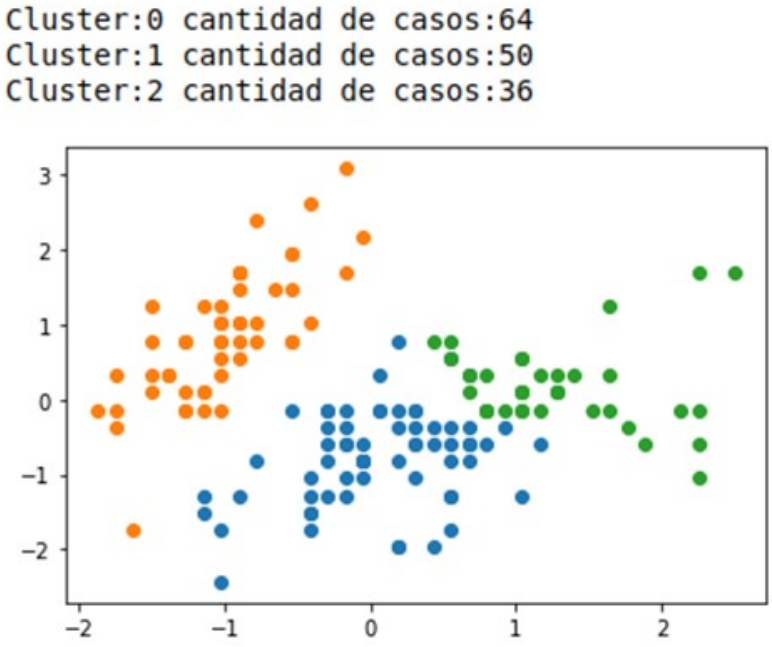

Figura 28. Resultados de la corrida del método BIRCH sobre el Iris Dataset.

Veremos ahora los parámetros que son necesarios para utilizar este modelo y la salida que nos arroja después del entrenamiento.

Parámetros:

- threshold: Este es el parámetro T que ya explicamos anteriormente y define el tamaño del árbol de forma conjunta con B. Este parámetro espera valores de tipo float y el valor defecto es 0.5.
- branching_factor: Este es el factor de ramificación y es el parámetro B del que hablamos cuando describimos el método. Este determina el número máximo de CF en cada nodo. Este parámetro espera valores enteros y su valor por defecto es 50.
- n_clusters: Este parámetro espera valores enteros y especifica el número de clústers que en los que se desea agrupar el conjunto de datos. El valor por defecto es 3. Si se le pasa None, como valor, no se ejecuta la fase de clustering y los subclusters son retornados de la manera en que los agrupa el modelo. También se le puede

pasar como valor uno de los métodos contenidos en el paquete sklearn.cluster, en cuyo caso el modelo es ajustado tratando los subclusters como muestras nuevas y el dato inicial es mapeado a la etiqueta del subclúster más cercano. Cuando se le pasa un valor entero el modelo se ajusta usando AgglomerativeClustering y n_clusters es igual al entero pasado.

Valores que retorna BIRCH:

- root_: Contiene el nodo raíz del árbol CF construido a partir del dataset.
- dummy_leaf_: Contiene el puntero inicial que permite recorrer todos los nodos hojas del árbol.
- subcluster_centers_: Contiene los centroides de todos los subclusters.
- labels_: Contiene las etiquetas asignadas a cada uno de los casos del dataset.

Claro que estos no son todos los parámetros y los valores que retorna la implementación de este método en scikit-learn, pero como siempre comento, para más detalles es necesario consultar la documentación.

Hasta aquí lo correspondiente al método BIRCH, aunque es necesario que destaquemos que solo hemos tocado la idea central del método y que este requiere un estudio mucho más profundo y la literatura sobre él no es del todo buena. Por eso sugerimos la lectura del artículo original: BIRCH: An Efficient Data Clustering Method for Very Large Databases, de los autores: Tian Zhang; Raghu Ramakrishnan y Miron Livny. Este artículo está disponible de forma gratuita en Internet.

Conclusiones

Hasta aquí hemos estado tratando lo concerniente a los métodos de aprendizaje no supervisado, está claro que no podemos ni siquiera imaginar el tema agotado y hemos solo mostrado las ideas básicas que soportan algunos de los métodos. La idea es, como con los métodos supervisados, abrir las puertas al mundo del

aprendizaje automático mostrando el funcionamiento de algunos de sus métodos; pero es un campo basto y complejo y que requerirá de mucho, mucho esfuerzo por dominar.

En esta lección, sin embargo, no hemos tocado lo concerniente a la evaluación de los métodos no supervisados pues lo haremos en la próxima lección, a diferencia de los métodos supervisados que lo hicimos al inicio de la primera.

Tema 4. Algoritmos de aprendizaje automático no supervisado

4.2. Algoritmos de aprendizaje automático no supervisado II

En la lección anterior vimos lo concerniente a los modelos de aprendizaje no supervisado, sin embargo, no tocamos los métodos de evaluar los resultados de su ajuste. El objetivo de esta, es describir y poner ejemplos de cómo se aplican estas medidas, así como abordar su clasificación.

Cuando termine esta lección, estará en condiciones de aplicar los métodos de aprendizaje no supervisado y evaluar los resultados.

En la lección anterior explicamos que son los métodos de aprendizaje no supervisado y revisamos las ideas que hay detrás de algunos de esos métodos. Sin embargo, no tocamos el tema de cómo evaluar sus resultados.

Evaluar los resultados de los métodos no supervisados es considerablemente más difícil que evaluar los resultados de los métodos supervisados, pues, normalmente no se cuenta con información acerca de la estructura de los datos. En los métodos no supervisados al no contar con información que nos diga cuán bien o mal lo estamos haciendo nos obliga a buscar tanto dentro de los mismos datos como fuera, algún criterio objetivo y mejor aún cuantitativo de la calidad del trabajo hecho por el método.

Otra de las causas que complican la evaluación de los métodos no supervisados es la subjetividad que es inherente a la actividad de agrupamiento de objetos o cosas.

La cantidad de métodos que se han desarrollado hablan de la complejidad del tema, sin embargo, de manera general, los métodos de evaluación de la validez de los resultados del agrupamiento arrojados por un método no supervisado se pueden agrupar en dos grandes grupos:

Tipos de criterios de validez de los clústers:

- Criterios Internos.
- Criterios Externos.

Cuando hablamos de Criterios Internos, hacemos referencia a un grupo de métricas que evalúan la validez de los agrupamientos usando solamente la información aportada por el propio agrupamiento.

Por el contrario, los criterios externos, usan información conocida a priori sobre el particionamiento o agrupamiento del conjunto de datos. Un ejemplo es el uso de conjuntos de datos usados para clasificación, donde se conoce de antemano a qué clase pertenece cada caso. Esta información puede provenir de anotadores humanos que son expertos en el campo. Este tipo de métrica puede usarse para evaluar y escoger métodos con los mejores resultados sobre un dataset determinado. Sin embargo, esta información, externa, no siempre está disponible.

Estás métricas, además de ayudarnos a evaluar la calidad de un agrupamiento basado, digamos en un conjunto de parámetros, también nos ayuda a escogerlos. Un caso típico es el uso de ellos para escoger el número óptimo de clústers.

La validación de Clustering, viene acompañada, básicamente por tres tareas: Evaluación del Clustering, que busca valorar la calidad de un agrupamiento. Estabilidad del clustering, que busca comprender la sensibilidad de un modelo a las variaciones de los distintos parámetros de los modelos y la Tendencia al clustering, que pretende evaluar lo adecuado de los datos para aplicar sobre ellos técnicas de agrupamiento, es decir, evaluar cuando los datos tienen de forma inherente una estructura de grupo.

A continuación, exploraremos algunos de los criterios más usados y aquellos implementados en el módulo scikit-learn de python. Esta decisión está basada en la

disponibilidad de documentación, la popularidad alcanzada por el framework y su facilidad de uso. Esto implica que puede comenzar a usarlos en sus proyectos de forma inmediata.

Criterios de validez Internos

Debido a la naturaleza misma de la tarea de agrupamiento, que trata de poner los casos más parecidos en grupos y que pretende que la distancia entre estos grupos sea máxima, se impone evaluar, básicamente, dos aspectos de cada agrupamiento: La cohesión de los grupos y la separación entre ellos.

La mayoría de las medidas o criterios de evaluación de los agrupamientos, miden alguno de estos dos aspectos.

La cohesión evalúa cuán parecidos o cercanos son los miembros de cada clúster, mientras que las medidas de separación permiten evaluar la distancia entre los grupos.

Una de las maneras más elementales de evaluar la cohesión y que aflora de manera natural cuando se razona sobre el tema es la Suma de Cuadrados Intra-clúster. (Sum of Squares Within)

$$SSW = \sum_{i=1}^{C} \sum_{C \vee dist^2(m_i, x_j)}^{} $$

Donde C es la cantidad de clúster, |C| es el número de puntos o casos en el clúster i, m_i es el centroide del clúster i y x_j es el caso j del clúster i.

Usando la misma idea se puede evaluar la separación:

$$SSB = \sum_{j=1}^{K} n_j \, dist^2 \, (c_j - \overline{x})$$

Donde, K es el número de clúster, n_j es el número de casos en el clúster j, c_j es el

centroide el clúster j y $\underline{x}$ es la media del dataset. A esta medida se le llama Suma cuadrada Inter-clúster. (SSB)

Es muy sencillo darse cuenta que los valores de estas medidas dependen del valor de n, es decir de la cantidad de casos en el dataset, lo que las hace difícil de interpretar cuando se comparan dos métodos.

Se han propuesto varios Índices que se basan en uno o los dos anteriores, que listamos abajo; pero que no todos los veremos en más detalles.

Algunos índices basados en SSW y SSB:

Ball y Hall (1965):

$$BH = \frac{SSW}{K}$$

Donde K es el número de clústers.

Calinski y Harabasz (1974):

$$CH = \frac{SSB\,/\,(k-1)}{SSW\,/\,(n-k)}$$

Donde K es el número de clústers y n el número de casos.

Hartigan (1975):

$$Har = \log\left(\frac{SSB}{SSW}\right)$$

Xu (1997):

$$xu = d * \log\left(\sqrt{\frac{SSW}{dn^2}}\right) + \log\,(k)$$

Donde d es la dimensión o cantidad de rasgos del dataset.

Veamos con más detalles algunas de estas medidas.

Calinski-Harabasz Index:

Este índice es la tasa entre la suma de la dispersión inter-clúster y la dispersión intra-clúster para todos los clústers. La dispersión se entiende aquí como la suma de la distancia cuadrada.

Por tanto, para un conjunto de datos E, con tamaño n_E el cual ha sido agrupado en k clústers, entonces:

$$s = \frac{\frac{tr(B_k)}{tr(W_k)} * n_E - k}{k-1}$$

Donde tr() es la función Trace. Y W_k y B_k son definidos como:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - C_q)(x - C_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

Una de las ventajas de usar este índice es su facilidad de interpretación, pues un valor alto de este implica clústers densos y bien separados. Sin embargo, este índice suele ser un poco más alto para clústers convexos.

En la siguiente tabla, mostramos los valores de este índice, para evaluar el resultado de la aplicación de varios métodos al Iris Dataset, usando la implementación de este en el módulo scikit-learn de python.

Método	Calinski-Harabasz Index
K-Means	241.90
Affinity Propagation	239.75
Mean Shift	152.47
Spectral Clustering	239.23
DBSCAN	84.26
BIRCH	237.65

En la siguiente imagen mostramos la manera en que se usa este índice de forma conjunta con el método de agrupamiento.

```
#Clustering con K-Means
from numpy import unique
from numpy import where
from sklearn.cluster import KMeans
from matplotlib import pyplot
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import calinski_harabasz_score
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
true_labels = idf.target
# Definimos el modelo, le pasamos como parámetro
# el número de clústers que queremos
model = KMeans(3)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
# Recuperamos los clusters
clusters = unique(yhat)
# Vamos a mostrar un Scatterplot para cada clúster
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(cluster,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    pyplot.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
pyplot.show()
# Calcula las métricas de validez
print("calinski_harabasz_score:",calinski_harabasz_score(X,model.labels_))
```

Fig. 1. Evaluando la validez del agrupamiento de un método K-Means sobre el Iris Dataset por medio del índice Calinski-Harabasz.

Este índice toma como parámetros el conjunto de datos y la asignación que ha hecho el método aplicado y retorna un valor de punto flotante.

Silhouette Coefficient

Este método de evaluación de la calidad del agrupamiento, combina la manera de evaluar la cohesión y la separación en un solo índice. Ve la cohesión como la

distancia promedio de x hasta todos los demás puntos en el mismo clúster y la separación como la distancia promedio de x a todos los puntos del clúster más cercano.

Matemáticamente el coeficiente Silhouette para un punto, puede expresarse como:

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

Dónde:

a: Es la distancia promedio entre el punto x y el resto de los puntos dentro del clúster.

b: Es la distancia promedio entre el punto x y todos los puntos del clúster más cercano.

El coeficiente para todo el agrupamiento es:

$$S = \frac{1}{N}\sum_{i=1}^{N} s\,(x)$$

Esta métrica toma valores entre -1 y 1, donde valores cercanos a -1 indican un agrupamiento incorrecto y valores cercanos a 1, clústers más densos. Valores cercanos a cero pueden indicar solapamiento en los clústers.

Como en la métrica anterior, mostramos a continuación una tabla con los valores del coeficiente para diferentes métodos de aprendizaje no supervisado que evalúa la calidad del agrupamiento sobre el Iris Dataset.

Método	Silhouette Coefficient
K-Means	0.46
Affinity Propagation	0.46
Mean Shift	0.40
Spectral Clustering	0.46
DBSCAN	0.31
BIRCH	0.46

En la siguiente imagen mostramos la manera de usar esta métrica junto a un método de aprendizaje automático no supervisado.

```
#Clustering DBSCAN
from sklearn.cluster import DBSCAN
import numpy as np
import matplotlib.pyplot as plt
from numpy import unique
from numpy import where
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
true_labels = idf.target
# Definimos el modelo, le pasamos como parámetro
# eps y min_samples
model = DBSCAN(eps=0.68, min_samples=15)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
# Recuperamos los clusters
clusters = unique(yhat)
# Vamos a mostrar un Scatterplot para cada clúster
i=0
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(i,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
    i+=1
# show the plot
plt.show()
# Calcula las métricas de validez
print("silhouette_score:",silhouette_score(X,model.labels_))
```

Fig. 2. Evaluando la validez del agrupamiento de un método DBSCAN sobre el Iris Dataset por medio del Coeficiente Silhouette.

La métrica silhouette_score, toma los siguientes parámetros:

- X: Representa el conjunto de datos al que se le aplica el agrupamiento.
- labels: el etiquetado que retorna el método de aprendizaje.
- metric: La métrica utilizada para el cálculo del índice. El valor por defecto es “euclidean”. Ahora bien, si X es una matriz de distancias entonces este parámetro debe tomar como valor ‘precomputed’.

Retorna un valor float.

Davies-Bouldin Index:

Este índice se basa en la idea de la similaridad promedio entre los clústers. Aquí la similaridad se entiende como una medida que compara la distancia entre clústers con el tamaño de los clústers en sí mismo.

Matemáticamente, este índice puede expresarse como sigue:

$$DB = \frac{1}{k}\sum_{i=1}^{k} R_{ij}$$

Dónde:

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

s_i : distancia media entre cada punto del clúster i y el centroide del clúster.

d_{ij} : distancia entre los centroides i y j.

En este índice los valores cercanos a cero son indicadores de una mejor partición. Por tanto, el agrupamiento que minimice el valor de este, se considera óptimo.

Veamos en la tabla, como se comporta a la hora de evaluar un agrupamiento sobre el Iris Dataset. Para eso usamos la implementación de scikit-learn de python.

Método	**Davies-Bouldin Index**
K-Means	0.83
Affinity Propagation	0.84
Mean Shift	0.79
Spectral Clustering	0.83
DBSCAN	7.71
BIRCH	0.84

Veamos cómo se usa de conjunto con uno de los métodos anteriores:

```python
#Clustering BIRCH
from sklearn.cluster import Birch
import numpy as np
import matplotlib.pyplot as plt
from numpy import unique
from numpy import where
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import davies_bouldin_score
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
true_labels=idf.target
# Definimos el modelo, le pasamos como parámetro
# B y T.  (B=branching_factor y T= threshold)
model = Birch(branching_factor=8, threshold=0.3)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
# Recuperamos los clusters
clusters = unique(yhat)
# Vamos a mostrar un scatterplot para cada clúster
i=0
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(i,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
    i+=1
# show the plot
plt.show()
# Calcula las métricas de validez
print("davies_bouldin_score:",davies_bouldin_score(X,model.labels_))
```

Fig. 3. Evaluando la validez del agrupamiento de un método BIRCH sobre el Iris Dataset por medio del Índice Davies-Bouldin.

davies_bouldin_score() toma como parámetros:

X: el conjunto de datos sobre el que se ejecuta el proceso de agrupamiento.

labels: el etiquetado producido por la aplicación del método.

Retorna un float.

Criterios de validez Externos

Lo que caracteriza a este tipo de criterio es el hecho de necesitar de un etiquetado de referencia externo, ya sea aportado por el conjunto de datos o por expertos humanos en el campo.

Rand Index.

Este es un índice que pertenece al conjunto de los criterios externos. Y la idea detrás

de este, es medir la similaridad entre las etiquetas que el método de aprendizaje asignó con aquellas que son conocidas de antemano. En realidad, se puede usar para medir la similaridad entre los resultados de dos agrupamientos.

Supongamos un conjunto S de n elementos tal que:

$$S = \{s_0, s_1, \ldots, s_n\}$$

Y tengamos dos particiones de S, X en r subconjuntos y Y en s subconjuntos tal que:

$$X = \{X_1, X_2, \ldots, X_r\}$$

Y

$$Y = \{Y_1, Y_2, \ldots, Y_s\}$$

Entonces podemos definir el Rand Index o Índice de Rand de la forma siguiente:

$$RI = \frac{a+b}{n(n-1)/2}$$

Dónde:

- a es el número de pares de elementos en S que están en el mismo subconjunto en X y en el mismo subconjunto en Y.
- b es el número de pares de elementos en S que están en subconjuntos diferentes en X y en subconjuntos diferentes en Y.
- $n(n-1)/2$ es el número total de pares en S.

Este índice toma valores entre 0 y 1, indicando el cero que no hay acuerdo entre los dos agrupamientos, mientras que 1 es el perfecto acuerdo.

Sin embargo, este índice no asegura que para un etiquetado aleatorio el valor sea exactamente cero, y por esto se usa el índice de Rand ajustado. Adjusted Rand Index.

En la siguiente tabla, mostramos los resultados arrojados de este índice y el ajustado, usado para evaluar varios métodos de aprendizaje no supervisado. Para elaborar la tabla usamos la implementación de este índice en scikit-learn de python y las medidas se tomaron sobre el Iris dataset.

Método	RI Ajustado	RI no Ajustado
K-Means	0.63	0.84
Affinity Propagation	0.63	0.84
Mean Shift	0.60	0.83
Spectral Clustering	0.58	0.81
DBSCAN	0.41	0.74
BIRCH	0.61	0.83

En la siguiente imagen mostramos un ejemplo de cómo usarlo en scikit-learn.

```
#Clustering BIRCH
from sklearn.cluster import Birch
import numpy as np
import matplotlib.pyplot as plt
from numpy import unique
from numpy import where
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import rand_score
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
true_labels=idf.target
# Definimos el modelo, le pasamos como parámetro
# B y T.  (B=branching_factor y T= threshold)
model = Birch(branching_factor=8, threshold=0.3)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
Labels = model.labels_
# Recuperamos los clusters
clusters = unique(yhat)
# Vamos a mostrar un Scatterplot para cada clúster
i=0
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(i,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
    i+=1
# show the plot
plt.show()
# Calcula el índice de Rand
Rand = rand_score(true_labels, Labels)
print("Rand Index:",Rand)
```

Fig. 4. Evaluando la validez del agrupamiento de un método BIRCH sobre el Iris Dataset por medio del índice de Rand.

Es importante notar que, en esta implementación, el índice de Rand nos permite

comparar el etiquetado conocido del Iris dataset con el etiquetado resultado del ajuste del modelo BIRCH. Esta es precisamente, una de las desventajas de usar este tipo de índice, pues esta información no siempre está disponible.

Medidas basadas en Información Mutua

La información mutua trata de medir el acuerdo entre dos agrupamientos ignorando las permutaciones. En otras palabras, esta medida trata de medir la cantidad de información compartida entre el agrupamiento y un particionamiento y es definida como sigue:

Dado un conjunto S de N elementos, tal que:

$$S = \{S_1, S_2, \ldots, S_N\}$$

Y dos particiones en este, que llamaremos X y Y tal que estas sean:

$$X = \{X_1, X_2, \ldots, X_r\}$$

$$Y = \{Y_1, Y_2, \ldots, Y_s\}$$

Suponiendo que: $X_i \cap X_j = \emptyset \forall i \neq j$ y que $Y_i \cap Y_j = \emptyset \forall i \neq j$, es decir que los clústers son duros y que además son completos, es decir:

$$\bigcup_{i=1}^{r} X_i = \bigcup_{i=1}^{S} Y_i = S$$

Entonces, la Información mutua entre X y Y vendría dada por:

$$MI(X,Y) = \sum_{i=1}^{r} \sum_{j=1}^{s} P_{XY}(i,j) * \log \frac{P_{XY}(i,j)}{P_X(i)\, P_Y(j)}$$

Donde la probabilidad $P_{XY}(i,j)$ es la probabilidad de que un caso o punto pertenezca tanto a un clúster X_i de X como a un clúster Y_i de Y, se puede calcular de la siguiente forma:

$$P_{XY}(i,j) = \frac{|X_i \cap Y_j|}{N}$$

La información mutua, expresada como cardinalidad de conjuntos puede verse de la siguiente manera:

$$MI(X,Y) = \sum_{i=1}^{|X|} \sum_{j=1}^{|Y|} \frac{|X_i \cap Y_j|}{N} * \log\left(\frac{N\,|X_i \cap Y_j|}{|X_i|\,|Y_j|}\right)$$

Dónde:

$|X|$ y $|Y|$ son las cantidades de clúster que contiene cada una de las respectivas particiones.

$|X_i|$ y $|Y_j|$ son las cantidades de puntos o casos que contiene el clúster i y j de cada partición.

N es el número de casos o puntos en S.

La información mutua, cuando X y Y son independientes, toma el valor cero; pero no tiene una cota superior. Por eso, aparece una versión de esta, llamada Información Mutua Normalizada y se puede expresar como:

$$NMI(X,Y) = \frac{MI(X,Y)}{mean}$$

Donde H(X) es la entropía de X y H(Y) la entropía de Y, y pueden expresarse como:

$$H(X) = -\sum_{i=1}^{|X|} P(i) * \log\left(P(i)\right)$$

$$H(Y) = -\sum_{j=1}^{|Y|} P(j) * \log\left(P(j)\right)$$

La Información Mutua Ajustada, por su parte, puede expresarse de la siguiente manera:

$$AMI(X,Y) = \frac{MI(X,Y) - E[MI(X,Y)]}{mean(H(X)\,H(Y)) - E[MI(X,Y)]}$$

Donde $E[MI(X,Y)]$ es el valor esperado de la Información Mutua.

Como ya comentamos estas métricas de evaluación de la calidad del agrupamiento llamadas Información Mutua, toman valores entre 0 y 1, en la tabla siguiente hemos recopilado, como en el caso anterior una comparación de sus valores al aplicarla sobre el Iris Dataset usando varios métodos de aprendizaje no supervisado.

Método	MI	AMI	NMI
K-Means	0.72	0.65	0.66
Affinity Propagation	0.73	0.66	0.67
Mean Shift	0.75	0.63	0.64
Spectral Clustering	0.68	0.61	0.62
DBSCAN	0.53	0.48	0.49
BIRCH	0.72	0.66	0.67

MI: Información Mutua. (Mutual Information)

AMI: Información Mutua Ajustada. (Adjusted Mutual Information)

NMI: Información Mutua Normalizada. (Normalized Mutual Information)

En la siguiente imagen mostramos la manera de cómo calcular estos índices usando scikit-learn de python.

```python
#Clustering DBSCAN
from sklearn.cluster import DBSCAN
import numpy as np
import matplotlib.pyplot as plt
from numpy import unique
from numpy import where
from sklearn import datasets
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
from sklearn.metrics import mutual_info_score
from sklearn.metrics import adjusted_mutual_info_score
from sklearn.metrics import normalized_mutual_info_score
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
true_labels = idf.target
# Definimos el modelo, le pasamos como parámetro
# eps y min_samples
model = DBSCAN(eps=0.68, min_samples=15)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
# Recuperamos los clusters
clusters = unique(yhat)
# Vamos a mostrar un Scatterplot para cada clúster
i=0
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(i,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
    i+=1
# show the plot
plt.show()
MI5 = mutual_info_score(true_labels, model.labels_)
print("MI:",MI5)
AMI5 = adjusted_mutual_info_score(true_labels, model.labels_)
print("AMI:",AMI5)
NMI5 = normalized_mutual_info_score(true_labels, model.labels_)
print("NMI:",NMI5)
Silhuette = silhouette_score(X, model.labels_, metric='euclidean')
print("Coeficiente Silhuette:",Silhuette)
```

Fig. 5. Evaluando la validez del agrupamiento de un método DBSCAN sobre el Iris Dataset por medio de Información Mutua, Ajustada y Normalizada.

Estos índices, comparan el etiquetado verdadero (true_labels), por decirlo de alguna manera, con el asignado por el método de aprendizaje sin información alguna sobre la estructura de los datos. (model.labels_)

Homogeneidad, Completitud y V-Measure

Estas son otras medidas consideradas dentro del grupo de los criterios de evaluación externos y los tres están estrechamente ligados.

Con el propósito de explicar las ideas detrás de estas medidas de validez de un

agrupamiento, debemos hacer las mismas suposiciones que hemos estado haciendo en las medidas anteriores.

Dado un conjunto S de N elementos, tal que:

$$S = \{S_1, S_2, \ldots, S_N\}$$

Y dos particiones en este, que llamaremos X y Y tal que estas sean:

$$X = \{X_1, X_2, \ldots, X_r\}$$

$$Y = \{Y_1, Y_2, \ldots, Y_s\}$$

$$\bigcup_{i=1}^{r} X_i = \bigcup_{i=1}^{S} Y_i = S$$

Entonces podemos comenzar a definir lo que se entiende por Homogeneidad, Completitud y V-Measure.

La Homogeneidad y la Completitud de un agrupamiento dado como resultado de la aplicación de un método de aprendizaje no supervisado a un conjunto de datos son dos medidas complementarias que se combinan para crear V-measure,

Para explicar estos conceptos es necesario que aclaremos que llamaremos a cada $X_i \in X$ una clase y a cada $Y_i \in Y$ un clúster. Y veremos a X como una plantilla contra la cual compararemos cuán bien el método de aprendizaje asignó cada punto de datos de S a Y sin información ninguna de la estructura dada por X.

Un agrupamiento, digamos Y, satisface el criterio de Homogeneidad, si todos los clústers en Y contienen puntos de datos de un mismo X_i , es decir de una misma clase en X. Por el contrario, un agrupamiento Y satisface el criterio de Completitud si un clúster $Y_i \in Y$ tiene entre sus puntos elementos de todas las clases $X_i \in X$.

Esto implica que la homogeneidad de un clúster en mayor mientras menos diverso es, el caso ideal, es decir, donde se maximizará esta medida es en un clúster con elementos de una sola clase.

Por otro lado, la Completitud sería mayor mientras más diverso es el clúster, es decir mientras más elementos de este correspondas a más clases de $X_i \in X$.

Estas medidas son simétricas y por tanto se complementan de alguna manera.

La Homogeneidad puede expresarse de la siguiente forma:

$$h = 1 - \frac{H(X \vee Y)}{H(X)}$$

Y la Completitud, se expresaría como sigue:

$$c = 1 - \frac{H(Y \vee X)}{H(Y)}$$

Donde $H(X \vee Y)$ es la entropía condicional y puede expresarse de la siguiente forma:

$$H(Y) = -\sum_{i=1}^{|X|}\sum_{j=1}^{|Y|}\frac{n_{x,y}}{N} * \log\left(\frac{n_{x,y}}{N}\right)$$

Mientras que $H(X)$ es la entropía de la clase y se expresa como:

$$H(X) = -\sum_{i=1}^{|X|}\frac{n_x}{n} * \log\left(\frac{n_x}{n}\right)$$

Donde n_x es el número de casos en la clase o clúster i de X o de Y. Solo mostramos la entropía de X pero la de Y se calcula de la misma forma.

Por otra parte, la V-measure se expresa como la media armónica de estas dos medidas, y se expresa como:

$$v = \frac{2 * h * c}{h + c}$$

Sin embargo, una variante de esta, permite ponderar cada una de las medidas.

$$v = \frac{(1+\beta) * h * c}{(\beta * h + c)}$$

Cundo $\beta < 1$ se pondera la homogeneidad, es decir se le da mayor importancia. Cuando $\beta > 1$ se le da más importancia a la completitud.

Sin embargo, uno de los problemas de estas medidas es que no están normalizadas con respecto al etiquetado aleatorio e implica que el resultado para este tipo de etiquetado no siempre sería el mismo ya que dependen del número de muestras, la cantidad de clústers y clases. Este problema puede ignorarse de manera segura si la cantidad de casos es mayor que 100 y la cantidad de clúster menor que 10.

En la tabla siguiente veremos, como hemos hecho con las medidas anteriores, una comparación de sus valores con el propósito de evaluar el agrupamiento arrojado por varios métodos sobre el Iris Dataset.

Método	h	c	v-measure
K-Means	0.66	0.66	0.66
Affinity Propagation	**0.67**	0.67	**0.67**
Mean Shift	0.69	0.60	0.64
Spectral Clustering	0.62	0.62	0.62
DBSCAN	0.48	0.49	0.48
BIRCH	0.66	0.68	0.67

Una de las ventajas de este tipo de medidas es que no hace asunción alguna sobre la forma de los clústers, lo que la hace apta para comparar el resultado del agrupamiento arrojado por varios métodos.

Como hemos estado haciendo hasta ahora, veremos la manera en que se aplican estas medidas usando la implementación en scikit-learn de Python. En la siguiente imagen se muestra el código de la aplicación de un método Mean Shift sobre el Iris Dataset y se evalúa por medio de la Homogeneidad, Completitud y v-measure. Los valores son lo que se muestran en la tabla anterior.

```python
#Clustering con Mean Shift
from sklearn.cluster import MeanShift
from sklearn.cluster import  estimate_bandwidth
import matplotlib.pyplot as plt
import statistics as stat
from numpy import unique
from numpy import where
from sklearn import datasets
from itertools import cycle
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import homogeneity_score
from sklearn.metrics import completeness_score
from sklearn.metrics import v_measure_score
# Cargamos el Iris Dataset. Tres clases, 150 casos
# 50 casos por clase, todos los atributos numéricos
idf= datasets.load_iris()
sc = StandardScaler()
X = sc.fit_transform(idf.data)
true_labels = idf.target
# Definimos el modelo, le pasamos como parámetro
# bandwidth y bin_seeding
#bandwidth = estimate_bandwidth(X, quantile=0.2, n_samples=150)
model = MeanShift(bandwidth=0.99)
# Hacemos el clustering y retornamos el clúster
# asignado a cada caso
yhat = model.fit_predict(X)
Labels = model.labels_
# Recuperamos los clusters
clusters = unique(yhat)
# Vamos a mostrar un Scatterplot para cada clúster
for cluster in clusters:
    print("Cluster:{0} cantidad de casos:{1}".format(cluster,len(yhat[yhat==cluster])))
    row_ix = where(yhat == cluster)
    plt.scatter(X[row_ix, 0], X[row_ix, 1])
# show the plot
plt.show()
# Calcula las métricas de validez
print("Homogeneidad:",homogeneity_score(true_labels, model.labels_))
print("Completitud:",completeness_score(true_labels, model.labels_))
print("V-Measure",v_measure_score(true_labels, model.labels_))
```

Fig. 6. Evaluando la validez del agrupamiento de un método Mean Shift sobre el Iris Dataset por medio de Homogeneidad, Completitud y v-measure.

Índice Fowlkes-Mallows

Este es otro de los índices o criterios de validez externa, como en las anteriores medidas, para usar esta, se necesita conocer el verdadero etiquetado de los casos del conjunto de datos.

El índice de Fowlkes-Mallows, puede ser interpretada como la media geométrica de la Precisión y el Recall y puede expresarse matemáticamente como:

$$FMI = \frac{TP}{\sqrt{(TP+FP)(TP+FN)}}$$

Donde TP, es el valor de los Casos Positive (TP), es decir, el número de pares de puntos que pertenecen al mismo clúster tanto en el etiquetado verdadero y el

etiquetado producido por el método aplicado. FP, es el número de Falsos Positivos (FP), es decir el número de pares de puntos que pertenecen al mismo clúster en el etiquetado verdadero, pero no así en el etiquetado arrojado por el método de aprendizaje. FN, son los llamados Falsos Negativos (FN), que son los pares de puntos en el etiquetado producido por el método de aprendizaje, pero no en el etiquetado verdadero.

Este tipo de medida tiene un rango entre 0 y 1. Valores altos son indicativos de una alta similaridad entre las clases predichas y las reales.

Entre las ventajas de este método está que para un etiquetado aleatorio arroja valores cercanos a cero para cualquier cantidad de clústers y casos. Otra de las ventajas es su interpretabilidad, pues valores cercanos a cero indican que dos asignaciones o etiquetados son prácticamente independientes, mientras que valores cercanos a uno, indican un significativo acuerdo entre las dos. Además de un valor cero, indica dos etiquetados completamente diferentes o independientes y un valor exacto de uno dos etiquetados exactamente iguales.

Este índice, tampoco hace asunciones sobre la forma de los clústers y por tanto puede usarse para comparar diferentes métodos de aprendizaje automático.

Conclusiones

Hasta aquí hemos hecho un breve recorrido por algunos de los métodos de evaluación de los resultados de los métodos de aprendizaje no supervisado. Sin ánimo de cubrir todo el campo, pues de hecho es muy basto. Literalmente hay cientos de métodos y esto refleja la complejidad del asunto.

Es importante para entender a cabalidad este tópico que se busque información adicional y se profundice en el tema.

En alguna literatura, a los criterios de evaluación, internos y externos se les suma una categoría adicional llamada criterios relativos que no hemos tocado en esta

lección con vistas a no complicarla más de lo necesario.

Tampoco hemos tocado otras métricas que no están implementadas en scikit-learn, pues como meta tenemos que se puedan usar de forma inmediata los conocimientos que adquieran en estas lecciones. Sin embargo, se pueden encontrar en la literatura y la mayoría son muy fáciles de implementar.

Tema 4. Algoritmos de aprendizaje automático no supervisado

4.3. Aprendizaje automático

En la lección anterior vimos lo concerniente a los modelos de aprendizaje no supervisado, sin embargo, no tocamos los métodos de evaluar los resultados de su ajuste. El objetivo de esta, es describir y poner ejemplos de cómo se aplican estas medidas, así como abordar su clasificación.

Cuando termine esta lección, estará en condiciones de aplicar los métodos de aprendizaje no supervisado y evaluar los resultados.

Introducción

En la lección anterior explicamos que son los métodos de aprendizaje no supervisado y revisamos las ideas que hay detrás de algunos de esos métodos. Sin embargo, no tocamos el tema de cómo evaluar sus resultados.

Evaluar los resultados de los métodos no supervisados es considerablemente más difícil que evaluar los resultados de los métodos supervisados, pues, normalmente no se cuenta con información acerca de la estructura de los datos. En los métodos no supervisados al no contar con información que nos diga cuán bien o mal lo estamos haciendo nos obliga a buscar tanto dentro de los mismos datos como fuera, algún criterio objetivo y mejor aún cuantitativo de la calidad del trabajo hecho por el método.

Otra de las causas que complican la evaluación de los métodos no supervisados es la subjetividad que es inherente a la actividad de agrupamiento de objetos o cosas.

La cantidad de métodos que se han desarrollado hablan de la complejidad del tema, sin embargo, de manera general, los métodos de evaluación de la validez de los resultados del agrupamiento arrojados por un método no supervisado se pueden agrupar en dos grandes grupos:

Tipos de criterios de validez de los clústers:

- Criterios Internos.
- Criterios Externos.

Cuando hablamos de Criterios Internos, hacemos referencia a un grupo de métricas que evalúan la validez de los agrupamientos usando solamente la información aportada por el propio agrupamiento.

Por el contrario, los criterios externos, usan información conocida a priori sobre el particionamiento o agrupamiento del conjunto de datos. Un ejemplo es el uso de conjuntos de datos usados para clasificación, donde se conoce de antemano a qué clase pertenece cada caso. Esta información puede provenir de anotadores humanos que son expertos en el campo. Este tipo de métrica puede usarse para evaluar y escoger métodos con los mejores resultados sobre un dataset determinado. Sin embargo, esta información, externa, no siempre está disponible.

Estás métricas, además de ayudarnos a evaluar la calidad de un agrupamiento basado, digamos en un conjunto de parámetros, también nos ayuda a escogerlos. Un caso típico es el uso de ellos para escoger el número óptimo de clústers.

La validación de Clustering, viene acompañada, básicamente por tres tareas: Evaluación del Clustering, que busca valorar la calidad de un agrupamiento. Estabilidad del clustering, que busca comprender la sensibilidad de un modelo a las variaciones de los distintos parámetros de los modelos y la Tendencia al clustering, que pretende evaluar lo adecuado de los datos para aplicar sobre ellos técnicas de agrupamiento, es decir, evaluar cuando los datos tienen de forma inherente una estructura de grupo.

A continuación, exploraremos algunos de los criterios más usados y aquellos implementados en el módulo scikit-learn de python. Esta decisión está basada en la

disponibilidad de documentación, la popularidad alcanzada por el framework y su facilidad de uso. Esto implica que puede comenzar a usarlos en sus proyectos de forma inmediata.

Criterios de validez Internos

Debido a la naturaleza misma de la tarea de agrupamiento, que trata de poner los casos más parecidos en grupos y que pretende que la distancia entre estos grupos sea máxima, se impone evaluar, básicamente, dos aspectos de cada agrupamiento: La cohesión de los grupos y la separación entre ellos.

La mayoría de las medidas o criterios de evaluación de los agrupamientos, miden alguno de estos dos aspectos.

La cohesión evalúa cuán parecidos o cercanos son los miembros de cada clúster, mientras que las medidas de separación permiten evaluar la distancia entre los grupos.

Una de las maneras más elementales de evaluar la cohesión y que aflora de manera natural cuando se razona sobre el tema es la Suma de Cuadrados Intra-clúster. (Sum of Squares Within)

$$SSW = \sum_{i=1}^{C} \sum_{C \vee dist^2(m_i, x_j)}^{} $$

Donde C es la cantidad de clúster, |C| es el número de puntos o casos en el clúster i, m_i es el centroide del clúster i y x_j es el caso j del clúster i.

Usando la misma idea se puede evaluar la separación:

$$SSB = \sum_{j=1}^{K} n_j \, dist^2 \, (c_j - \overline{x})$$

Donde, K es el número de clúster, n_j es el número de casos en el clúster j, c_j es el centroide el clúster j y $\underline{x}$ es la media del dataset. A esta medida se le llama Suma cuadrada Inter-clúster. (SSB)

Es muy sencillo darse cuenta que los valores de estas medidas dependen del valor de n, es decir de la cantidad de casos en el dataset, lo que las hace difícil de interpretar cuando se comparan dos métodos.

Se han propuesto varios Índices que se basan en uno o los dos anteriores, que listamos abajo; pero que no todos los veremos en más detalles.

Algunos índices basados en SSW y SSB:

Ball y Hall (1965):

$$BH = \frac{SSW}{K}$$

Donde K es el número de clústers.

Calinski y Harabasz (1974):

$$CH = \frac{SSB/(k-1)}{SSW/(n-k)}$$

Donde K es el número de clústers y n el número de casos.

Hartigan (1975):

$$Har = \log\left(\frac{SSB}{SSW}\right)$$

Xu (1997):

$$xu = d * \log\left(\sqrt{\frac{SSW}{dn^2}}\right) + \log(k)$$

Donde d es la dimensión o cantidad de rasgos del dataset.

Veamos con más detalles algunas de estas medidas.

Calinski-Harabasz Index.

Este índice es la tasa entre la suma de la dispersión inter-clúster y la dispersión intra-clúster para todos los clústers. La dispersión se entiende aquí como la suma de la distancia cuadrada.

Por tanto, para un conjunto de datos E, con tamaño n_E el cual ha sido agrupado en k clústers, entonces:

$$s = \frac{\frac{tr(B_k)}{tr(W_k)} * n_E - k}{k-1}$$

Donde tr() es la función Trace. Y W_k y B_k son definidos como:

$$W_k = \sum_{q=1}^{k} \sum_{x \in C_q} (x - C_q)(x - C_q)^T$$

$$B_k = \sum_{q=1}^{k} n_q (c_q - c_E)(c_q - c_E)^T$$

Una de las ventajas de usar este índice es su facilidad de interpretación, pues un valor alto de este implica clústers densos y bien separados. Sin embargo, este índice suele ser un poco más alto para clústers convexos.

En la siguiente tabla, mostramos los valores de este índice, para evaluar el resultado de la aplicación de varios métodos al Iris Dataset, usando la implementación de este en el módulo scikit-learn de python.