INSTITUTO POLITÉCNICO NACIONAL

UNIDAD PROFESIONAL INTERDISCIPLINARIA EN INGENIERÍA Y TECNOLOGÍAS AVANZADAS

PRÁCTICA 1

Inteligencia Artificial

Alumnos:

Domínguez González Luis Antonio

Munguía Gallegos Mauricio

Profesor:

Álvaro Anzueto Ríos

Grupo:

3BM8

Fecha de entrega: 29 de Enero del 2020

Marco teórico

Sistema adaptativo neurodifuso para segmentación automática de imágenes y detección de bordes.

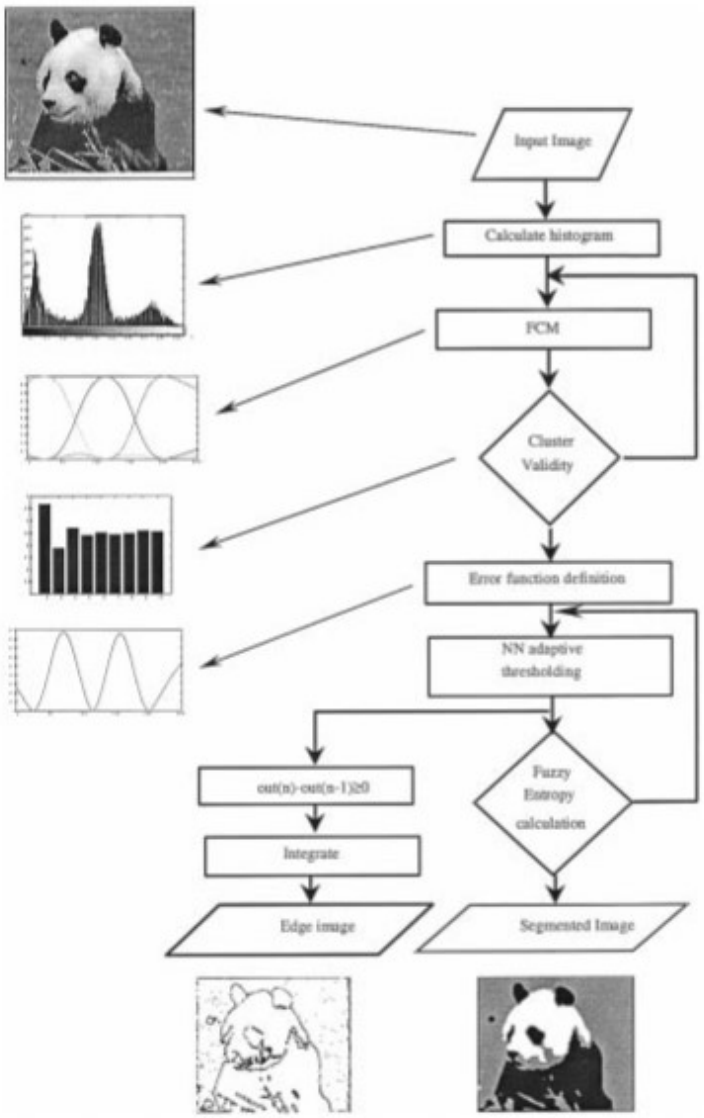

Figura 1 proceso de un sistema adaptativo neurodifuso para segmentación automática de imágenes y detección de bordes

$$PE = -\frac{1}{nln(\frac{1}{C})}\sum_{k=1}^{n}\sum_{i=1}^{C}[\mu_{ik}\ln(\mu_{ik})]$$

Se ocupa la ecuación como validity cluster de entropía después de haber pasado por todo el proceso que se muestra en la figura 1.

Algoritmo K-means

"Clustering algorithms" pueden ser aplicados para resolver problemas de segmentación. Consisten en elegir un píxel o región inicial que pertenezca a un objeto de interés, seguido de un proceso interactivo de análisis de vecindades, que decide si cada píxel vecino pertenece o no al mismo objeto. Se utiliza k-means para resolver la detección de masa en mamografías utilizando la información de textura obtenida de los descriptores de Haralick. El algoritmo K-means es uno de los algoritmos de aprendizaje no supervisados más simples. El método sigue los pasos habituales para satisfacer el objetivo principal: agrupar todos los objetos de imagen en K grupos distintos. Primero, se definen K centroides, uno para cada grupo, siendo su posición inicial muy importante para el resultado. Después de eso, se determina una región de propiedad para cada centroide, que agrupa un conjunto de objetos similares. Se inicia la etapa interactiva del algoritmo, en la que se recalcula el centroide de cada grupo para minimizar la función objetivo. Esta función, para K-means, es el método de mínimos cuadrados que se calcula con:

$$J = \sum_{j=1}^{k} \sum_{i=1}^{n} ||x_i^{(j)} - c_j||^2$$

Dónde $||x_i^{(j)} - c_j||^2$ es la distancia métrica desde cualquier punto $x_i^{(j)}$ en el grupo centroide c_j. Así, la J (función objetiva) representa la medida de similitud de los objetos contenidos en sus respectivos grupos [2].

Silhouette Coefficient

Permite que $D_M = \{\overline{D_1}, \dots, \overline{D_k}\}$ describa un resultado de agrupamiento, es decir, es una partición exhaustiva del conjunto de documentos D^*. La distancia de un documento d∈D^*a un agrupamiento $\overline{D_l} \in D_M$ está dada como

$$dist(d, D_i) = \frac{\sum_{P \in \overline{D_l}} dist(d, p)}{|\overline{D_l}|}$$

Permitiendo más adelante ser a$(d, D_i) = dist(d, D_i)$ la distancia del documento de a su agrupamiento $\overline{D_l}(d \in \overline{D_l})$, y a$(d, D_M) = min_{\overline{D_l} \in D_M, d \notin \overline{D_l}} dist(d, \overline{D_l})$ la distancia del documento d al agrupamiento de vecindad más cercano.

La silueta ϑ(d,D_M) de un documento d está definido cómo

$$s(d, D_M) = \frac{b(d, D_M) - a(d, D_M)}{\max\{a(d, D_M), b(d, D_M)\}}$$

El coeficiente de silueta como

$$SC(D_M) = \frac{\sum_{P \in D^*} s(p, D_M)}{|D^*|}$$

El coeficiente de silueta es una medida de la calidad de agrupamiento, que es bastante independiente de el número de grupos, k. Experiencias, como las documentadas en (Kaufman y Rousseeuw, 1990), muestran que valores entre 0.7 y 1.0 indican resultados de agrupamiento con una excelente separación entre agrupamientos, *viz*. los puntos de datos están muy cerca del centro de su grupo y lejos del siguiente grupo más cercano.Para el rango de 0.5 a 0.7 se encuentra que los puntos de datos están claramente asignados a los centros de agrupamiento. Valores de 0.25 a 0.5 indican que se pueden encontrar centros de agrupamiento, aunque hay un considerable "ruido", es decir hay muchos puntos de datos que no se pueden asignar claramente a los agrupamientps. Por debajo de un valor de 0.25 se vuelve prácticamente imposible encontrar centros de agrupación significativos y asignar definitivamente la mayoría de puntos de datos[3].

DESARROLLO

Se tiene que desarrollar un programa que indique para un número aleatorio de muestras cuántas clases son las más óptimas para separar todas las muestras usando lo conocido como “Validity Clusters”

Se probaron dos validity clusters siendo Silhouette Coefficient y por entropía los dos utilizados

A continuación se presenta el código donde se uso C-means para el validity cluster de entropía. Se hizo con C-means porque otorga valores basados en lógica difusa que son necesarios para las ecuaciones de entropía.

En el validity cluster de silueta se decidió usar k-means porque no son necesarios los datos basados en lógica difusa y el tiempo de computo era menor

```python
"""Practice 1: Compare Validity Clusters."""
import numpy as np
from scipy.stats import entropy
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from sklearn.metrics import silhouette_score

# Generating the sample data from make_blobs

X, Y = make_blobs(centers=4)

no_of_clusters = [2, 3, 4, 5, 6]

plt.figure(1)
plt.plot(X[:, 0], X[:, 1], 'bo')

"""C-means with Entropy."""
n, car = X.shape

max_data_number = max(max(X[:, 0], X[:, 1], key=lambda x: abs(x[1])))

for n_clusters in no_of_clusters:

    V = np.random.rand(n_clusters, car) * max_data_number
    D = np.zeros((n_clusters, 100))
    U = np.zeros((n_clusters, 100))
    epsilon = 1

    while epsilon > 10**(-10):
        Vold = np.array(V)
        for i in range(n_clusters):
            for k in range(n):
                a = 1**(-10)
                for j in range(car):
                    a = (X[k, j] - V[i, j])**(2) + a
                D[i, k] = a ** (1 / 2)

        for i in range(n_clusters):
            for k in range(n):
                b = 0
                for j in range(n_clusters):
                    b = (D[i, k] / D[j, k])**2 + b
                U[i, k] = 1 / b

        for i in range(n_clusters):
            for j in range(car):
                c = 0
                d = 0
                for k in range(n):
                    c = ((U[i, k]**2) * X[k, j]) + c
                    d = ((U[i, k]**2)) + d
                V[i, j] = c / d
        epsilon = sum(sum(abs(Vold[:] - V[:])))

    total_entropy = 0
```

```
57
58          for i in range(n_clusters):
59              total_entropy += entropy(U[i])
60
61          total_entropy *= -(1 / 100)
62
63          print("For no of clusters =", n_clusters,
64                " The entropy is :", total_entropy)
65
66          plt.figure('Entropy {}'.format(n_clusters))
67          plt.plot(X[:, 0], X[:, 1], 'bo')
68          plt.plot(V[:, 0], V[:, 1], 'r*')
69
70      """K-means with Silhoutte."""
71
72      for n_clusters in no_of_clusters:
73
74          kmeans = KMeans(n_clusters=n_clusters)
75          # Fitting with inputs
76          kmeans = kmeans.fit(X)
77          # Predicting the clusters
78          labels = kmeans.predict(X)
79          # Getting the cluster centers
80          centers = kmeans.cluster_centers_
81
82          plt.figure('Silhoutte {}'.format(n_clusters))
83          plt.plot(X[:, 0], X[:, 1], 'bo')
84          plt.plot(centers[:, 0], centers[:, 1], 'r*')
85
86          # The silhouette_score gives the average value for all the samples.
87          silhouette_avg = silhouette_score(X, labels)
88
89          print("For no of clusters =", n_clusters,
90                " The average silhouette_score is :", silhouette_avg)
91
92      plt.show()
```

CONCLUSIÓN

El método de entropía se vuelve tardado por el hecho de que se utiliza el algoritmo de c-means para el agrupamiento ya que como se ha mencionado las ecuaciones de entropía necesitan una matriz con valores basados en lógica difusa además de que el tiempo de computo cada vez se vuelve mayor cuando agregamos más clases ya que si queremos llegar el épsilon adecuado se tiene que ejecutar más veces el algoritmo para alcanzar el valor de épsilon adecuado en cambio con el método de la silueta el tiempo de ejecución fue notoriamente más rápido ya que se utilizó el algoritmo de k-means para el agrupamiento donde no se necesitaron valores de lógica difusa. Se pudo observar que utilizando el método de la silueta los valores que arrojaba hacían más sentido con las gráficas que muestra el programa. Con todo esto se puede concluir que el método de la silueta otorga mejor y más rápida validación sobre el número de clases a usar.

REFERENCIAS

[1] Boskovitz, V., & Guterman, “An adaptive neuro-fuzzy system for automatic image segmentation and edge detection”. *IEEE Transactions on fuzzy systems*, 2002, vol. 10, no 2, p. 247-262. [online], consultado el 29 de enero del 2020 en

https://ieeexplore.ieee.org/abstract/document/995125

[2] Martins,O.Braz,G. Aristófanes,S. Paiva, Anselmo. Gattass, Marcelo. “Detection of Masses in Digital Mammograms using K-means and Support Vector Machine”. *ELCVIA: electronic letters on computer vision and image analysis*, [online], 2009, Vol. 8, Num. 2, pp. 43, consultado el 29 de enero del 2020 en https://www.raco.cat/index.php/ELCVIA/article/view/150156 [View: 30-01-2020]

[3] Hotho, A., Maedche, A., & Staab, S. (2002). “Ontology-based text document clustering”. *KI*, *16*(4), 48-54.[online], consultado el 29 de enero del 2020 en https://www.kde.cs.uni-kassel.de/wp-content/uploads/benz/hotho/pub/Ontology_based_Text_Document_Clustering_2002.pdf