El arte de la analítica:
Módulo 3: Machine Learning y el algoritmo K-Means

MSc Edoardo Bucheli

1 Introducción

En este módulo tendremos haremos una breve introducción al **Machine Learning**, implementaremos un algoritmo conocido como **K-Means** y aprenderemos a usar **scikit-learn**, un módulo de Python para Machine Learning.

2 Introducción a Machine Learning

El **Machine Learning** o Aprendizaje Automático es una rama de la **Inteligencia Artificial** que se enfoca en algoritmos que aprenden algo de su ambiente para la solución de problemas. Existe mucha variedad en los algoritmos de Machine Learning pero en general este se separa en dos ramas principales.

El **Aprendizaje Supervisado** (Supervised Machine Learning) es aquel donde los datos que utilizamos para entrenar el algoritmo contienen la respuesta. Es decir, nuestra meta es poder predecir una de las variables en el conjunto de datos para nuevos ejemplos que surjan. A esta variable que buscamos predecir le llamamos *target*. Éste es el tipo más común de Machine Learning y a su vez se divide en muchos tipos de algoritmos.

Por otro lado en el **Aprendizaje No-Supervisado** (Unsupervised Machine Learning) no contamos con ejemplos de la respuesta al problema, por lo que los algoritmos de esta clase se basan en el análisis de los datos para encontrar patrones o representaciones abstractas de los mismos. Este análisis revela cosas sobre los datos que no conocíamos anteriormente.

Estas formas de aprendizaje son a su vez solo una forma de atacar un problema, por lo que suele pasar que los algoritmos de uno se adaptan al otro y viceversa. Aún así es la forma más sencilla y consensuada para clasificarlos, por lo que esta sección intentaremos explicarlos por separado pero debes tener en mente que en cualquier momento las ideas de uno se pueden aplicar al otro y estas ideas a su vez se pueden mezclar para generar un nuevo tipo de algoritmo.

Esto contrasta con la Inteligencia Artificial Clásica que se basa en algoritmos de búsqueda de soluciones, u optimización de parámetros.

2.1 Conceptos Clave

- **Modelo**: Es lo que creamos en Machine Learning. Un modelo es lo que hace la predicción con la que intentamos resolver el problema.
- **Métrica**: Es el parámetro (por lo general numérico) con el que evaluamos la calidad de una predicción.

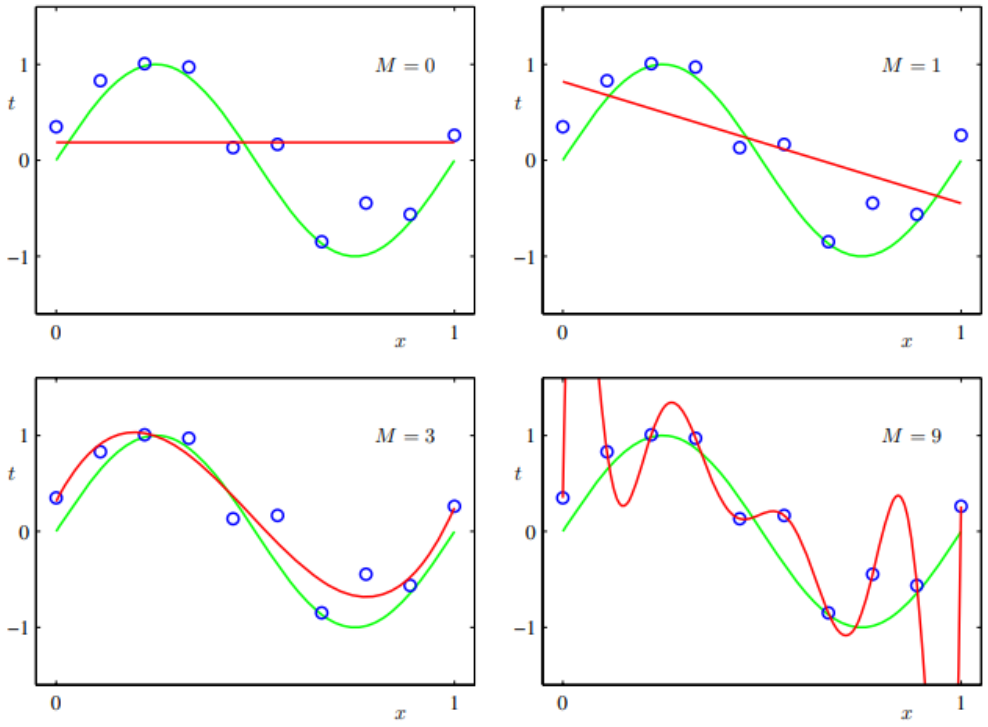

Figure 1: Underfitting y Overfitting en Regresión

- **Regresión**: En los problemas de regresión buscamos predecir una variable numérica. Ej. El precio de una casa, la temperatura, etc.
- **Clasificación**: En los problemas de clasificación buscamos predecir una variable categórica a la cual nos referimos como clases.
- ***Target***: el valor numérico o clase a predecir.
- ***Overfitting***: Cuando nuestro algoritmo aprende el ruido en los datos.
- ***Underfitting***: Lo contrario al Overfitting, cuando nuestro algoritmo no ha aprendido lo suficiente sobre los patrones de nuestros datos.

2.2 Tipos de Aprendizaje Supervisado

Es dificil hacer una separación limpia de algortmos de aprendizaje supervisado pero en esto caso lo haremos en 4 ramas:

1. Aprendizaje por Optimización de Parámetros
2. Aprendizaje por Inducción o Deducción de Reglas
3. Aprendizaje Vago (Lazy Learning)
4. Aprendizaje Bayesiano o Inferencia Estadística

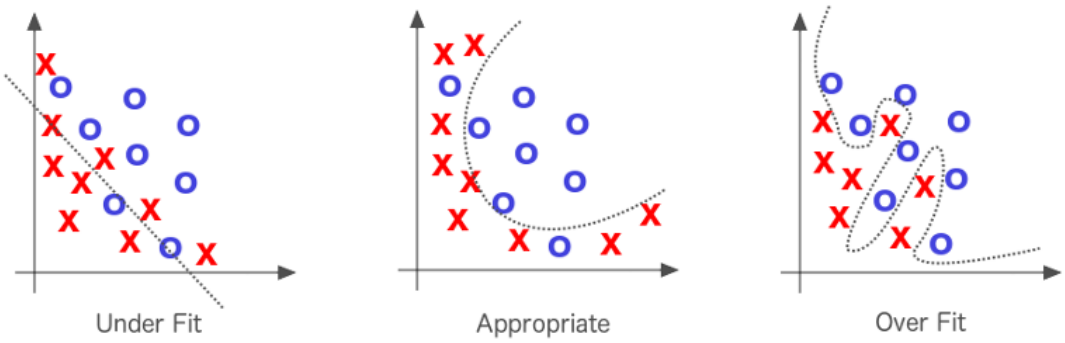

Figure 2: Underfitting y Overfitting en Clasificación

2.2.1 Optimización de Parámetros

Todos los algoritmos de este tipo se componen de manipulaciones matemáticas (por lo general sumas y multiplicaciones) aplicadas a los datos de entrada. El reto en este tipo de algoritmos es encontrar las manipulaciones correctas en la forma de optimización de parámetros.

El ejemplo más sencillo es la regresión lineal. Imaginemos que queremos predecir el valor de una variable numérica y dado el valor de otra variable x. En ese caso podríamos como hypótesis establecer una ecuación lineal de la siguiente forma,

$$y = w_1 * x + w_2 \tag{1}$$

En este caso, lo que queremos es encontrar un valor óptimo para los parámetros w_1 y w_2. En este tipo de algoritmos solemos empezar por establecer una relación matemática de este tipo para así, dados muchos ejemplos de valores de x y y correspondientes, encontrar los valores de w_1 y w_2 (pesos o *weights*) donde la predicción de y es más cercana a la realidad.

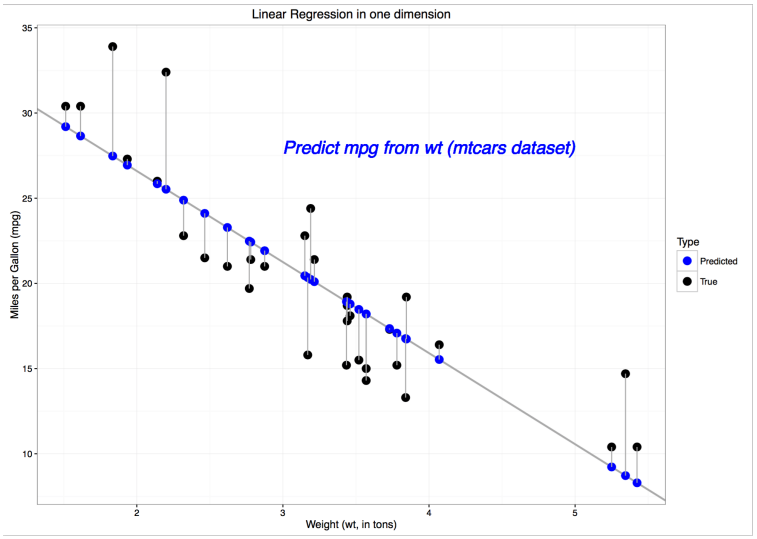

Figure 3: Ejemplo de Regresión Lineal

Las consideraciones más importantes para este tipo de algoritmo son,

- La relación matemática a utilizar
- El tipo de optimización con la que encontraremos los parámetros (notoriamente el Descenso de Gradiente, *Gradient Descent*)
- Bajo qué parámetro podemos evaluar que una hipótesis es mejor que otra.

La Regresión Lineal, Regresión Logística, Maquinas de Vectores de Soporte (Support Vector Machines) y Redes Neuronales (Deep Learning) son algunos ejemplos de algoritmos basados en la optimización de parámetros.

2.2.2 Deducción de Reglas

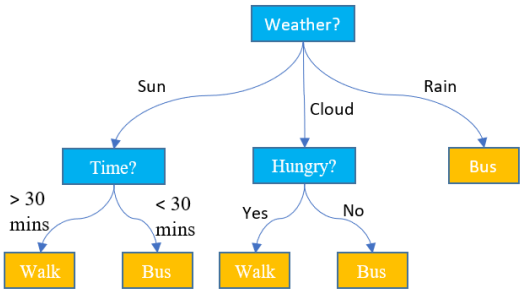

Figure 4: Ejemplo de Árbol de decisión

Esta familia de algoritmos busca generar reglas lógicas que describan un tipo de objeto o clase en el conjunto de datos. Por ejemplo, al buscar un clasificador que pueda predecir si una persona puede asumir un crédito de manera segura, un algoritmo de esta familia deberá obtener alguna regla como,

$$\text{sueldo mensual} > 50000 \text{ AND retraso más grande en días} < 10 \text{ AND edad} < 70 \text{ AND edad} > 20$$

Las consideraciones más importantes para este tipo de algoritmo son,

- Cada proposición que agregamos a una regla debe separar de la mejor manera las clases a predecir.
- Una regla debe filtrar el mayor número de ejemplos posibles que a su vez son mayoritariamente de una sola clase.
- Donde ponemos la raya que separa las reglas descritas en el punto anterior.

Los Árboles de Decisión y los clasificadores basados en Patrones son los algoritmos más típicos dentro de la inducción de reglas.

2.2.3 Lazy Learners

Estos algoritmos parten de un principio muy sencillo, eres lo que pareces. En estos algoritmos buscamos un principio matemático que nos ayude a distinguir las diferencias entre una instancia de un conjunto de datos y otra. Una vez que determinamos esto, podemos clasificar nuevos objetos basados en su similitud con otros. La consideración más importante para este tipo de algoritmo es,

- Qué métrica utilizaremos para evaluar la similitud o distancia entre dos ejemplos.

El ejemplo más tipo de Lazy Learning es el Algoritmo K-Nearest Neighbors.

2.2.4 El Teorema de Bayes y la Inferencia Estadística

Los algoritmos de esta familia se basan ampliamente en las disciplinas de Probabilidad y Estadística y por lo tantos a veces no son incluidos o presentados como Machine Learning. La predicción en este tipo de algoritmos se puede basar en muchas técnicas de esta amplia rama de las matemáticas. Uno de los usos más discutidos en el campo de Machine Learning es el Análisis Bayesiano.

El anásis Bayesiano se basa en el trabajo del Thomas Bayes, específicamente el Teorema de Bayes,

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)} \tag{2}$$

La idea central en el Análisis Bayesiano es la predicción de eventos dada su probabilidad de ocurrir. Bajo este esquema debemos siempre basarnos en probabilidades a priori (*priors*). Por ejemplo, basado en nuestro conocimiento actual, qué tan probable es que ocurra algún evento A (a este valor de probabilidad le llamamos *prior*). Dada nueva evidencia (Por ejemplo, algún evento B ocurre) ¿debe cambiar nuestro *prior*?, a esta probabilidad actualizada se le llama probabilidad posterior (*posterior*) la cual podemos calcular con la ecuación 2. El proceso anterior se debe repetir infinitamente, para cada nueva iteración, el *posterior* se convierte en *prior*.

Imaginemos que hemos vivido siempre en una cueva donde nunca hemos visto un amanecer. Un día por fin salimos y vemos un amanecer. Una persona con pensamiento Bayesiano observa esto y asigna un *prior* al evento **el sol saldrá mañana** al cual nos referiremos como A. Por lo pronto solo ha visto esto pasar una vez por lo que la probabilidad de A es muy baja. Ese día, el sol sale (evento B) por lo que la probabilidad de A crece y tenemos un nuevo *prior*. Esto se repite día con día hasta que la probabilidad del evento A es prácticamente igual a 1.

2.3 Tipos de Aprendizaje No Supervisado

Todos los métodos que acabamos de discutir se pueden aplicar de una forma u otra al aprendizaje no supervisado. En esta sección describiremos dos aplicaciones comunes de Aprendizaje No Supervisado las cuales se basan en los métodos previamente discutidos.

2.3.1 Clustering

El tipo de Aprendizaje no supervisado más común es el *clustering* donde la meta es encontrar grupos dentro de nuestro conjunto de datos con características similares. El *clustering* es muy útil cuando no sabemos el tipo de clases que tenemos en nuestra base de datos. Existen varios métodos de clustering y algunos de los más populares se basan en las ideas de Lazy Learning que discutimos en la sección anterior.

2.3.2 Autoencoders

Otro tipo de Aprendizaje No Supervisado son los *autoencoders* los cuales son generalmente versiones especiales de Redes Neuronales. La idea está en aprender a codificar y decodificar los datos de entrada para

entenderlos de una manera más abstracta. Este tipo de Machine Learning es el que nos ha llevado al desarrollo de modelos generativos que en los útimos años han presentado grandes avances en la generación de imágenes falsas, voces sintetizadas interpretación de texto y aplicaciones como los chatbots y Deep Fakes.

2.4 Validación

Uno de los procesos más importantes y complicados en cualquier proyecto de Machine Learning es la validación de resultados. Si no tenemos cuidado en esta fase es muy fácil que nuestro algoritmo tenga deficiencias muy importantes sin que nos demos cuenta, lo cual puede tener consecuencias muy importantes.

El proceso de validación más sencillo es el siguiente:

1. Entrenamos el modelo con un subconjunto de los ejemplos en nuestro conjunto de datos.
2. Una vez entrenado el modelo hacemos una prueba con los datos restantes.
3. Comparamos nuestras predicciones con los valores reales.

Podríamos dedicar incontables horas de estudio a cada uno de estos pasos e idealmente los repetiremos muchas veces antes de dar por terminado el proceso.

En el paso 1 podemos hablar de distintos algoritmos, la cantidad de ejemplos que utilizamos para entrenar, la estrategia para escoger ese subconjuntos de datos, entre muchas otras cosas.

En el paso 2 debemos asegurarnos que los ejemplos que utilizamos no estuvieron incluidos entre los que utilizamos para entrenar. También puede ser conveniente hacer una tercera segmentación para evaluar la posibilidad de ruido en esta muestra.

En el paso 3 debemos escoger qué métrica es la mejor para evaluar nuestros resultados. Este paso es extremadamente importante y nuestra decisión debe estar basada en el algoritmo utilizado, los datos, el tipo de problema, entre muchos otros factores.

Por ejemplo, estamos creando un clasificador y nuestra métrica es el *accuracy* (porcentaje de ejemplos clasificados correctamente). Imaginemos también que 90% de nuestros ejemplos son de una sola clase. Si nuestro clasificador decidiera siempre predecir esa clase, entonces su accuracy sería del 90%. Lo cual suena bien pero cuando analizamos nos damos cuenta que no es un buen resultado. Poner atención a estos detalles no solo es importante para que sepamos si nuestro modelo es bueno, muchos modelos se basan en este valor para optimizar sus resultados, por lo que si usamos esta métrica el modelo no sabrá como mejorar. Otros ejemplos de métricas son el *Precision*, *Recall*, *F-Score*, entre otros, pero no siempre es fácil incluir estas métricas en nuestro modelo.

3 El Algoritmo K-Means

En este caso, estudiaremos a fondo el algoritmo K-means. K-means es un algoritmo de clustering, es decir, un algoritmo no supervisado. Como mencionamos anteriormente, este algoritmo se basa en los conceptos de Lazy Learning donde queremos clasificar objetos en base a su similitud con otros objetos. Pero en este caso en lugar de hacerlo para clasificar nuevos ejemplos, lo utilizamos para encontrar patrones en nuestro conjunto de datos.

3.1 Descripción del Algoritmo

1. Escoger un número k de clusters a encontrar.
2. Escoger k ejemplos aleatorios del conjunto los cuales llamaremos nuestros *centroides* (Inicialicación Forgy).
3. Para cada ejemplo en el conjunto de datos, determinamos el centroide más cercano (utilizando una métrica de distancia) y les asignamos esa clase.
4. Tomamos el promedio de todos los elementos de cada clase y actualizamos los centroides.
5. Repetimos el proceso hasta que los centroides dejen de cambiar.

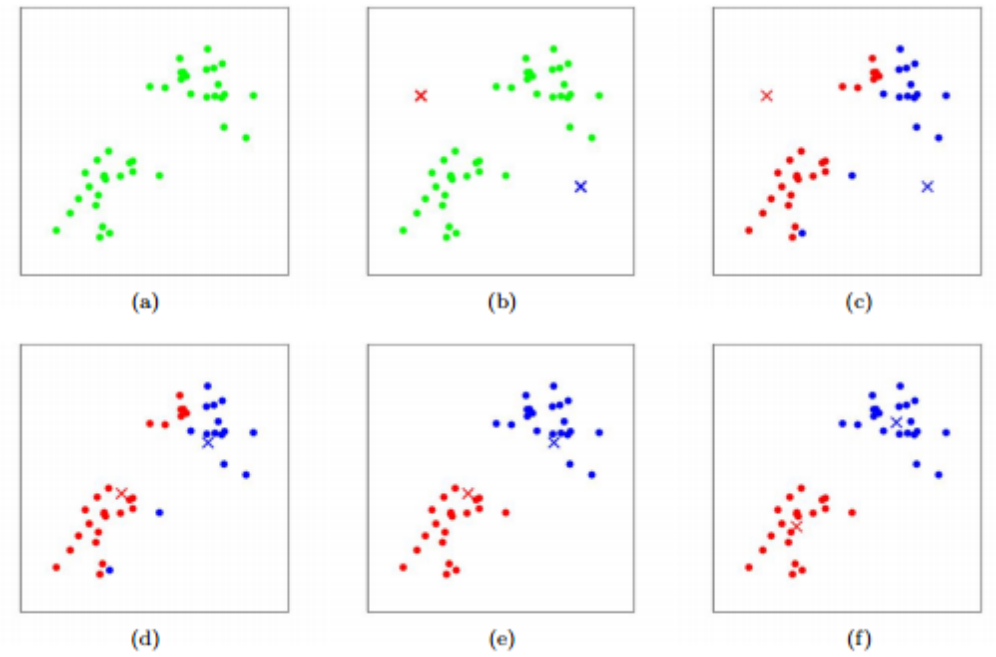

Figure 5: 6 Iteraciones de Kmeans

3.2 Mejoras

La versión del algoritmo descrita en el proceso anterior es la más sencilla. Algunos posibles cambios para el algoritmo incluyen.

- En lugar de inicializar los centroides escogiendo los ejemplos de manera aleatoria podemos: (1) (Random Partition): Empezar asignando un cluster de manera aleatoria a cada elemento del conjunto de datos, calcular así los centroides originales (2) (Kmeans ++): Escoger elementos que sean lo más lejanos posibles entre si.
- Cambiar la métrica de distancia.

3.3 Métrica de Distancia

Hablemos de las métricas de distancia del paso 3 del algoritmo. La decisión que hacemos aquí puede potencialmente cambiar nuestros resultados de manera drástica.

3.3.1 Distancia Euclideana

La métrica más común es la Distancia Euclideana. La Distancia Euclidiana no es más que el teorema de pitágoras entre dos puntos ejemplos de dos o más dimensiones.

Por ejemplo, sean a y b un par de ejemplos de de n dimensiones donde $a = [a_1, a_2, \ldots, a_n]$ y $b = [b_1, b_2, \ldots, b_n]$. La distancia euclideana entre a y b, $d(a, b)$ será,

$$d(a,b) = d(b,a) = \sqrt{(b_1 - a_1)^2 + (b_2 - a_2)^2 + \cdots + (b_n - a_n)^2} \tag{3}$$

3.3.2 Otras métricas de distancia

Algunas otras métricas de distancia incluyen,

- Manhattan: $D = \sum_{i=1}^{n} |a_i - b_i|$
- Chebyshev: $D = \max_i |x_i - y_i|$
- Minkowski de orden p: $D = \left(\sum_{i=1}^{n} |a_i - b_i|^p \right)^{1/p}$
- Cosenos (distancia angular) ($\mathbf{a}$ y $\mathbf{b}$ son vectores): $D = 1 - \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = 1 - \frac{\sum_{i=1}^{n} a_i \cdot b_i}{\sqrt{\sum_{i=1}^{n} a_i^2} \sqrt{\sum_{i=1}^{n} b_i^2}}$