Actividad 1

Datos del estudiante

- **Nombre:** Sergio
- **Apellidos:** Roselló Morell
- **DNI:** 53632974X
- **email:** Sergio-resello@hotmail.con

Información sobre el entorno

- **Sistema Operativo:** Arch Linux
- **Entorno de escritorio:** dwm
- **Versión de Python:** Python 3.8.2
- **Editor de texto:** NeoVim
- **Generación del documento:** Escrito en MD, compilado a LaTeX con

```
nnoremap <leader>e :! pandoc % -f markdown -t latex -s
-o %:r.pdf<cr>
```

Resumen

Se va a realizar un ejercicio de clustering guiado en la primera fase del ejercicio, para luego pasar a un ejercicio no guiado, en el que se tendrá que llegar a analizar tanto el mecanismo de clustering como el resultado que propone el algoritmo y justificarlo.

Apartado 1 - Cálculos con los datos de ejemplo

En el algoritmo de K-means, únicamente podemos cambiar el número de clusters que queremos generar. En este ejemplo, tiene sentido que lo asignemos a dos, ya que un cluster debería pertenecer a las IP con tráfico estándar y el otro debería pertenecer a las IP con tráfico malicioso.

A partir de los resultados, tenemos que determinar si tienen sentido o no. Este es el verdadero trabajo del analista.

Valores de parámetros con K-Menas En la preparación de los datos de entrada, el analista decide usar únicamente las direcciones IP que tengan mas de 5 entradas en el registro. Esto es útil para buscar un tipo específico de ataque, pero dependiendo del ataque que se quiera buscar, se deberá decidir el número de entradas y valores a revisar. Si el analista busca encontrar un ataque con persistencia en el sistema, no debería contar el número de peticiones, sino el valor de la petición.

En el ejemplo del libro, se establece el hyperparámetro 2, de acuerdo con la lógica descrita anteriormente.

En siguientes iteraciones, he creado un script sencillo que itere desde 2 a 22 clusters para posteriormente analizar los resultados y determinar una solución que englobe los máximos casos de peticiones maliciosas posibles.

Modificaciones de código con K-Menas No ha habido modificaciones de código en la solución guiada.

Se ha generado un script sencillo para realizar la iteración de los clusters

```
#!/bin/bash

COUNTS=20
CLUSTERS=2

while [ $COUNTS -gt 0 ]
do
  python cluster_vectors.py -c kmeans -n $CLUSTERS -i secrepo.h5 -o secrepo-$CLUSTERS.h5
  python stats_vectors.py -i secrepo-$CLUSTERS.h5 > secrepo-$CLUSTERS.log
  python label_notes.py -i secrepo-$CLUSTERS.h5 | grep  70.32.104.50 >> secrepo-$CLUSTERS.lo
  COUNTS=$[$COUNTS-1]
  CLUSTERS=$[$CLUSTERS+1]
done
```

Fallos o errores y solución con K-Menas No he tenido ningún fallo en el cálculo del ejercicio. Lo único relevante es que se ha tenido que cambiar el comando para obtener las estadísticas de:

```
python stats_vectors.py archivo.h5
```

A

```
python stats_vectors.py -i archivo.h5
```

Mis resultados siguiendo la guía del libro base de la asignatura son algo similares al mismo.

Ambos clusters se han agrupado de la siguiente forma:

- **Label 0:** "silhouette" 0.833 con 77.98% pertenencia (las conexiones "normales")
- **Label 1:** "silhouette" 0.242 con 22.02% pertenencia (las conexiones "diferentes" que no necesariamente significan que son ataques, pero son más probables)

Buscando la dirección IP que sabemos que es maliciosa, vemos que efectivamente, se cumple nuestra hipótesis. En nuestro caso, pertenece al grupo 1.

Una nueva revisión de los clusters, esta vez añadiendo más clusters, nos indica que, efectivamente con 12 clusters llegamos al cluster mas preciso con el menor nivel de computación necesario.

```
12 clusters:
  Number of items: 69 (0.69%) (avg silhouette: 0.30414125323295593)
```

A partir del cluster 12 vuelve a haber una subida de imprecisión hasta la separación en 17 clusters. De ahí en adelante, el cluster con la IP maliciosa queda estable con 28 elementos. Estos resultados ya parecen sobreoptimizados, de forma que pienso que la separación el 12 clusters es la más óptima dados los datos de entrada.

Dado que los resultados del algoritmo dependen de la posición inicial a la que se asignan los primeros centroides, el procedimiento se ha realizado tres veces para asegurar que es un resultado correcto. En los tres casos, el mejor resultado surge al dividir el dataset en 12 clusters.

Valores de parámetros con DBSCAN En las primeras pruebas se han usado los mismos parámetros que en el ejercicio guiado del libro base de la asignatura.

Al ver que no estaba siendo efectivo el método clustering, se ha cambiado `epsilon` a valores menores. Este cambio hace que se contemplen dos puntos como que están en un "vecindario" distinto si su distancia (En este caso Euclídea) es superior a `epsilon`. Esto quiere decir, que a menor `epsilon`, mayor la subdivisión en clusters de los datos.

Puesto a que `min_samples` controla la capacidad de crear clusters en relación a los puntos que un centroide tiene a su alrededor, no se quiere reducir demasiado el valor ya que esto hará que se creen clusters de valores irrelevantes.

Modificaciones de código con DBSCAN No se ha modificado el código de los scripts proporcionados por el repositorio del libro de la asignatura.

Fallos o errores y solución con DBSCAN Una de las curiosidades que se han dado es que, a diferencia del resultado que le aparece a los autores del libro, el algoritmo DBSCAN me indica que hay dos o tres clusters en el grupo proporcionado. Esto es un resultado un tanto extraño y no consigo averiguar a qué se debe.

```
python cluster_vectors.py -c dbscan -e 0.5 -m 5 -i secrepo.h5 -o secrepo.h5
Label -1 has 18 samples
Label 0 has 9977 samples
Label 1 has 5 samples
python cluster_vectors.py -c dbscan -e 0.6 -m 5 -i secrepo.h5 -o secrepo.h5
Label -1 has 16 samples
Label 0 has 9984 samples
```

Con los resultados obtenidos en el análisis anterior (Provisto por el libro) no se clasifican bien las IP maliciosas. Como ejemplo, la IP 70.32.104.50 aparece en el cluster 0, junto con la mayoría de IP en el sistema.

Tras probar con una variedad de valores, tanto de `min_samples` como de `epsilon`, teniendo en cuenta nuestros datos, pienso que nos interesa encontrar un número de clusters prudente relacionando puntos cercanos entre si. Esto se traduce a aumentar `min_samples` a 8 y reducir `epsilon` a 0.1

Estos datos no se ven respaldados puesto que la dirección IP 70.32.104.50 se encuentra en el grupo 0, que tiene 22 otras IP's de las cuales muchas tienen comportamientos distintos y no parecen seguir un comportamiento predecible.

Por ejemplo, otra dirección IP con intención maliciosa es la `192.187.126.162`, que pertenece al cluster -1.

Apartado 2 - Calculo con otros datos diferentes

Valores de parámetros - K-means Se han probado las agrupaciones de clusters desde 2 a 21 usando el algoritmo K-MEANS. A continuación, se ha creado una tabla para analizar los resultados. Esta tabla se ha ordenado según el valor de silhouette por cada cluster.

Una muestra de la tabla:

10:		11:		12:		13:	
14418	0.935	14184	0.956	14427	0.934	14316	0.946
571	0.926	576	0.919	225	0.920	571	0.925
480	0.695	258	0.832	576	0.911	259	0.827
674	0.687	426	0.823	426	0.822	429	0.815
618	0.675	661	0.667	659	0.701	654	0.694
321	0.654	409	0.598	46	0.662	618	0.681
408	0.601	264	0.568	406	0.603	320	0.579
804	0.454	768	0.508	666	0.578	48	0.579
310	0.248	800	0.472	248	0.422	265	0.553
1627	0.192	70	0.138	957	0.339	208	0.530
		1815	0.080	1526	0.263	769	0.502
				69	0.150	52	0.323
						1721	0.144

Se ha creado esta tabla porque se pretende hacer un seguimiento de los clusters a medida que se van separando. En este proceso, se pueden identificar los clusters que tienen relación entre iteraciones de subdivisión y posiblemente explicar la separación de algunos de los nuevos clusters.

Modificaciones de código Podemos observar como se mantienen algunos grupos entre iteraciones de clusters (Estos son los que nos interesan, especialmente, cuando tienen una silhouette alta), pero hay otros, que no se mantienen, y que se dividen en sub-clusters.

Usando el numero de elementos en el cluster y su silhouette, se ha intentado identificar a cada cluster. Algunos clusters saltan a la vista debido a su gran

numero de elementos y alto silhouette. Estos han sido sencillos de identificar. Los que no tienen un silhouette ni un numero de elementos en el cluster diferentes a los demás, ha sido muy complicado identificarlos.

El proceso de identificación ha sido clave para averiguar que clusters se mantienen estables con el paso de la división en mas clusters.

Fallos o errores y solución Un problema encontrado es que al ser la numeración de clusters aleatoria, no podemos determinar exactamente que grupo se divide y cual se queda. Solo podemos inferirlo. Esto introduce un nivel de complejidad mucho mayor al análisis.

Para superar esta limitación, seria necesario implementar un algoritmo que analiza la salida de K-means y que compare los clusters de los datos con IP, para obtener el cluster en el que estaban anteriormente y el cluster en el que están ahora.

Según los datos obtenidos, pienso que una posible agrupación optima es dividir el grupo en 15 clusters. En el paso de 14 clusters a 15 clusters, hay uno con silhouette 0.69, que luego pasa a subdividirse y recobrar una pertenencia mucho mayor de 0.90. A partir de la subdivisión 15, este cluster se mantiene bastante estable, con una pertenencia también estable.

Al analizar los datos del cluster, veo que la IP `5.101.65.160` pertenece al mismo y que sus registros parecen tener un comportamiento un tanto malicioso. Un extracto de las peticiones:

```
/administrator/index.php HTTP/1.1"
/admin/index.php HTTP/1.1"
/admin.php HTTP/1.1"
/admin.php?/cp/login&return= HTTP/1
/manager/ HTTP/1.1"
/admin.php HTTP/1.1"
/index.php/admin HTTP/1.1"
```

Una revisión de otra IP que también pertenece al cluster de la anterior es la `5.101.65.120`. Esta también tienen indicios de ser maliciosa. Revisando sus logs, vemos el siguiente trafico:

```
/administrator/index.php
/admin/index.php HTTP/1.
/admin.php HTTP/1.1"
/admin.php?/cp/login&ret
/admin.php HTTP/1.1"
/manager/ HTTP/1.1"
/index.php/admin HTTP/1.
```

Esto es una prueba de que el algoritmo esta dividiendo correctamente los datos presentados.

Esto no es una prueba de que este cluster es el único que existe con posibles IP maliciosas. Recomendaría realizar el análisis puncionado anteriormente con demás clusters que se comporten de una forma similar y revisar una de las IP que pertenecen a el. Si esta IP contiene peticiones maliciosas, podemos pasar a añadir el cluster a la lista de clusters a analizar.

Comentarios y opiniones

Estos algoritmos de subdivisión tienen un potencial tremendo, especialmente para analizar cantidades de datos grandes, ya que cuantos mas datos (Seleccionados correctamente), mejor analizara el algoritmo.

La desventaja que les encuentro es que el analista tiene que ser experto en el campo que esta analizando el algoritmo, porque sino, no hay forma de que sepa separar y analizar la salida de los datos.

En definitiva, veo que estos algoritmos son una forma de ordenar los datos, pero que debe existir una capa lógica que se encargue de verificar y validar la salida de los mismos.

Dificultades/Problemas encontradas Una de las dificultades de esta practica ha sido el análisis de los datos que saca el algoritmo. De toda la información que te presenta, debes ser capaz de saber que datos son necesarios y cuales no.

Programas/Ayudas utilizadas Ha sido indispensable para mi tener algo de conocimiento en expresiones regulares y mas reduce, ya que para analizar los resultados del algoritmo, he tenido que aislar los datos que me interesan y sonetizar las entradas de los logs del sistema.

Los programas que he usado para realizar estas tareas son:

- Python
- NeoVim y su capacidad de búsqueda y sustitución
- column (Bash)
- sort (Bash)
- awk (Bash)

Comentarios sobre la realización de la actividad En definitiva, esta practica me he proporcionado una visión mas practica del proceso de análisis de datos con algoritmos de clustering. Una cosa es saber implementar el algoritmo y otra muy distinta, es realizar un análisis con el.

Bibliografía

- *scikit-learn:* https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html