ISSN 1870-4069

# Algoritmo de clasificación basado en la función Softmax

Arturo Zúñiga-López, Carlos Avilés-Cruz,
Andrés Ferreyra-Ramírez, Eduardo Rodríguez-Martínez

Universidad Autónoma Metropolitana,
Departamento de Electrónica,
División de Ciencias Básicas e Ingeniería,
México

{azl,caviles,fra,erm}@azc.uam.mx

**Resumen.** El análisis de datos en los campos de *ciencia de datos* y la *minería de datos* involucra la agrupación y clasificación de información que posee propiedades comunes. Existe gran cantidad de algoritmos para clasificación, sin embargo, uno solo no es capaz de clasificar todo tipo de distribuciones y densidades. En el presente trabajo se introduce un algoritmo de clasificación basado en la función softmax implementada en redes neuronales artificiales. El algoritmo propuesto concatena un número $k-$funciones *softmax* para llevar a cabo clasificaciones de datos linealmente no separable. Se presentan resultados usando bases de datos tipo benchmark, en donde se muestra casos "simples" de clasificación, i.e., linealmente separables, así como casos "complejos", i.e., casos no linealmente separables. Se presenta el cambio del espacio de partición de las funciones *softmax* en función del número de iteraciones.

**Palabras clave:** Función Softmax, agrupamiento, clasificación, atributos, bases de datos sintéticas.

## Classification Algorithm Based on the Softmax Function

**Abstract.** Data analysis in *data science* and *data mining* involves the clustering and classification of information with common properties. There are many algorithms for classification; however, no one of them is capable of classifying all types of distributions and densities. In the present work, a classification algorithm based on the *softmax* function implemented in artificial neural networks is introduced. The proposed algorithm concatenates many $k-$softmax functions to carry out linearly non-separable data classifications. Results are presented using benchmark databases, where simple classification cases are shown, i.e., linearly separable, as well as more complex cases, i.e., non-linearly separable cases. The change of the partition space of the *softmax* functions is presented according to the number of iterations.

**Keywords:** Softmax function, clustering, classification, features, synthetic databases.

## 1. Introducción

El creciente interés por agrupar y clasificar información es debido a la gran cantidad de información que se maneja en las redes sociales y en Internet en general. La ardua tarea de agrupar y clasificar información -atributos- que poseen o comparten características comunes, es una tarea angular en las áreas de, *ciencia de datos*, *minería de datos* y *reconocimiento de patrones*. A pesar de la gran cantidad de algoritmos tanto de agrupamiento como de clasificación, aún no se tienen un solo algoritmo que funcionen para agrupar y clasificar todo tipo de datos.

Los algoritmos de agrupamiento-clasificación se clasifican de acuerdo con su criterio de agrupamiento, y también de acuerdo con la cantidad de datos que pueden manejar (dimensiones y extensión). Existen diferentes clasificaciones sobre los algoritmos de agrupamiento-clasificación, por ejemplo, Dong et al. [8] los clasifican en *supervisados* y *no supervisados*. Una clasificación más extensa es la hecha por Zhao et al. [23] en donde los autores presentan las clasificaciones siguientes: *Particional* [10,9,13], *Jerárquica* [22,17,4], *Basada en densidad* [12,3,14], *Basada en rejilla* [6,5,21], *Espectral* [20,15,16], *Gravitacional* [19,7,2], y *basada en redes neuronales* [1,18,11] . La presente propuesta pertenece al grupo de los *Particionales*, es decir, que particiona o divide es espacio de los atributos.

El algoritmo propuesto particiona el espacio de los atributos de acuerdo con funciones *Softmax* (funciones de tipo exponencial normalizadas, se explica en capítulo 2). Se utilizó redes neuronales artificiales para su diseño y se implementó en lenguaje de programación Python 3.7 con todos los módulos necesarios. La metodología propuesta fue evaluada usando las bases bi-dimensionales *Shape* ampliamente usadas en la bibliografía de agrupamiento-clasificación. La base de datos *Shape* se tomó del grupo *Machine Learning* de la Universidad del Este de Finlandia [1]. Esta base se tomó porque contiene distribuciones simples (linealmente separables, R15, D31). Distribuciones más complejas en donde clases están embebidas en otras (Jain, flame, and compound). Por último, distribuciones muy complejas en forma de espiral con clases embebidas (spiral and pathbased).

El resto de este documento esta organizado de la siguiente manera. La sección 2 introduce la metodología propuesta. La sección 3 introduce los conjuntos de datos utilizados y el protocolo de evaluación. La sección 4 muestra los resultados experimentales. Finalmente, la sección 5 discute las limitaciones y conclusiones de este trabajo.

[1] http://cs.joensuu.fi/sipu/datasets/

## 2. Propuesta

La presente propuesta está basada en el clasificador *Softmax*, cuya función de probabilidad está en términos de funciones exponenciales normalizadas. Sea $X \in \Re^d$ un punto muestra en el espacio $d$ y contando con $N$ puntos muestra.

Sea $c$ la etiqueta de la clase correspondiente, con $c = 1, c = 2, c = 3..., c = K$ y $K$ es el número total de clases. La ecuación 1 representa el cálculo de las probabilidades para $K-$clases:

$$\Psi(X, W) = \begin{bmatrix} p(c=1|X) \\ p(c=2|X) \\ p(c=3|X) \\ \vdots \\ p(c=K|X) \end{bmatrix} = \frac{1}{\sum_{j=1}^{K} e^{W_j^T \cdot X}} \begin{bmatrix} e^{(W_1^T \cdot X)} \\ e^{(W_2^T \cdot X)} \\ e^{(W_3^T \cdot X)} \\ \vdots \\ e^{(W_K^T \cdot X)} \end{bmatrix}. \tag{1}$$

Siendo $W$ el vector de pesos, con $W \in \Re^d$.

Por otro lado, Si consideramos a la entropía cruzada $H$ como métrica para medir la precisión entre *modelos pronósticos probabilísticos* y los valores reales, la ecuación 2 representa la entropía cruzada:

$$H(p, q) = -\sum_{X} p(X) log[q(X)]. \tag{2}$$

Siendo $p$ la probabilidad verdadera y $q$ la probabilidad obtenida (o predicha) a partir de un modelo, en éste caso será $\Psi(X, W)$ (ver ecuación 3):

$$H(p, q) = -\sum_{X} p(X) log[\Psi(X, W)]. \tag{3}$$

Cabe remarcar que, en realidad $H$ es la función que observa el error, de ahí que vamos a definir a la función de costo por medio de la ecuación 4. Observamos que en la función de costo está en término de los vectores de peso $W$. Es decir, la optimización versará en encontrar los vectores de peso $W$ que mejor separen las clases:

$$J(W) = \frac{1}{N} \sum_{i=1}^{N} H(p, q), \tag{4}$$

donde $N$ es el número de observaciones.

Ahora la función $J(W)$ se puede reescribir de la siguiente manera:

$$J(W) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{K} p(c) log(q(c)). \tag{5}$$

Si consideramos que las clases $c$ (o etiquetas) las podemos expresar como 'one hot' por ejemplo $c_1 = [1, 0, 0, ...]$ para la clase 1, $c_2 = [0, 1, 0, ...]$ para la clase 2 y así para $K$ clases. La ecuación 5 se puede escribir como:

$$J(W) = -\frac{1}{N} \sum_{i=1}^{N} \sum_{c=1}^{K} \delta(y_i) \cdot log \left[ \frac{e^{(W_c^T \cdot X_i)}}{\sum_{j=1}^{K} e^{(W_j^T \cdot X_i)}} \right], \tag{6}$$

donde $\delta(y_i)$ es una delta de Dirac y vale 1 para una $c$ clase dada.

Para el cálculo de actualización de pesos $W$, se determinará de la siguiente forma (ver ecuación 7):

$$W_j = W_j - \alpha \nabla_{W_j} J(W), \tag{7}$$

donde $W_j$ es un vector de pesos de la matriz de pesos $W \in \Re^{d \times K}$, $\alpha$ es tasa de aprendizaje y $\nabla_{W_j} J(W)$ es el gradiente del costo para esa vector $j$.

Por último, el gradiente descendente se calcula de la expresión 8 siguiente:

$$\nabla_{W_j} J(W) = -\frac{1}{N} \sum_{i=1}^{N} [\delta(c_i) - \frac{e^{(W_c^T \cdot X_i)}}{\sum_{j=1}^{K} e^{(W_j^T \cdot X_i)}}] X_i. \tag{8}$$

Finalmente, remarcamos que los espacios de partición delimitado por el clasificador propuesto serán linealmente separables, pudiendo ser linealmente separable a trozos.

## 3. Bases de datos

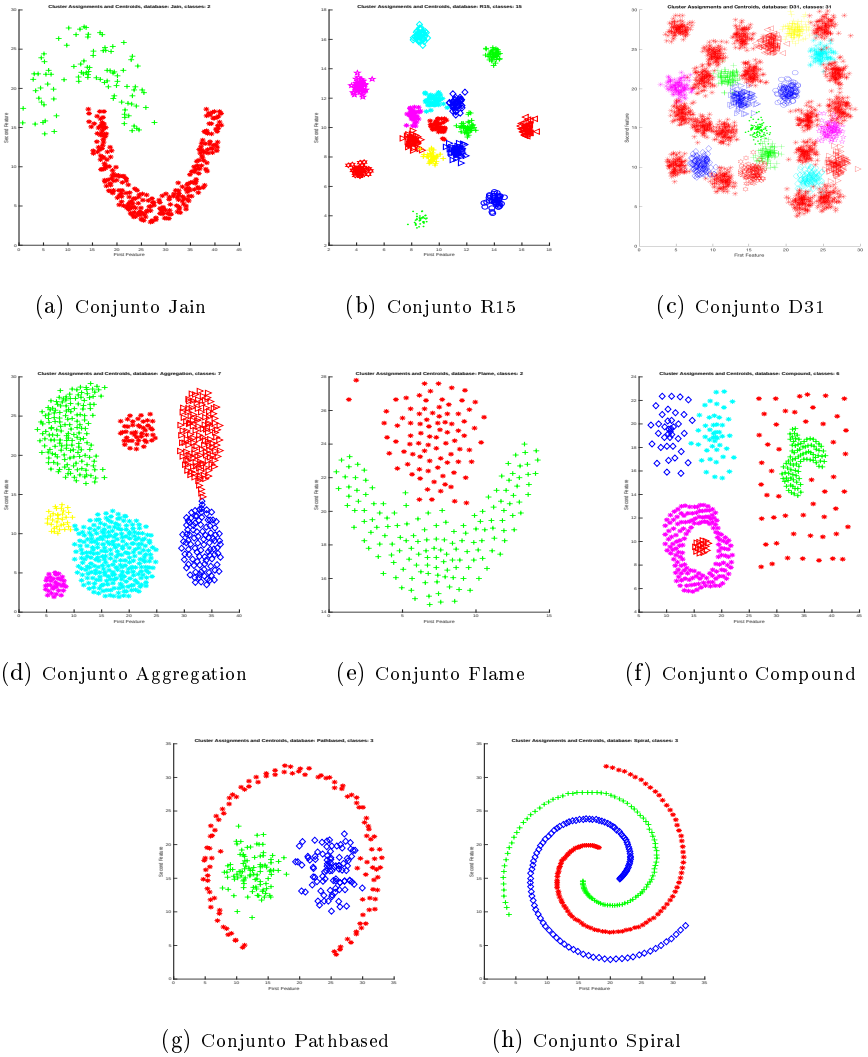
Se utilizaron bases de atributos bi-dimensionales tomados del grupo de aprendizaje de máquina de la Universidad del Este de Finlandia [2]. Se eligió la base de atributos *Shape* por sus características desde las simples, distribuciones esféricas separadas, distribuciones embebidas y aún más complejas, en forma de espiral.

La Fig. 1 muestra los 8 conjuntos de datos, así mismo, la Tabla 1 muestra las características de cada conjunto. Los conjuntos de datos son : *Aggregation* (Fig. 1(a)), *Compound* (Fig. 1(b)), *Pathbase* (Fig. 1(c)), *Spiral* (Fig. 1(d)), *D31* (Fig. 1(e)), *R15* (Fig. 1(f)), *Jain* (Fig. 1(g)), y *Flame* (Fig. 1(h)). El conjunto de bases de atributos contiene desde 2 clases hasta 31. El mínimo número de atributos es de 240 y el máximo de 3,100.

**Tabla 1.** Bases de datos sintéticas *Shape* a clasificar.

| Conjunto | No. clases | No. muestras |
|---|---|---|
| *Aggregation* | 7 | 788 |
| *Compound* | 6 | 399 |
| *Pathbase* | 3 | 300 |
| *Spiral* | 3 | 312 |
| *D31* | 31 | 3100 |
| *R15* | 15 | 600 |
| *Jain* | 2 | 373 |
| *Flame* | 2 | 240 |

[2] http://cs.joensuu.fi/sipu/datasets/



(a) Conjunto Jain (b) Conjunto R15 (c) Conjunto D31

(d) Conjunto Aggregation (e) Conjunto Flame (f) Conjunto Compound

(g) Conjunto Pathbased (h) Conjunto Spiral

**Fig. 1.** Base de datos *Shape* original.

## 4. Metodología

El algoritmo de clasificación propuesto fue implementado en una computadora Apple-MAC i27" [3], se utilizó Python 3.7 como lenguaje de simulación con los módulos requeridos tanto de redes neuronales como de graficación.

[3] 3.1GHz quad-core Intel Core i5 con 6MB on-chip shared L3 cache, 16GB de memoria, procesador gráfico AMD Radeon HD 6970M con 1GB de memoria.

**Función Principal(x,y, iteraciones)**
Comentario: $x$ atributos y $y$ etiquetas de archivo txt
**inicio**
  X ← matrix 'x' + una columna de $1's$
  Y ← transformación de las etiquetas 'y' a la representación "one-hot"
  k ← número de clases
  n ← número de atributos
  W ← valores aleatorios de W de tamaño $[n + 1 \times k]$
  Z ← hipotesis(X,W)
  **mientras** $m <$ *iteraciones* **hacer**
    W=gradiente(X,Y,Z,n,k)
    L ← costo(Z,Y)
    Z ← hipotesis(X,W)
  **fin**
  ypred ← max_por_fila_de$(W * X)$
**fin**

**Algoritmo 1:** Función Principal.

**Función hipotesis(X,W)**
Comentario: $X$ es la matriz de atributos y $W$ es la matriz de pesos
z ← X*W
s ← matriz de ceros de tamaño z
**mientras** $j <$ *números de observaciones* **hacer**
  Z ← z[j fila]
  s[j fila] ← exp(Z)/suma(exp(Z))
**fin**
Regresar s

**Algoritmo 2:** Función hipótesis.

**Función costo(Z,Y)**
Comentario: $Z$ es la matriz de probabilidades y $Y$ es la matriz de etiquetas
suma ← 0
**mientras** $j <$ *números de observaciones* **hacer**
  Z ← z[j fila]
  J ← producto_punto(Z[j fila],Y[j fila])
  J ← log(J)
  suma ← suma+J
**fin**
Regresar suma$*(-1/$número de observaciones$)$

**Algoritmo 3:** Función costo.

El seudocódigo del programa principal se muestra en el Algoritmo 1 y las funciones invocadas son tres: *Función de hipótesis* (ver Algoritmo 2), *Función de costo* (ver Algoritmo 3) y la *Función de gradiente* (ver Algoritmo 4). Se usó la base de datos *Shape* descrita en el capítulo anterior, la cual cuanta con 8 tipo de distribuciones-densidades.

**Función gradiente(X,Y,Z,n,k)**
Comentario: $Z$ es la matriz de probabilidades y $Y$ es la matriz de etiquetas
**mientras** $j <$ *números de observaciones* **hacer**
  suma ← matriz de 1's de tamaño 1x(n+1)
  alfa ← 0.01
  **mientras** $i <$ *números de observaciones* **hacer**
    suma ← suma + (Y[i,j]-Z[i,j])*X[i fila]
  **fin**
  W[j columna] ← W[j columna] - (-alfa/número de observaciones)∗suma
**fin**
Regresar W

**Algoritmo 4:** Función gradiente.

Dado que el algoritmo es iterativo, se tomó como criterio de *paro* el error, es decir, cuando el error encuentra su valor mínimo, para cada una de las bases de datos el número de iteraciones es variable.

## 5. Resultados experimentales

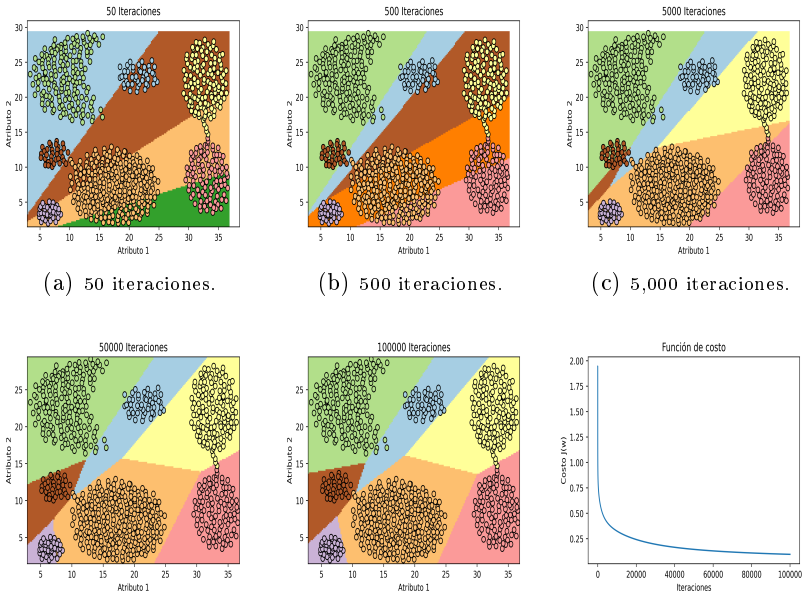
Se ha llevado a cabo la segmentación-clasificación con el algoritmo propuesto usando la base de atributos bi-dimensional *shape*. Los ocho conjuntos sintéticos fueron clasificados dando una medida del error ($\epsilon$) para cada grupo. La ecuación 9 permite expresar un cociente entre el número de puntos clasificados incorrectamente, con respecto al número total de puntos de cada clase:

$$\epsilon = \frac{\text{Clasificación incorrecta}}{\text{Total elementos por clase}}. \tag{9}$$

### 5.1. Evolución de clasificación

A manera de ejemplo, presentamos la evolución del espacio de partición en dos dimensiones, para el conjunto de datos *Aggregation*. La base está constituida de 7 clases con 788 puntos muestra por clase. En la Fig. 2 podemos apreciar la evolución para 50, 500, 5,000, 50,000 y 100,000 iteraciones, respectivamente. Así mismo, la Fig. 2(f) presenta la evolución del error en función del número de iteraciones.

Como se puede apreciar en la Fig. 2(e), el espacio bidimensional pudo ser dividido en 7 zonas independientes, zonas: verde, azul, amarilla, naranja, café, magenta y rosa). Cada zona logró abarcada los puntos correspondientes a cada una de las clases de forma exitosa, obteniendo una precisión del 98.98 % (ver primera línea de la Tabla 2). Cabe recordar que el sistema de clasificación está basado en la función *Softmax*, la cual trabaja eficientemente en espacios linealmente separables.




(a) 50 iteraciones. (b) 500 iteraciones. (c) 5,000 iteraciones.

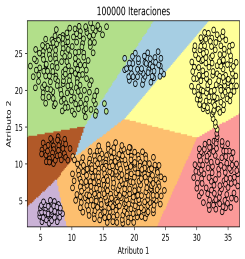
(d) 50,000 iteraciones. (e) 1000,000 iteraciones. (f) Error en aprendizaje

**Fig. 2.** Evolución del espacio de división del clasificador Softmax sobre la base de datos *Aggregation*.

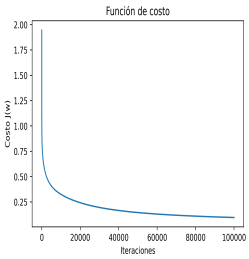
### 5.2. Evolución de desempeño sobre la base "Shape"

Las ocho bases de atributos de la base *Shape*, fueron clasificadas por el sistema de clasificación propuesto. Se analizaron tres componentes: el error de aprendizaje, el espacio bi-dimensional de partición y la matriz de confusión. Las Figs. 3 y 4 muestran los resultados para las 8 bases de atributos. Los resultados cuantificables del desempeño se muestran en la Tabla 2. El performance se obtuvo como el promedio de la diagonal principal de la matriz de confusión. Las bases de atributos se agruparon en tres categorías, de acuerdo con su forma de distribución y su complejidad en: *"Fáciles"*, *"Medios"* y *"Complejos"*.

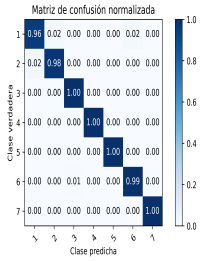
- **Fáciles:** El sistema propuesto funciona bien sobre los conjuntos *Aggregation*, *R15* y *Jain*, con desempeños de 98.98 %, 94.50 % y 90.08 %, respectivamente (ver Tabla 2).
- **Medios:** Sobre las bases *Flame*, *Compound* y *R31*, los resultados son buenos con desempeños promedio de 88.36 %, 77.94 % y 76.87 % y 65.66 %, respectivamente.
- **Complejos:** Para éste tercer grupo, el más complejo, los resultados con la metodología propuesta, son los siguiente: Para *Pathbased* y *Spiral* los resultados de desempeño son 65.66 % y 32.69 %, respectivamente. Los valores de desempeño son muy bajos debido a que son distribuciones linealmente no separables.



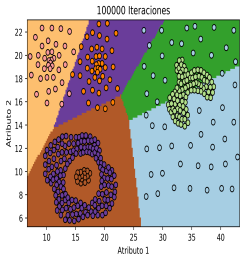

(a) Conjunto Aggregation: resultado de clasificación



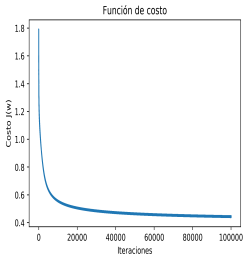

(b) Conjunto Aggregation: error en aprendizaje



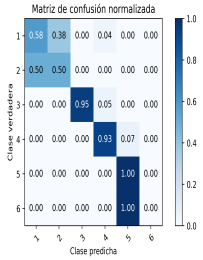

(c) Conjunto Aggregation: matriz de confusión



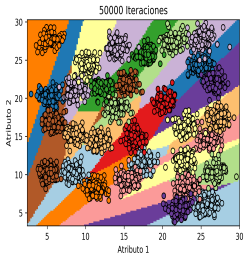

(d) Conjunto Compound: resultado de clasificación



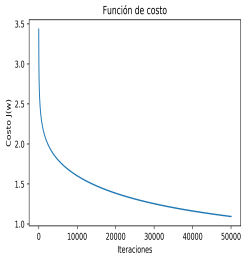

(e) Conjunto Compound: error en aprendizaje



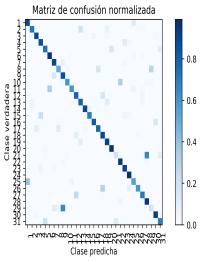

(f) Conjunto Compound: matriz de confusión



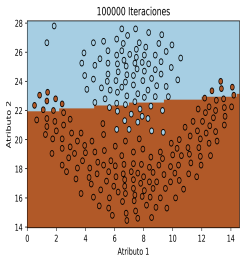

(g) Conjunto D31: resultado de clasificación



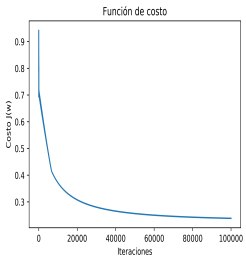

(h) Conjunto D31: error en aprendizaje



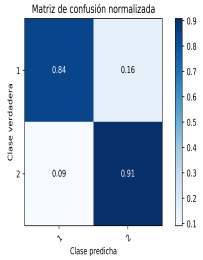

(i) Conjunto D31: matriz de confusión




(j) Conjunto Flame: resultado de clasificación

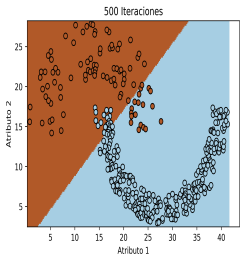



(k) Conjunto Flame: error en aprendizaje

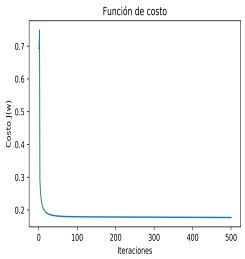



(l) Conjunto Flame: matriz de confusión

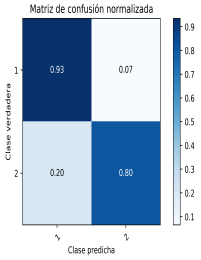
**Fig. 3.** Clasificación del sub conjunto "simple" de la base de datos *Shape*.



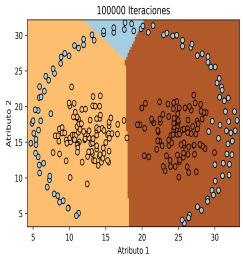

(a) Conjunto Jain: resultado de clasificación



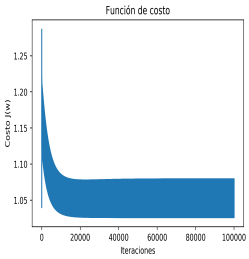

(b) Conjunto Jain: error en aprendizaje



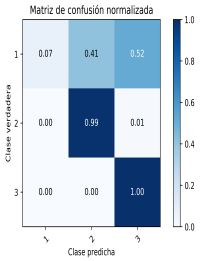

(c) Conjunto Jain: matriz de confusión



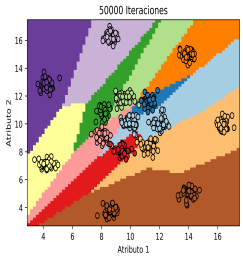

(d) Conjunto Pathbased: resultado de clasificación



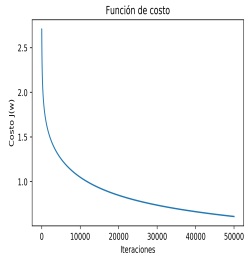

(e) Conjunto Pathbased: error en aprendizaje



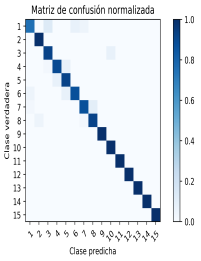

(f) Conjunto Pathbased: matriz de confusión



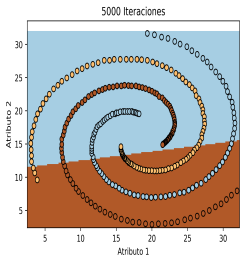

(g) Conjunto R15: resultado de clasificación



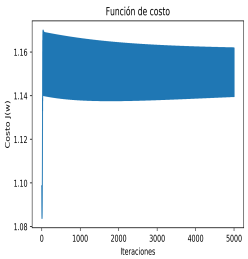

(h) Conjunto R15: error en aprendizaje



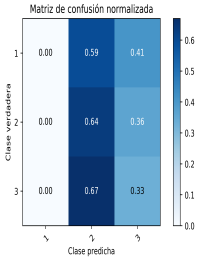

(i) Conjunto R15: matriz de confusión




(j) Conjunto Spiral: resultado de clasificación




(k) Conjunto Spiral: error en aprendizaje




(l) Conjunto Spiral: matriz de confusión

**Fig. 4.** Clasificación del sub conjunto "complejo" de la base de datos *Shape*.

**Tabla 2.** Clasificación de las 8 bases de datos sintéticas *Shape*.

| Conjunto | Precisión promedio |
|---|---|
| *Aggregation* | 98.98 % |
| *Compound* | 77.94 % |
| *Pathbase* | 65.66 % |
| *Spiral* | 32.69 % |
| *D31* | 76.87 % |
| *R15* | 94.50 % |
| *Jain* | 90.08 % |
| *Flame* | 88.36 % |

## 6. Conclusiones

Se ha podido diseñar e implementar un clasificador basado en la función *Softmax* de manera exitosa. El sistema propuesto es capaz de desplegar $n$ clasificadores *Softmax*, que como es bien sabido, es un clasificador lineal. A partir de subdivisiones del espacio de clasificación se logró, linealmente clasificar la base de datos *Shape*.

Las bases de atributos se agruparon en tres categorías, de acuerdo con su forma de distribución y su complejidad en: *Fáciles*, *Medios* y *Complejos*.

- **Fáciles: *Aggregation*, *R15* y *Jain***, para los cuales el sistema propuesto funciona excelentemente (desempeños entre el 90 % y 99 %)
- **Medios: *Flame*, *Compound* y *R31***, para este tipo de distribuciones los resultados son buenos, con desempeños entre 66 % y 88 %.
- **Complejos: *Pathbased* y *Spiral*** Para este tercer grupo, los desempeños son muy bajos (en los $30s$ %) debido a que son distribuciones linealmente no separables.

Finalmente, podemos afirmar que el sistema propuesta funciona muy bien cuando se trata de distribuciones linealmente separables.

Como trabajo futuro se probará el algoritmo propuesto a $n$ dimensiones de los atributos (más de 2 dimensiones). Adicionalmente se trabajará con más bases de datos utilizadas (*benchmark*) en el campo de clasificación de atributos.

## Referencias

1. Abavisani, M., Patel, V.: Deep multimodal subspace clustering networks. IEEE Journal on Selected Topics in Signal Processing 12(6), 1601–1614 (2018)
2. Alswaitti, M., Ishak, M., Isa, N.: Optimized gravitational-based data clustering algorithm. Engineering Applications of Artificial Intelligence 73, 126–148 (2018)
3. Behzadi, S., Ibrahim, M.A., Plant, C.: Parameter free mixed-type density-based clustering. In: Hartmann, S., Ma, H., Hameurlain, A., Pernul, G., Wagner, R.R. (eds.) Database and Expert Systems Applications. pp. 19–34. Springer International Publishing, Cham (2018)

4. Cao, X., Su, T., Wang, P., Wang, G., Lv, Z., Li, X.: An optimized chameleon algorithm based on local features. In: Proceedings of the 2018 10th International Conference on Machine Learning and Computing. pp. 184–192. ICMLC 2018, ACM, New York, NY, USA (2018)
5. Chen, J., Lin, X., Xuan, Q., Xiang, Y.: Fgch: a fast and grid based clustering algorithm for hybrid data stream. Applied Intelligence 49(4), 1228–1244 (Apr 2019)
6. Deng, C., Song, J., Sun, R., Cai, S., Shi, Y.: Gridwave: a grid-based clustering algorithm for market transaction data based on spatial-temporal density-waves and synchronization. Multimedia Tools and Applications 77(22), 29623–29637 (Nov 2018)
7. Deng, Z., Qian, G., Chen, Z., Su, H.: Identifying tor anonymous traffic based on gravitational clustering analysis. In: 2017 9th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC). vol. 2, pp. 79–83 (Aug 2017)
8. Dong, X., Yu, Z., Cao, W., Shi, Y., Ma, Q.: A survey on ensemble learning. Frontiers of Computer Science 14(2), 241–258 (2020)
9. Gupta, T., Panda, S.P.: A comparison of k-means clustering algorithm and clara clustering algorithm on iris dataset. International Journal of Engineering & Technology 7(4), 4766–4768 (2019)
10. Kanika, Rani, K., Sangeeta, Preeti: Visual analytics for comparing the impact of outliers in k-means and k-medoids algorithm. In: 2019 Amity International Conference on Artificial Intelligence (AICAI). pp. 93–97 (Feb 2019)
11. Kimura, M.: Autoclustering: A feed-forward neural network based clustering algorithm. In: Proceedings of the IEEE International Conference on Data Mining Workshops, ICDMW. vol. 2018-November, pp. 659–666 (2019)
12. Kumar, K.M., Reddy, A.R.M.: A fast dbscan clustering algorithm by accelerating neighbor searching using groups method. Pattern Recognition 58, 39 – 48 (2016)
13. Li, Y., Cai, J., Yang, H., Zhang, J., Zhao, X.: A novel algorithm for initial cluster center selection. IEEE Access 7, 74683–74693 (2019)
14. Matioli, L.C., Santos, S., Kleina, M., Leite, E.A.: A new algorithm for clustering based on kernel density estimation. Journal of Applied Statistics 45(2), 347–366 (2018)
15. Nemade, V., Shastri, A., Ahuja, K., Tiwari, A.: Scaled and projected spectral clustering with vector quantization for handling big data. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 2174–2179 (Nov 2018)
16. Nemade, V., Shastri, A., Ahuja, K., Tiwari, A.: Scaled and projected spectral clustering with vector quantization for handling big data. pp. 2174–2179 (2019)
17. Pitolli, G., Aniello, L., Laurenza, G., Querzoni, L., Baldoni, R.: Malware family identification with birch clustering. In: 2017 International Carnahan Conference on Security Technology (ICCST). pp. 1–6 (Oct 2017)
18. Ren, Z., Chen, J., Ye, L., Wang, C., Liu, Y., Zhou, W.: Application of rbf neural network optimized based on k-means cluster algorithm in fault diagnosis. In: 2018 21st International Conference on Electrical Machines and Systems (ICEMS). pp. 2492–2496 (Oct 2018)
19. Sheshasaayee, A., Sridevi, D.: Fuzzy c-means algorithm with gravitational search algorithm in spatial data mining. In: 2016 International Conference on Inventive Computation Technologies (ICICT). vol. 1, pp. 1–5 (Aug 2016)
20. Tian, L., Du, Q., Kopriva, I., Younan, N.: Spatial-spectral based multi-view low-rank sparse sbuspace clustering for hyperspectral imagery. In: IGARSS 2018 - 2018 IEEE International Geoscience and Remote Sensing Symposium. pp. 8488–8491 (July 2018)

21. Yang, Y., Zhu, Z.: A fast and efficient grid-based k-means++ clustering algorithm for large-scale datasets. In: Krömer, P., Zhang, H., Liang, Y., Pan, J.S. (eds.) Proceedings of the Fifth Euro-China Conference on Intelligent Data Analysis and Applications. pp. 508–515. Springer International Publishing, Cham (2019)
22. Yao, W., Dumitru, C., Loffeld, O., Datcu, M.: Semi-supervised hierarchical clustering for semantic sar image annotation. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 9(5), 1993–2008 (2016)
23. Zhao, Y., Tarus, S., Yang, L., Sun, J., Ge, Y., Wang, J.: Privacy-preserving clustering for big data in cyber-physical-social systems: Survey and perspectives. Information Sciences 515, 132–155 (2020)