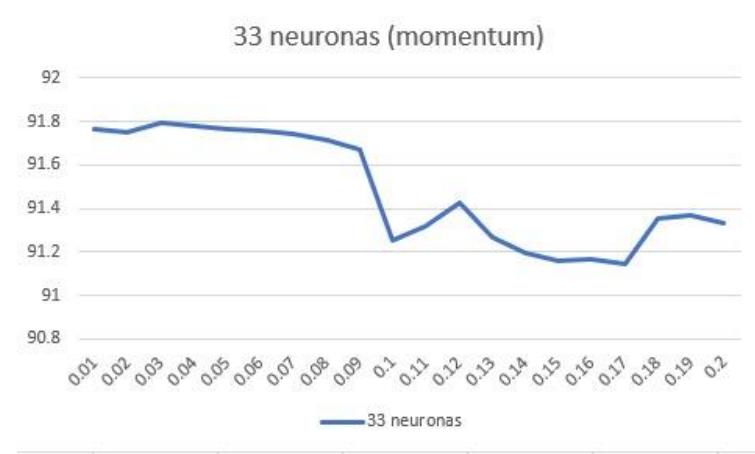
Los experimentos realizados para **momentum** se muestran en la siguientes gráficas:

- 26 neuronas, 1 capa, 2300 épocas y 0.01 de learning rate.



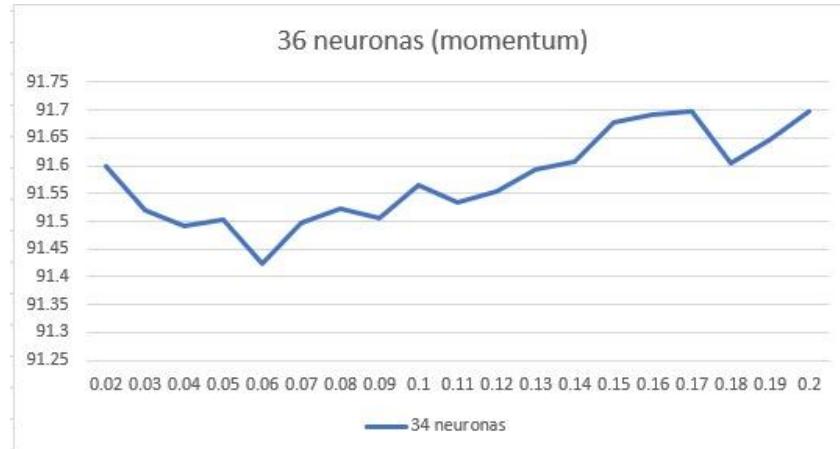
Para este caso la gráfica presenta que el mejor resultado es 0.09 con 91.76% de exactitud.

- 33 neuronas, 1 capa, 2300 épocas y 0.01 de learning rate



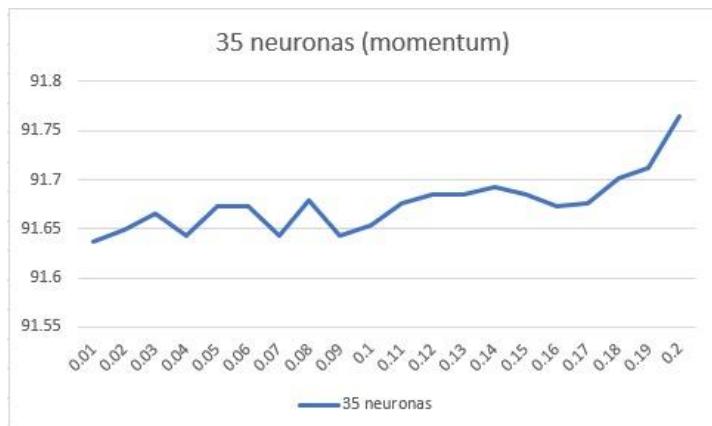
El momentum seleccionado para esta configuración de 33 neuronas es 0.03, con 91.79% de precisión.

- 36 neuronas, 1 capa, 2200 épocas y 0.02 de learning rate.



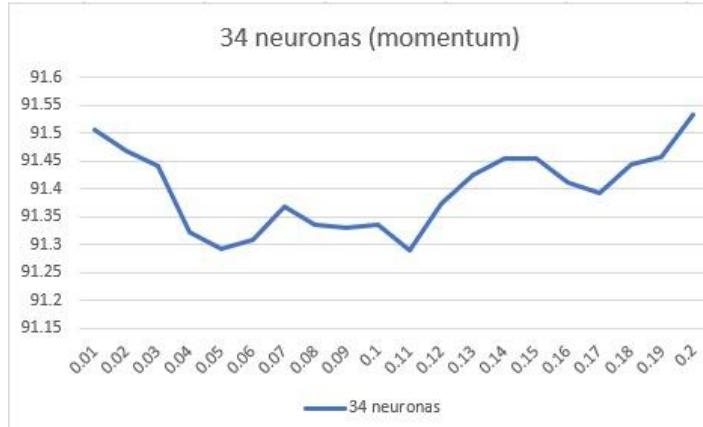
El valor con mejor resultado es 0.17 con 91.6% de exactitud.

- 35 neuronas, 1 capa, 1300 épocas y 0.01 de learning rate



El valor con mejor resultado es 0.2 con un 91.76% de instancias clasificadas correctamente.

- 34 neuronas, 1 capa, 1200 épocas y 0.08 de learning rate.



Los resultados mostrados en la gráfica, indica que el mejor resultado es 0.2 con un 91.534% de instancias correctamente clasificadas.

7. (1.0 pts) Hacer un análisis de los resultados obtenidos, donde se explique la forma en la cual influyó la búsqueda de parámetros para obtener un mejor resultado en el proceso de clasificación.

Gracias a la cantidad de experimentos que se hicieron en la red podemos decir con certeza que no es un proceso instantáneo, de hecho es un proceso tedioso y muy tardado, que requiere de mucho análisis y paciencia.

En la disposición de buscar los valores más elevados para la red, empezamos por el primer valor que fue el número de neuronas, en el rango de 3 a 37. En consecuencia de la gráfica resultante de este experimento se logró ver que la cantidad de neuronas influye en los resultados significativamente, pero eso no es lo único, para poder hacer una mejor evaluación es necesario también encontrar el número de capas ideales para la clasificación, esto hizo que se hicieron los experimentos de 5 capas con el mismo número de neuronas.

Con los resultados anteriores y concluyendo que el número de capas mayor no siempre resulta en un mejor resultado, por el tiempo, decidimos que lo mejor era conservar una capa y con un número de neuronas relativamente alto, el siguiente punto a tomar en cuenta fueron las épocas, se hicieron los experimentos correspondientes de 100 hasta 3000, en las gráficas ,se puede ver que las épocas afectan el resultado dependiendo el número de épocas pero si se dejan muchas épocas este resultado puede variar muy poco y ser engañoso porque se puede estar cayendo en un sobreajuste de la red.

Los dos últimos parámetros a considerar son el momentum y el learning rate, con este mismo se hicieron los experimentos en el rango de 0.01 a 0.3, como análisis para nuestro caso, nos damos cuenta que con los valores más bajos existe una cantidad más elevada de éxito en la red y conforme se hace más grande la cantidad la red empieza a tener más error en unos casos es más significativo que en otros.

Hasta ahora los resultados han sido cercanos en los experimentos, por lo que el último valor a considerar debería dar un resultado un poco distinto en las gráficas, y así fue, el momentum hace que la red tenga mayor numeros de aciertos en los experimentos de 35,26

y 34 neuronas, siendo en 34 donde se nota más el cambio de aciertos, en las redes de 35 y 26 también se nota una leve mejoría de resultados.

Mejores resultados (1 capa)					
#neurona	# Capas	Épocas	Learning-rate	Momentum	Porcentaje de clasificación correcta
26	1	2300	0.01	0.09	91.76%
33	1	2300	0.01	0.03	91.79%
34	1	1200	0.08	0.2	91.534%
35	1	1300	0.01	0.2	91.76%
36	1	2200	0.02	0.17	91.6%

Sobre el mejor modelo de todo el entrenamiento considerando todos los experimentos hasta ahora, el resultado es:

- 33 neuronas
- 2300 épocas
- learning rate de 0.01
- momentum de 0.03

Esto con una precision de 91.79% de instancias correctamente clasificadas

A Continuación se muestra la matriz de confusión del modelo anterior

==== Overall Confusion Matrix ===

a    b    c    <- classified as
11454    263    731      a = 0 HUMO
222    6020    142      b = 1 INCENDIO
1002    132    10415      c = 2 NO INCENDIO

De aquí se puede observar que el error en la clase de incendio fue bajo, de las 6,384 clasificó mal 364 (5.7%), lo cual a nuestro parecer es bajo. el mayor error se encuentra en las otras dos clases, y creemos que es por las siguientes dos razones, la primera es porque son más instancia que hay, y la segunda es porque el modelo podría llegar a confundir algunas nubes con el humo.

En general el momentum, el número de neuronas y el número de épocas afectaron de manera más significativa la red, esto se puede notar en la cantidad de instancias que clasificaron de manera correcta las redes. Cabe resaltar que al hacer varios experimentos a la vez nos dio un abanico de comparación que si solo hubiéramos usado un número específico de capas y de neuronas. Para el momentum en valores más altos hace que la red sea un poco más flexible resultando en un mejor valor de instancias bien clasificadas. Por último las épocas, el número de épocas es muy importante porque la red se puede estropear fácilmente colocando un número elevado de épocas.

## BIBLIOGRAFÍA

- <https://weka.sourceforge.io/doc.stable/>
- <https://github.com/search?l=Java&q=using+weka+as+API&type=Code>
- <https://github.com/aiformankind/wildfire-dataset/tree/master/wildfire-smoke/bing/positive>
- <https://data.mendeley.com/datasets/gjmr63rz2r/1>
- <https://numpy.org/doc/stable/reference/generated/numpy.mean.html>
- <https://numpy.org/doc/stable/reference/generated/numpy.std.html>
- [http://www3.inpe.br/unidades/cep/atividadescep/jornada/programa/t-9\\_trab\\_27.pdf](http://www3.inpe.br/unidades/cep/atividadescep/jornada/programa/t-9_trab_27.pdf)