Universidad Simón Bolívar
Departamento de Computación y Tecnología de la Información
Inteligencia Artificial II

Integrantes:

Leslie Rodrigues	10-10613
Erick Silva	11-10969
Georvic Tur	12-11402

Resumen

Se implementó el algoritmo *K-means clustering* y se aplicó a dos problemas: compresión de imágenes y agrupamiento. En el primer caso, se aplicó el algoritmo en dos imágenes distintas y se corrió con las primeras siete potencias de dos como número de clusters. El color de cada cluster es calculado como el promedio vectorial del color de todos los puntos del mismo, siguiendo la codificación RGB. Posteriormente, dicho promedio reemplaza a todos los puntos de su cluster y se obtiene una imagen comprimida con una cantidad de colores igual al número de centros.

El otro problema al cual se aplicó el algoritmo fue al agrupamiento de la Iris Setosa. En este caso, se calculó el número de instancias asignadas a cada cluster por cada clase real y las distancias de cada instancia a todos los clusters obtenidos. Estos resultados se guardan, finalmente, en archivos de formato CSV.

Detalles de la Implementación

Como se está probando una implementación del *k-means clustering*, siendo este un algoritmo de aprendizaje no supervisado, se removió el atributo de clase del conjunto de datos *Iris.data*.

Por otra parte, el algoritmo implementado devuelve dos objetos: la asignación de instancias a un *cluster* y el promedio de los puntos de cada *cluster*. La métrica usada es la norma cuadrática usual.

Para acelerar la corrida del algoritmo al usarlo en las imágenes, se han seleccionado imágenes de una pequeña variedad de colores y limitado las iteraciones a solo 3 por imagen. A pesar de esto, los efectos de la compresión son fácilmente apreciables. El criterio de parada

para otros casos fue que los datos que son reclasificados en una iteración fueran menor al 0.01% de la data.

Presentación de los Resultados

Iris Setosa

A continuación se muestran las tablas de tres corridas distintas del algoritmo *K-means* con el conjunto de datos *Iris Setosa*. A dicho conjunto se le ha removido la columna de clase. Si se desean ver los resultados de una nueva corrida, éstos se almacenan en el directorio *Resultados*. Dentro de una misma columna se muestran los errores en color rojo. En azul se muestran las instancias cuyos clusters separan bien a alguna de las clases.

En general, la clase *Iris-Setosa* se agrupa bien para todos los números de clusters usados. Sin embargo, el agrupamiento de las otras dos clases no discrimina bien sus verdaderas clases. Sin embargo, esto mejora levemente a medida que aumenta el número de centros.

Resultados Para dos *clusters*				
Clase Real	Etiqueta	Cantidad Corrida 1	Cantidad Corrida 2	Cantidad Corrida 3
Iris-Setosa	0	0	50	50
	1	50	0	0
Iris-Versicolor	0	50	0	0
	1	0	50	50
Iris-Virginica	0	50	0	0
	1	0	50	50

Tabla N° 1

Resultados Para tres *clusters*				
Clase Real	Etiqueta	Cantidad Corrida 1	Cantidad Corrida 2	Cantidad Corrida 3
Iris-Setosa	0	0	0	49
	1	50	50	1

	2	0	0	0
Iris-Versicolor	0	42	8	0
	1	0	0	37
	2	8	42	13
Iris-Virginica	0	17	33	0
	1	0	0	8
	2	33	17	42

Tabla N° 2

Resultados Para cuatro *clusters*				
Clase Real	Etiqueta	Cantidad Corrida 1	Cantidad Corrida 2	Cantidad Corrida 3
Iris-Setosa	0	13	23	27
	1	37	0	0
	2	0	0	0
	3	0	27	23
Iris-Versicolor	0	0	0	0
	1	0	39	12
	2	12	11	38
	3	38	0	0
Iris-Virginica	0	0	0	0
	1	0	17	39
	2	39	33	11
	3	11	0	0

Tabla N° 3

Resultados Para cinco *clusters*				
Clase Real	Etiqueta	Cantidad	Cantidad	Cantidad

		Corrida 1	Corrida 2	Corrida 3
Iris-Setosa	0	1	0	0
	1	0	15	49
	2	0	35	0
	3	0	0	1
	4	49	0	0
Iris-Versicolor	0	11	3	15
	1	22	0	0
	2	0	0	0
	3	17	19	13
	4	0	28	22
Iris-Virginica	0	1	31	17
	1	7	0	0
	2	26	0	26
	3	16	3	2
	4	0	16	5

Tabla N° 4

En cuanto a las distancias entre los clusters y cada instancia, éstas también son guardadas en el directorio *Resultados/*. Al examinar los archivos para dos y tres componentes, notamos que para una mismo centro las distancias tienen a ser de alta magnitud o de baja magnitud comparativamente. Esto se debe a la maximización de la cohesión interna y la separación externa

Compresión de Imágenes

Se usó también el algoritmo de k-medias para hacer una especie de compresión de imágenes usando la posición de los clusters.

Para lograr esta comprensión se uso la libreria pillow de python para obtener una descripción pixel por pixel de la imagen, después de esto se usó el algoritmo de k-medias para asociar grupos de pixeles a clusters usando los colores RGB de cada pixel como posiciones,

una vez finalizado el algoritmo se tenían K diferentes clusters con diferentes posiciones que representaban colores en RGB y las asociaciones de cada píxel original a uno de esos clusters, con esto se cambiaron los colores de los píxeles originales a los de el respectivo cluster para obtener una imagen con solo K diferentes colores. Debido al tiempo que toma K-medias en llegar a convergencia, se usaron solo 3 iteraciones de K-medias para llegar a estas imágenes (Puesto que los cambios después de 3 iteraciones se consideraron muy bajos)

Se usó este procedimiento para generar imágenes con solo 2,4,8,16,32,64 y 128 colores a partir de una imagen original y se guardaron los diferentes resultados, mientras que el hecho de tener menos colores no decrece el tamaño de la imagen, los algoritmos usados por cada formato de imagen toman en cuenta esto y mediante sus propios algoritmos de compresión hacen que la imagen pese menos al tener menos píxeles de colores diferentes. Así, se vio claramente un cambio de tamaño entre cada imagen.

Las imágenes usadas se encuentran anexadas, los tamaños observados en cada compresión son los siguientes:

stardewValley.png	
Número de Colores	Tamaño (KiloBytes)
Original	553,0 KB
2	43,6 KB
4	74,6 KB
8	139,6 KB
16	202,0 KB
32	228,9 KB
64	294,2 KB
128	336,9 KB

Mudkipz.jpg	
Número de Colores	Tamaño (KiloBytes)
Original	38,0 KB
2	12,9 KB
4	17,2 KB
8	21,5 KB
16	21,7 KB
32	22,0 KB
64	21,9 KB
128	21,7 KB

Se puede ver claramente la reducción del tamaño causada por menos colores, aunque en la imagen “Mudkipz.jpg” no se encontró que los pesos de la imagen crecieran sólo con el número de colores, esto indica que el algoritmo interno de compresión que usa el formato debe estar realizando otras operaciones que no dependen únicamente del número de colores.

Conclusiones

Aunque en muchos casos no son tan poderosas como el aprendizaje supervisado, las técnicas de *clustering* son de gran utilidad, dado que agrupan los datos de acuerdo a las distancias entre sus atributos, lo que permite buscar una similitud entre varias instancias aun cuando no se conozca alguna clasificación entre ellas.

Para separar atributos con clase sin usar o necesitar las etiquetas, las técnicas de clustering, dados suficientes clusters, pueden obtener clasificaciones bastante exactas aunque pueden no demostrar claramente que cluster pertenece a que clasificación.

El aprendizaje no supervisado es muy útil cuando se quiere agrupar grandes cantidades de data en búsqueda de patrones o similitudes entre la misma, ya que no tiene la necesidad de crear una clasificación específica o hacer una clasificación manual de varios de los datos para el entrenamiento, lo que puede salvar mucho tiempo.