Proyecto final

Reconocimiento de patrones usando aprendizaje supervisado y no supervisado.

Remigio Alvarado Alberto
201628015

17 de noviembre de 2020

Índice general

Capítulo 1

Proyecto final

1.1. Introducción

Con la cantidad de datos que son generados diariamente y que continúan aumentando, es indispensable crear sistemas cada vez más inteligentes, los sistemas inteligentes a menudo hacen uso de técnicas capaces de reconocer patrones. La capacidad para reconocer patrones puede considerarse un requisito previo para el comportamiento inteligente [6].

El reconocimiento de patrones no es una técnica, en cambio, son un amplio conjunto de conocimientos y técnicas. Entonces, podemos decir que el reconocimiento de patrones es un área con varias aplicaciones, tiene como propósito extraer información relevante capaz de describir objetos y relaciones entre objetos.

Para hacer uso del reconocimiento de patrones se necesitan tener patrones, un patrón puede verse como un conjunto de mediciones, observaciones o características de un objeto. El comportamiento o categoría de un objeto puede describirse mediante sus patrones distintivos, es decir patrones similares pueden denotar objetos parecidos, mientras que patrones distintos denotan objetos diferentes. Regularmente para hacer reconocimiento de patrones se siguen los siguientes pasos:

- Recopilación de datos.
- Preprocesamiento (Limpieza de datos).
- Extracción de características.
- Clasificación.
- Toma de decisiones.

En este reporte se hace uso del reconocimiento de patrones para clasificar o agrupar individuos de acuerdo con sus características, para lograr esto se hace uso de 2 enfoques:

1. Aprendizaje supervisado.
2. Aprendizaje no supervisado.

1.2. Aprendizaje supervisado

Cuando se habla de aprendizaje automático existen diferentes enfoques para generar este aprendizaje, los modelos de aprendizaje supervisado aprenden funciones y relaciones que asocian datos de entrada con una determinada salida, se ajustan a un conjunto de ejemplos de los que se conoce la relación entre las entradas y la salida deseada. El objetivo que se tiene al usar aprendizaje supervisado es poder predecir el valor correspondiente a cualquier entrada después de haber visto una serie de ejemplos, el modelo deberá generalizar los ejemplos de manera correcta para que los resultados sean confiables.

Existen 2 subcategorías en el aprendizaje supervisado, estas dependen del tipo de salida deseada. Si la salida es un valor categórico estamos hablando de clasificación, mientras que si la salida es un valor de un espacio continuo el modelo será de regresión.

1.2.1. Algoritmo k-nearest neighbors (k-NN)

Propuesto por Thomas Cover y utilizado en modelos de aprendizaje automático para clasificación y regresión, es un algoritmo basado en instancias, k-NN usa los datos de entrenamiento para predecir el valor de un nuevo objeto. Casos de uso de k-NN:

- Clasificación. Un nuevo objeto es clasificado mediante la clase mayoritaria de sus vecinos más cercanos, los vecinos a tomar en cuenta van desde $k = 1...n$, mayormente son usados valores de k pequeños e impares. Normalizar los datos de entrenamiento puede mejorar la precisión del algoritmo.
- Regresión. El valor predicho para el nuevo objeto será el promedio de sus vecinos más cercanos.

La idea detrás del algoritmo es “Objetos más parecidos son aquellos más cercanos”.

Algoritmo 1: k-Nearest Neighbors

Entrada:
X ← Conjunto de entrenamiento.
Y ← Conjunto de prueba.
k ← Vecinos a considerar.

Salida :
Y clasificado.

```
Inicio;
para cada Individuo ∈ Y hacer
    para cada Elemento ∈ X hacer
        Calcular la distancia D(Individuo, Elemento);
    fin
    Ordenar X según su distancia D(Individuo, Elemento);
    Tomar las clases de los k-vecinos con menor distancia;
    Asignar al Individuo la clase con mayor aparición;
fin
Fin;
```

Consideraciones al usar k-NN:

- Es necesaria una función de distancia válida, al elegir la función de distancia se deben considerar el tipo de atributos de los objetos.
- Cuando existe un sesgo en el conjunto de entrenamiento se puede ponderar la contribución de cada vecino dependiendo de su distancia.
- El tiempo que tarda k-NN depende del conjunto de entrenamiento, entre más grande sea, el tiempo necesario para realizar los cálculos aumenta. Existen propuestas para reducir el tiempo de ejecución como lo es The linear approximating and eliminating search algorithm (LAESA).
- La idea de k-NN también puede ser aplicada en enfoques diferentes a clasificación y regresión.
- Las matrices de confusión, cálculo de la exactitud y visualización de los datos son buenas herramientas para validar los resultados después de aplicar k-NN.

1.3. Aprendizaje no supervisado

En los modelos de aprendizaje no supervisado a diferencia de los supervisados no se busca ajustar pares de entrada/salida, en cambio, si aumentar el conocimiento estructural de los datos. Hay varias formas en las que se puede hacer esto:

- Clustering. Agrupando los datos según sus similitudes
- Reducción de dimensionalidad. Simplificando la estructura de los datos manteniendo características fundamentales.
- Aprendizaje topológico. Extrayendo la estructura interna con la que se distribuyen los datos en su espacio original.
- Entre otras.

Entonces, al utilizar aprendizaje no supervisado podríamos decir que se cuenta con más incertidumbre, y los algoritmos son más costosos al requerir más pruebas de ensayo y error. En particular en este reporte se usa una técnica de agrupamiento. El agrupamiento es el proceso de organizar objetos en distintos grupos donde los miembros son similares entre ellos de alguna manera.

1.3.1. Algoritmo k-MEANS

El algoritmo k-means propuesto por Stuart Lloyd tiene como objetivo dividir un conjunto de datos en k grupos, concretamente cada individuo en el conjunto de datos pertenecerá al grupo cuyo centroide este más cercano. Como no se tiene conocimiento sobre como deban agruparse los datos, no hay una manera certera que indique lo bien o mal que se está haciendo la agrupación.

Los datos juegan el papel más importante ya que se depende completamente de ellos para generar los grupos, el valor de k no podrá conocerse a priori. k-medias intenta minimizar la varianza dentro de los grupos, pero con el algoritmo estándar no se sabe un valor de k y se tendrán que realizar pruebas para definirlo. Una vez teniendo un valor de k el problema converge a un óptimo local, ya que no existe manera de garantizar que es el óptimo.

Algoritmo 2: k - MEANS

Entrada:

E ← Conjunto de elementos.
k ← Número de grupos.
Max ← Número máximo de iteraciones.

Salida :

K grupos definidos.

```
Inicio;
C ← k centroides aleatorios (C ⊂ E);
mientras Iteraciones<Max & los centroides sigan cambiando
 hacer
    para cada Elemento ∈ E hacer
        para cada Centroide ∈ C hacer
            Calcular la distancia D(Elemento, Centroide);
        fin
        Asignar al Elemento el grupo del centroide más cercano;
    fin
    Calcular los nuevos centroides con la media de los Elementos que
     pertenecen al grupo;
fin
Fin;
```

Consideraciones al usar k-Means:

- Al no tener un estimado sobre el valor de k la cantidad de pruebas a realizar aumenta.
- Dado que los centroides iniciales son aleatorios con cada iteración los resultados pueden variar, una solución para que los centroides no cambien es utilizar una semilla al momento de la selección al azar.
- Se requieren pocas iteraciones para converger al óptimo local.
- Si se utiliza un valor de k elevado, la cantidad de información resultante del grupo disminuye.

1.4. Implementación

Para lograr la implementación de los algoritmos anteriormente descritos se uso:

- Lenguaje de programación: Python.
- Integrated Development Environment (IDE): Jupyter Notebook.

- Bibliotecas: Matplotlib, Random y Ceil.
- Conjuntos de datos provistos por el profesor.
- Computadora personal.

1.4.1. Funciones de distancia

Los algoritmos explorados en este reporte hacen uso de la cercanía entre objetos, para calcular esta cercanía es necesaria una función de distancia que nos diga que tan lejos o cerca se encuentran los datos. La función de distancia debe cumplir con 3 condiciones:

1. La distancia entre 2 puntos debe ser siempre positiva $f(x, y) \geq 0$
2. Simetría. $f(x, y) = f(y, x)$
3. Desigualdad del triángulo: dados 3 puntos cualesquiera se debe cumplir $f(1, 2) \leq f(1, 3) + f(3, 2)$

Se utilizan 3 funciones de distancia que si se analizan a profundidad son básicamente lo mismo, en las 3 se toma en cuenta la diferencia atributo a atributo entre 2 objetos, solo que a diferente potencia.

Distancia Euclidiana

$$d_E(P, Q) = \sqrt{(p1 - q1)^2 + (p2 - q2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (pi - qi)^2} \tag{1.1}$$

Donde $P = (p1, p2, p3, ..., pn)$ y $Q = (q1, q2, q3, ..., qn)$ son vectores de características.

Distancia Manhattan

$$d_m(P, Q) = |p1 - q1| + |p2 - q2| + ... + |p_n - q_n| = \sum_{i=1}^{n} |pi - qi| \tag{1.2}$$

Donde $P = (p1, p2, p3, ..., pn)$ y $Q = (q1, q2, q3, ..., qn)$ son vectores de características.

Distancia Minkowski

$$d(P, Q) = (\sum_{i=1}^{n} |p_i - q_i|^n)^{\frac{1}{n}} \tag{1.3}$$

Donde $P = (p1, p2, p3, ..., pn)$ y $Q = (q1, q2, q3, ..., qn)$ son vectores de características, para $n = 1$, la expresión anterior coincide con la distancia Manhattan, y para $n = 2$ coincide con la distancia Euclideana.

1.5. Resultados

A continuación, se mostrarán los resultados usando las 3 funciones de distancia, dependiendo del algoritmo los resultados se mostraran de forma distinta, de tal manera que sea más fácil su entendimiento. En casos donde el algoritmo tiene variantes se profundizará un poco más sobre la variante.

1.5.1. k-NN

Para realizar la evaluación al método k-NN se hace uso de un conjunto de prueba y de validación cruzada con el mismo conjunto de entrenamiento. La exactitud usada se define como

$$Exactitud = (100/tam\_prueba) * (tam\_prueba - error) \tag{1.4}$$

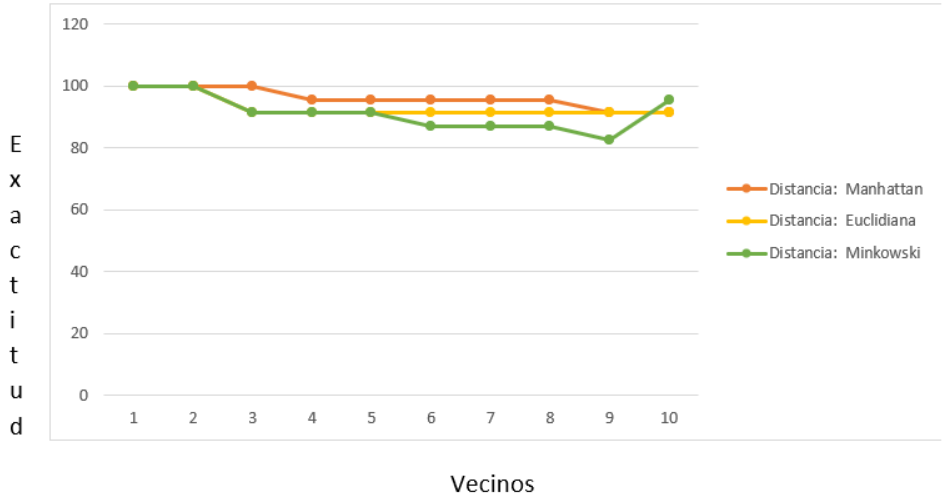


Figura 1.1: Comparación de los resultados obtenidos tomando en cuenta k vecinos diferentes, para el caso de la distancia Minkowski $n = 3$ en todas las pruebas. Como se aprecia, en promedio el uso de la distancia Manhattan devuelve mejores resultados.

Cuando se hace uso de validación cruzada estamos hablando de una técnica que garantiza que los resultados sean independientes del conjunto de prueba y entrenamiento, la técnica divide el conjunto de datos en k partes iguales (Cuando k es múltiplo del número de objetos en el conjunto de datos), en cada iteración la parte correspondiente será el conjunto de prueba y el

resto el conjunto de entrenamiento. La exactitud resultante será el promedio de todas las exactitudes obtenidas en el proceso.

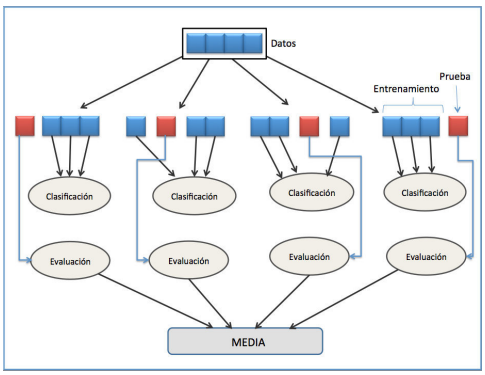


Figura 1.2: Ejemplo de validación cruzada en un conjunto de datos

Dado que, si se divide el conjunto de datos partiendo de la manera en que está ordenado, esto pueda sesgar el modelo se optó por escoger la partición de manera aleatoria en las k iteraciones. Idealmente todo objeto en el conjunto de datos deberá ser parte del conjunto de prueba, y k-1 veces parte del conjunto de entrenamiento.

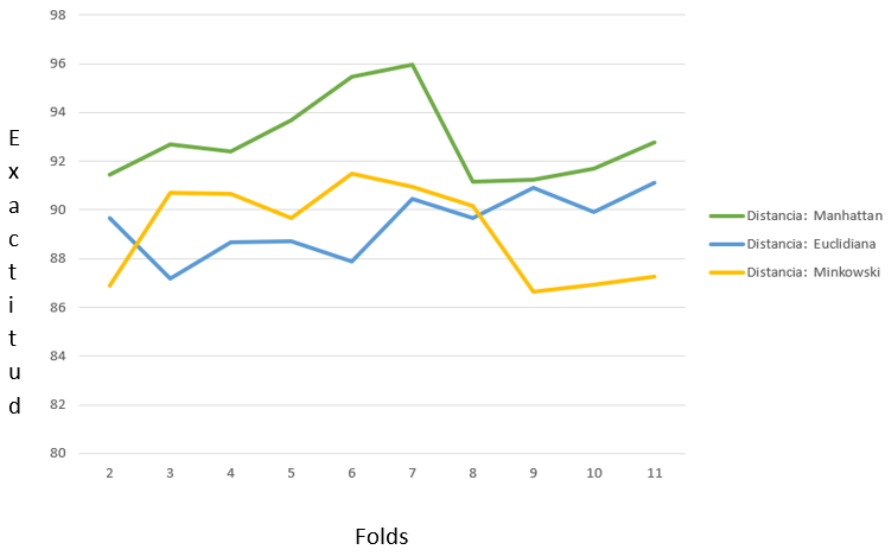


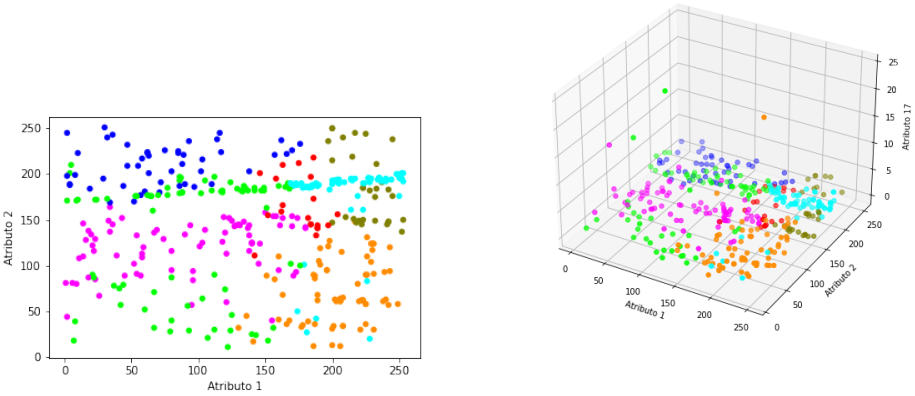
Figura 1.3: Resultados tomando en cuenta 3 vecinos más cercanos y diferentes valores para la validación cruzada. Los mejores resultados parecen estar cuando se hace uso de la distancia Manhattan, esto puede deberse a que solo se hace uso del valor absoluto de la diferencia atributo a atributo y no se castiga cuando la diferencia es grande.

Para todas las pruebas realizadas es calculada y mostrada la respectiva matriz de confusión, con la cual es más sencillo visualizar en que casos el método se confunde más.

1.5.2. k-Means

Medir la exactitud en un modelo de agrupamiento es prácticamente imposible, ya que se desconoce en realidad como deben estar formados los grupos. En nuestro caso tenemos el conjunto de datos etiquetado, será entonces sencillo hacer una comparación con los grupos originales y los creados, además conocemos la cantidad de grupos que se deben crear con lo cual la tarea es más sencilla de lo que puede ser cuando se aplica a problemas donde no existen antecedentes.

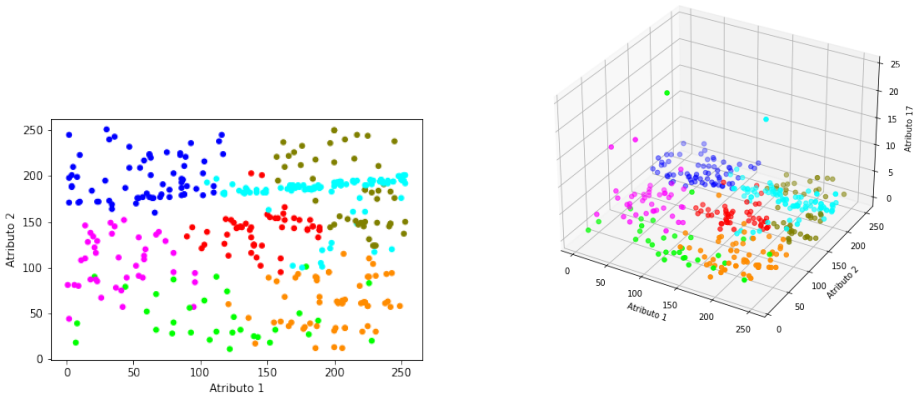
Los centroides iniciales son aleatorios y no es usada ninguna semilla para generar los números aleatorios.



(a) Inicio de los grupos en 2D

(b) Inicio de los grupos en 3D

Figura 1.4: Grupos creados inicialmente



(a) Grupos finales en 2D

(b) Grupos finales en 3D

Figura 1.5: Grupos encontrados

El valor de k para el algoritmo es de 7, como en el conjunto original. La cantidad de objetos en cada grupo es:

- El Cluster 0 tiene 50.
- El Cluster 1 tiene 48.
- El Cluster 2 tiene 59.
- El Cluster 3 tiene 31.
- El Cluster 4 tiene 100.
- El Cluster 5 tiene 68.
- El Cluster 6 tiene 41.

	Grupos creados						
	0,	1,	2,	3,	4,	5,	6
0	[9,	1,	1,	0,	0,	0,	15]
1	[3,	0,	6,	3,	10,	1,	4]
2	[6,	2,	1,	1,	0,	0,	16]
3	[3,	30,	0,	0,	0,	42,	0]
4	[0,	0,	0,	0,	90,	25,	0]
5	[0,	0,	0,	27,	0,	0,	0]
6	[29,	15,	51,	0,	0,	0,	6]

Clases originales

Figura 1.6: Distribución de los grupos encontrados en las clases originales

Es realmente incierto saber si los grupos creados describen mejor las características de un grupo de objetos que las clases originales, pero en general las agrupaciones creadas parecen tener coherencia repecto a los objetos que agrupa.

1.5.3. Comparación

Realizar una comparación entre 2 enfoques totalmente diferentes es prácticamente imposible, afortunadamente el conjunto de datos se encuentra etiquetado y para ambos enfoques existen coincidencias, las cuales serán destacadas intentando tener una comparación lo más justa posible.

- Ambos métodos aciertan generalmente bien. k-NN tiene muy buenos resultados cuando se usa el conjunto de prueba y con validación cruzada el porcentaje de exactitud siempre se mantiene por encima del 85 %. k-Means agrupa bastante bien con respecto a las clases originales, en algunos casos mantiene la misma cantidad de objetos de una clase en uno de los grupos creados.
- El tiempo de ejecución de los 2 métodos es muy rápido, específicamente en la implementación descrita en este reporte puede que k-Means tardé un poco más ya que genera durante su ejecución los Plots 2D y 3D.
- El que k-means agrupe los objetos de cierta manera tal vez pueda decir que las clases originales en realidad no describen muy bien a los objetos que pertenecen a ellas, y esto podría verse reflejado en los resultados obtenidos con validación cruzada.
- Cuando se trata de problemas en la vida real tener un 100 % de exactitud o agrupar perfectamente todos los objetos puede ser considerado imposible, esto implica tener todas las posibles variantes de los objetos y en ese caso tal vez ya no sea necesario un método de aprendizaje.

1.6. Conclusiones

El reconocimiento de patrones tiene una infinidad de aplicaciones y su uso podría resolver muchos de los problemas actuales, los resultados y el conocimiento obtenido al realizar este proyecto muestran el potencial que existe en los datos, puede que muchos problemas no sean tan complejos como aparentan y solo se necesite un mejor análisis sobre los datos disponibles.

Los métodos de aprendizaje no supervisado parecen tener más incertidumbre que los de su contraparte, las relaciones entre objetos que puede encontrar una máquina no siempre serán las más lógicas para nosotros. Se debe tener mucha paciencia por la cantidad de pruebas que se necesiten realizar antes de tener certeza sobre los resultados obtenidos, cada método y sus variantes trabajan de manera distinta, entonces lo más importante será la idea detrás de estos métodos, la claridad que se tenga sobre el problema y la solución propuesta.

Cuando de aprendizaje supervisado se habla la cantidad de técnicas parece ser más grande, pero no siempre lo más sofisticado o complejo es lo mejor. k-NN obtiene resultados muy buenos en la mayoría de los problemas donde es usado, su simplicidad ayudado de la lógica que sigue ofrece una solución que hoy en día sigue siendo muy usada.

La base o inspiración detrás de tantos sistemas de reconocimiento de patrones puede guiarnos a ideas que ofrezcan una solución inteligente y precisa a muchos problemas, prestar atención a nuestro entorno en busca de ideas puede no ser tan descabellado.

Referencias

[1] T. Cover and P. Hart. Nearest neighbor pattern classification. *IEEE transactions on information theory*, 13(1):21–27, 1967.

[2] R. O. Duda, P. E. Hart, and D. G. Stork. *Pattern classification*. John Wiley & Sons, 2012.

[3] A. K. Jain. Data clustering: 50 years beyond k-means. *Pattern recognition letters*, 31(8):651–666, 2010.

[4] T. Mitchell. *Machine learnig*. McGraw Hill, 1997.

[5] R. Sathya and A. Abraham. Comparison of supervised and unsupervised learning algorithms for pattern classification. *International Journal of Advanced Research in Artificial Intelligence*, 2(2):34–38, 2013.

[6] R. J. Schalkoff. Pattern recognition. *Wiley Encyclopedia of Computer Science and Engineering*, 2007.